ATIS-0100509.1995(R2013)

Packetized Circuit Multiplication Equipment – Interface Specification

AMERICAN NATIONAL STANDARD FOR TELECOMMUNICATIONS

As a leading technology and solutions development organization, ATIS brings together the top global ICT companies to advance the industry's most-pressing business priorities. Through ATIS committees and forums, nearly 200 companies address cloud services, device solutions, emergency services, M2M communications, cyber security, ehealth, network evolution, quality of service, billing support, operations, and more. These priorities follow a fast-track development lifecycle – from design and innovation through solutions that include standards, specifications, requirements, business use cases, software toolkits, and interoperability testing.

ATIS is accredited by the American National Standards Institute (ANSI). ATIS is the North American Organizational Partner for the 3rd Generation Partnership Project (3GPP), a founding Partner of oneM2M, a member and major U.S. contributor to the International Telecommunication Union (ITU) Radio and Telecommunications sectors, and a member of the Inter-American Telecommunication Commission (CITEL). For more information, visit < www.atis.org >.

AMERICAN NATIONAL STANDARD

Approval of an American National Standard requires review by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer.

Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that a concerted effort be made towards their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give an interpretation of any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

Notice of Disclaimer & Limitation of Liability

The information provided in this document is directed solely to professionals who have the appropriate degree of experience to understand and interpret its contents in accordance with generally accepted engineering or other professional standards and applicable regulations. No recommendation as to products or vendors is made or should be implied.

NO REPRESENTATION OR WARRANTY IS MADE THAT THE INFORMATION IS TECHNICALLY ACCURATE OR SUFFICIENT OR CONFORMS TO ANY STATUTE, GOVERNMENTAL RULE OR REGULATION, AND FURTHER, NO REPRESENTATION OR WARRANTY IS MADE OFMERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR AGAINST INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS. ATIS SHALL NOT BE LIABLE, BEYOND THE AMOUNT OF ANY SUM RECEIVED IN PAYMENT BY ATIS FOR THIS DOCUMENT, AND IN NO EVENT SHALL ATIS BE LIABLE FOR LOST PROFITS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES. ATIS EXPRESSLY ADVISES THAT ANY AND ALL USE OF OR RELIANCE UPON THE INFORMATION PROVIDED IN THIS DOCUMENT IS AT THE RISK OF THE USER.

NOTE - The user's attention is called to the possibility that compliance with this standard may require use of an invention covered by patent rights. By publication of this standard, no position is taken with respect to whether use of an invention covered by patent rights will be required, and if any such use is required no position is taken regarding the validity of this claim or any patent rights in connection therewith. Please refer to [http://www.atis.org/legal/patentinfo.asp] to determine if any statement has been filed by a patent holder indicating a willingness to grant a license either without compensation or on reasonable and non-discriminatory terms and conditions to applicants desiring to obtain a license.

ATIS-0100509.1995(R2013), *Packetized Circuit Multiplication Equipment Interface Specification*

Is an American National Standard developed by the **ATIS Network Performance, Reliability and Quality of Service Committee (PRQC)**.

Published by

Alliance for Telecommunications Industry Solutions
1200 G Street, NW, Suite 500
Washington, DC 20005

American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer.

Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that a concerted effort be made toward their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give an interpretation of any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

Published by

American National Standards Institute
11 West 42nd Street, New York, New York 10036

Printed in the United States of America

APS2.5C595/150

ANSI®
T1.509-1995
(R2013)

American National Standard
for Telecommunications –

Packetized Circuit Multiplication Equipment – Interface Specification

Secretariat

Alliance for Telecommunications Industry Solutions

Approved January 12, 1995

American National Standards Institute, Inc.

Abstract

The purpose of this American National Standard is to standardize the interface to packetized circuit multiplication equipment (PMCE). PMCE converts speech, voiceband data, facsimile, channel-associated (i.e., in-band) signaling, common-channel signaling, video, and digital data information from channelized DS1 or Synchronous Optical Network (SONET) formats to LAPD-like frame format.

A PCME is one application of Wideband packet technology. This technology refers to packet systems operating at 1.5 Mbit/s to 150 Mbit/s. T1.PCME defines procedures and interface standards to allow vendors to provide compatible equipment for the US marketplace and to permit both exchange and interexchange carriers to operate compatibly.

Contents

Foreword (This foreword is not part of American National Standard T1.509-1995.)

Contemporary transmission techniques are evolving according to three scenarios. In the first scenario, the various types of traffic are treated in a homogeneous manner with respect to transmission and switching. The second scenario calls for augmenting the number of available paths without affecting service quality by time division techniques. Finally, in the third scenario, which is often associated with the increasing digitization of networks, the information is packetized before transmission.

Wideband Packet Technology is founded on the three scenarios simultaneously. Although packet transmission is not new, the originality of the new approach resides in routing over the same path the digital signals corresponding to services of a different nature (speech, voiceband data, facsimile, signaling, etc.). Packetized circuit multiplication equipment (PCME), therefore, uses the techniques available for digital circuit multiplication, packet networks, and frame relay networks to transport a variety of traffic types.

The purpose of this American National Standard is to standardize the interfaces to PCME so that vendors can provide internetworking units for the United States and the international marketplace. Additionally, both exchange and interexchange carriers may be assured of compatible network interfaces and specific functional compatibilities. It is expected that both manufacturers and carriers will utilize this standard.

This standard is new; it does not replace, revise, or supplement any existing North American standard for PCME system interface or functional requirements.

There are six annexes to this standard. The first annex is normative; the others are informative, and are not considered part of this standard.

Suggestions for improvement of this standard are welcome. They should be sent to the T1 Secretariat, c/o Alliance for Telecommunications Industry Solutions, 1200 G Street, NW, Suite 500, Washington, DC 20005.

This standard was processed and approved for submittal to ANSI by Accredited Standards Committee on Telecommunications, T1. Committee approval of the standard does not necessarily imply that all members voted for its approval. At the time it approved this standard, the T1 committee had the following members:

A. K. Reilly, Chairman
G. H. Peterson, Vice-Chairman
O. J. Gusella, Jr., Secretary

B. Lerich, Senior Editor
M. Sharif, Technical Editor
R. Wright, Technical Editor

Organization Represented	*Name of Representative*
EXCHANGE CARRIERS	
Ameritech Services, Inc.	Laurence A. Young Peter K. Cencer (Alt.)
Bell Atlantic Corporation	John W. Seazholtz Roger Nucho (Alt.)
Bellcore	D. S. Jordan James C. Staats (Alt.)

Organization Represented	*Name of Representative*
BellSouth Telecommunications, Inc.	Leonard Strickland, Jr. William J. McNamara, III (Alt.)
Centel Corporation	Bruce Becker
Cincinnati Bell Telephone	William P. Keidel Kevin R. Sullivan (Alt.)
Exchange Carriers Standards Association	Joseph Mendoza Gregory L. Theus (Alt.)
GTE Telephone Operations	Gregory L. Theus Richard L. Cochran (Alt.)
National Telephone Cooperative Association	Joseph M. Flanigan
NYNEX	James F. Baskin Jim Papadopoulos (Alt.)
Pacific Bell	Steve Grimm Sal R. Tesoro (Alt.)
Puerto Rico Telephone Company	Segundo Ruiz Alberto E. Morales (Alt.)
Southwestern Bell Corporation	Joseph Mendoza C. C. Bailey (Alt.)
Sprint – Local Telecommunications Division	Robert P. McCabe Harold L. Fuller (Alt.)
US WEST	James L. Eitel James Dahl (Alt.)
GENERAL INTEREST	
Ashford Associates	Donald A. Ashford
Base-2 Systems, Inc.	Douglas M. Brady
BT North America, Inc.	Richard A. Rawson Michel J. Darnaud (Alt.)
C.S.I. Telecommunications	Michael S. Newman
Cable Television Labs, Inc.	Stephen D. Dukes James S. Meditch (Alt.)
Capital Cities/ABC, Inc.	Warner W. Johnston
Creative Communications Consulting	Richard Bobilin James Boe (Alt.)
Defense Information Systems Agency	C. Joseph Pasquariello Granger Kelley (Alt.)
GTE Mobile Communications	John C. Chiang Steve Pankow (Alt.)
International Communications Association	Edward F. Bonkowski Robert M. Eilers (Alt.)
National Communications System	Dennis Bodson Frank M. McClelland (Alt.)
National Institute of Standards and Technology	Robert Rountree, Jr. Michael D. Hogan (Alt.)
National Telecommunications and Information Administration/Institute for Telecommunication Sciences (NTIA/ITS)	William F. Utlaut Neal B. Seitz (Alt.)
NTT America, Inc.	Hideo Yamamoto Naobumi Kanemaki (Alt.)
Rural Electrification Administration	Donald M. Van Bellinger George J. Bagnall (Alt.)
U. S. General Services Administration	Douglas K. Arai Larry L. Jackson (Alt.)
Interexchange Carriers	
American Mobile Satellite Corporation	Michael K. Ward William Garner (Alt.)
AT&T Communications	Charles A. Dvorak Dennis Thovson (Alt.)
Comsat Corporation	Carl A. Sederquist Mark T. Neibert (Alt.)
MCI Telecommunications Corporation	Stephen J. Engelman Peter Guggina (Alt.)
Sprint – Long Distance Division	Thomas G. Croda Peter J. May (Alt.)
Telecom Canada	E. J. Exton Douglas I. Hughes (Alt.)

Organization Represented	*Name of Representative*
Unitel Communications, Inc.	David H. Whyte
	George Tadros (Alt.)
Vyvx, Inc.	Howard Meiseles
	Mark Elden (Alt.)
MANUFACTURERS	
ADC Telecommunications, Inc.	Jack P. Reily
	Don Berryman (Alt.)
AG Communication Systems	Nigel J. E. Reynolds
	J. C. Gibson (Alt.)
Alcatel Network Systems (ANS)	T. Rowley
	Kevin Pickles (Alt.)
Amdahl Corporation	Paul Lue
AMP, Inc.	George Lawrence
	Jack Bradbery (Alt.)
Apple Computer, Inc.	Karen Higginbottom
Applied Innovation, Inc.	Gerry Moersdorf
Ascom Timeplex, Inc.	D. Protopapas
	L. H. Eberl (Alt.)
AT&T Network Systems	Stanley W. Johnston
	Sigrid K. Llewellyn (Alt.)
Digital Equipment Corporation	Robert Ray
	Richard Hovey (Alt.)
DSC Communications Corporation	Peter Waal
	Allen Adams (Alt.)
ECI Telecom, Inc.	Ron Murphy
	Charles T. Throop (Alt.)
Ericsson, Inc.	Linda Troy
	Al Way (Alt.)
Fujitsu Network Switching	Steven A. Minneman
Fujitsu Network Trans Sys, Inc.	Rodney Boehm
General DataComm, Inc.	Frederick Cronin
	Frederick Lucas (Alt.)
Harris Corporation	Allen Jackson
	Yogi Mistry (Alt.)
Hekimian Laboratories	David R. Gellerman
	Mike F. Toohig (Alt.)
Hewlett-Packard	Don C. Loughry
	Richard van Gelder (Alt.)
IBM Corporation	Robert M. Amy
	Nicholas S. Huslak (Alt.)
Mitel Corporation	Keith Richardson
	John Needham (Alt.)
Mitsubishi Electronics America	Philip Jongeneel
Motorola, Inc.	David Morgan
	Gail Smith (Alt.)
NEC America, Inc.	Art Graham
	Donovan Nak (Alt.)
Northern Telecom, Inc.	Mel N. Woinsky
	Myron Allen (Alt.)
Novatel Communications, Ltd.	Allan Angus
Picturetel Corporation	Marshall Schachtman
	Antony Crossman (Alt.)
Racal-Datacom, Inc.	Donald O'Connor
Reliance Comm/Tec	Randall Summers
	Philip L. Dillon (Alt.)
Rockwell International Corporation	Thomas P. Jones
	Carl J. Stehman (Alt.)
Siemens Stromberg-Carlson	Michael A. Pierce
	Robert Poignant (Alt.)
Superior Teletec, Inc.	M. Farrant
	Brian Cole (Alt.)
Tekelec, Inc.	Willy M. Verbestel
	Ron Riegert (Alt.)
Telecom Solutions	M. J. Narasimha
	Robert Yapp (Alt.)
Telecommunications Techniques Corporation	Bernard E. Worne

Organization Represented	*Name of Representative*
Teleos Communications, Inc.	Hascall Sharp
	Ken Araujo (Alt.)
Tellabs Operations, Inc.	Charles Rohrs
	Michael J. Birck (Alt.)
Transwitch Corporation	John Wyatt
	Daniel C. Upp (Alt.)
Verilink Corporation	William J. Buckley
	Robert Beebe (Alt.)
Wandel & Goltermann	Glenn C. Dunlap
	Norm Christiansen (Alt.)

Technical Subcommittee T1A1, which developed this standard, had the following participants:

G. Williams, Chairman
C. A. Dvorak, Vice-Chairman
A. Niedzweicki, Secretary

Organization Represented	*Name of Representative*
Alcatel Network Systems (ANS)	Fred Ellefson
	Kevin Pickles (Alt.)
AT&T Communications	Anthony Schiano
	Charles A. Dvorak (Alt.)
AT&T Network Systems	J. H. James
	K.C. Glossbrenner (Alt.)
Base-2 Systems, Inc.	Douglas M. Brady
Bell Atlantic Corporation	Hank Hudson
	Max Roesch (Alt.)
Bellcore	Ralph E. Jensen
	Thomas C. Sprang (Alt.)
BellSouth Services	Hal B. Holton
	Gregory A. Wos (Alt.)
BT North America, Inc.	Richard A. Rawson
Canadian Broadcasting Corporation	Kenneth P. Davies
	Jean Aubry (Alt.)
Compression Labs, Inc.	Dan Klenke
	Padmanabha Rao (Alt.)
Creative Communications Consulting	Richard Bobilin
	James Boe (Alt.)
Defense Information Systems Agency	Granger Kelley
ECI Telecom, Inc.	Ron Murphy
	Yuval Shalev (Alt.)
General DataComm, Inc.	David Moon
	Frederick Lucas (Alt.)
GTE Telephone Operations	Michael L. Kanotz
	Norman Epstein (Alt.)
MCI Telecommunications Corporation	Michael Varrassi
	Morton Blanchard (Alt.)
Mitsubishi Electronics America	Philip Jongeneel
National Communications System	Gary Rekstad
National Telecommunications and Information Administration/Institute for Telecommunication Sciences (NTIA/ITS)	Neal B. Seitz
	Randall S. Bloomfield (Alt.)
NEC America, Inc.	Donovan Nak
Newbridge Networks Corporation	Zlatko Krstulich
	Francois Bessette (Alt.)
Northern Telecom, Inc.	Mel N. Woinsky
	Andy Niedzwiecki (Alt.)
NYNEX	John McDonough
	F. T. Burns (Alt.)
Pacific Bell	Ali Zolfaghari
	Stanley Chum (Alt.)

Organization Represented	*Name of Representative*
Picturetel Corporation	Antony Crossman
	Bob Reynolds (Alt.)
Racal-Datacom, Inc.	Donald O'Connor
Scientific-Atlanta. Inc.	Michael Maslaney
	David Fellows (Alt.)
Siemens Stromberg-Carlson	Gil Hassell
	Michael A. Pierce (Alt.)
Southwestern Bell Corporation	Hal Holzwarth
	Bill Litzinger (Alt.)
Tekelec, Inc.	Willy M. Verbestel
Telecom Canada	J.A. Zebarth
	Steve Corkovic (Alt.)
Telecommunications Techniques Corporation	Bernard E. Worne
Teleos Communications, Inc.	Hascall Sharp
	Ken Araujo (Alt.)
Tellabs Operations, Inc.	Maurice Givens
	Jim Mills (Alt.)
Unitel Communications, Inc.	David H. Whyte
US Sprint	James Lord
	Peter J. May (Alt.)
US Telephone Association (USTA)	Robert Creighton
US WEST	Kipling E. Ferguson
	Randon K. Sheridan (Alt.)
VYVX, Inc.	Howard Meiseles
	Mark Elden (Alt.)

Working Group T1Y1.2 (T1A1.6), which developed this standard, had the following participants:

Bob Kubichek, Chairman

D. J. Atkinson
Udaya Bhaskat
Bill Bigelis
M. K. Blanchard
Dick Boblin
Timothy Bower
Douglas M. Brady
Fred Burns
Jose Corleto
Antony Crossman
Ayse Dibler
Spiros Dimolitsas
Bodo Dunker
Michael Floyd
Dick Gillitzer
M. Givens
Hal Holzwarth
L. D. Humphrey
Ralph E. Jensen
Simon Lam
Meir Leiberman
Tze-Wo Leung
Imran Malik
J. D. Mills
Ron Murphy
Ming D. Ni
Michael Onufry
Chris Pratt
Zigmunds Putnins
Jack H. Rieser
John Rosenberger
G. Ferrell Sandy
Marshall Schachtman
Venkita Seshadri
Mostafa Hashem Sherif
Maninder Sohi
Ying T. Tao
C. Frank Taylor
Thomas Tremain
Willy Verbestel
Steve Voran
Phil Wilson
J. A. Zebarth

American National Standard
for Telecommunications –

(R2013)

Packetized Circuit Multiplication Equipment – Interface Specification

1 Scope, purpose, and application

1.1 Scope

This standard is intended as a base document for the specification and interconnection of packetized circuit multiplication equipment (PCME) and Packet Circuit Multiplication Systems (PCMS) from various manufacturers.

This standard covers the following interface and functional requirements for PCME:

- interface requirements (network-network and user-network), including
  - interface with Link Access Procedure D-Channel (LAPD);
  - interface with DS0-A circuits;
  - interface with V.120;
  - interface with cellular speech packetization networks and systems;
  - signaling interfaces;
  - Pulse Code Modulation (PCM)/Adaptive Differential Pulse Code Modulation (ADPCM); code switching for voiceband traffic;
  - PCM encoding standards conversion.
- functional requirements, including
  - build-out delay, echo control;
  - congestion control strategy;
  - maintenance and alarm functions;
  - testing procedures.
- networking applications requirements, including
  - configurations;
  - procedures at intermediate cross-connect nodes, including congestion strategies;
  - implementation and applications considerations.

1.2 Purpose

PCME converts speech, voiceband data, facsimile, channel-associated (i.e., in-band) signaling, common-channel signaling, video, and digital data information from channelized DS1 or Synchronous Optical Network (SONET) formats to LAPD-like frame format. The LAPD-like frames are transported as packet streams in a wideband packet network over a full or fractional DS1, or a SONET virtual tributary.

Wideband packet technology, as used herein, refers to packet systems operating at 1.5 Mbit/s to 150 Mbit/s. To streamline processing and to limit delay so as to accommodate voice, voiceband data, facsimile and video, this technology uses a LAPD-like link layer protocol with unacknowledged operation and application-specific layer 3 protocols. Digital data transmission is supported by several layer 3 protocols, as explained in this document.

1.3 Application

Wideband packet technology can be applied in public or private networks, virtual private networks, and satellite networks to achieve the following benefits:

- better utilization of long-haul digital lines by implementing low-bit-rate voice coding, silence elimination, and variable-bit-rate video coding;

– efficient sharing of the same network among voice, voiceband data, image, video, and data;

– automatic shifting of available transmission capacity between non-coincident traffic loads without time-of-day administrative procedures;

– tandem switch relief;

– automatic re-routing of traffic around failed links and nodes;

– uniform approach to dynamic load control and congestion control;

– common link testing procedures.

2 Normative references

The following standards contain provisions through which reference in this text constitutes provisions of this American National Standard. At the time of this publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this American National Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

ANSI T1.101-1994, *Telecommunications – Synchronization interface standards for digital networks*

ANSI T1.102-1993, *Digital Hierarchy – Electrical Interfaces*

ANSI T1.105-1991, *Digital Hierarchy – Optical Interface Rates and Formats Specifications (SONET)*

ANSI T1.111-1992, *Signaling System Number 7 (SS7) – Message Transfer Part (MTP)*

ANSI T1.303-1990, *Digital Processing of Voice-Band Signals – Algorithm for 24, 32, and 40 kbit/s ADPCM*[1)]

ANSI T1.306-1989, *Digital Processing of Audio Signals – Algorithm and Line Format for Transmission of 7 kHz Audio Signals at 64/56 kbit/s*

ANSI T1.310-1991, *Digital Processing of Voice-Band Signals – 5-, 4-, 3-, and 2-bits/sample Embedded Adaptive Differential Pulse Code Modulation*

ANSI T1.312-1991, *Voice Packetization – Packetized Voice Protocol*

ANSI T1.606-1990, *Frame Relaying Bearer Service – Architectural framework and service description*

ANSI T1.606a-1992, *Frame Relaying Bearer Service – Architectural framework and service description (congestion management and frame size)*

ANSI T1.618-1991, *DSSI – Core Aspects of Frame Protocol for Use with Frame Relay Bearer Services*

CCITT P Series Recommendations, Blue Book, Vol. V, 1988[2) 3) 4)]

CCITT Recommendation G.114, *Mean One-Way Propagation Time*, pp. 84-94[2) 3) 4)]

CCITT Recommendation G.165, *Echo Cancellers*, pp. 225-243[2) 3) 4)]

CCITT Recommendation G.703, *Physical/electrical characteristics of hierarchical digital interfaces*, September 1991[2) 3) 4)]

CCITT Recommendation G.704, *Synchronous frame structures used at primary and secondary hierarchical levels*, October 1991 [2) 3) 4)]

CCITT Recommendation G.711, *Pulse Code Modulation (PCM) of Voice Frequencies*, pp. 175-184[2) 3) 4)]

CCITT Recommendation G.722, *7 kHz audio-coding within 64 kbit/s*, pp. 269-340[2) 3) 4)]

CCITT Recommendation G.726, *40-, 32-, 24-, 16-kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)*, December 1990[2) 3) 4)]

CCITT Recommendation G.727, *5-, 4-, 3-, and 2-bit/sample Embedded Adaptive Differential Pulse Code Modulation*, December 1990[2) 3) 4)]

---

1) This standard has been withdrawn. However, it is available in archive format through the ANSI sales department.

2) Available from American National Standards Institute, Inc., 11 West 42nd Street, New York, NY 10036, (212) 642-4900.

3) Available from NTIS, Attn: Order Department, Springfield, VA 22161, (703) 487-4650.

4) Available from OMNICOM, Inc., 501 Church Street, NE, Suite 304, Vienna, VA 22180, (703) 281-1135

CCITT Recommendation G.728, *Coding of speech at 16 kbit/s using low-delay code excited linear predictions (LD-CELP)*, September 1992[2) 3) 4)]

CCITT Recommendation G.732, *Characteristics of Primary PCM Multiplex Equipment Operating at 2048 kbit/s*, pp. 375-381[2) 3) 5)]

CCITT Recommendation G.733, *Characteristics of Primary PCM Multiplex Equipment Operating at 1544 kbit/s*, pp. 381-384[2) 3) 4) 5)]

CCITT Recommendation G.764, *Packetized Voice Protocol*, December 1990[2) 3) 4) 5)]

CCITT Recommendation H.221, Rev. 1, *Frame structure for a 64 to 1920 kbit/s channel in audiovisual teleservices*, December 1990[2) 3) 4) 5)]

CCITT Recommendation H.261, Rev.1, *Video codec for audiovisual services at p × 64 kbit/s*, December 1990[2) 3) 4) 5)]

CCITT Recommendation H.320, *Narrow-band visual telephone systems and terminal equipment*, December 1990

CCITT Recommendation I.431, *Primary rate user-network interface – Layer 1 specification*, pp. 241-269[2) 3) 4) 5)]

CCITT Recommendation Q.140, *Definition and Function of Signals*, pp. 51-53[2) 3) 4) 5)]

CCITT Recommendation Q.141, *Signal Code for Line Signaling*, pp. 55-59[2) 3) 4) 5)]

CCITT Recommendation Q.151, *Signal Code for Register Signaling*, pp. 65-66[2) 3) 4) 5)]

CCITT Recommendation Q.310, *Definition and Function of Signals*, pp. 5-6[2) 3) 4) 5)]

CCITT Recommendation Q.311, *2600 Hz Line Signaling*, pp. 7-8[2) 3) 4) 5)]

CCITT Recommendation Q.314, *PCM Line Signaling*, p. 11[2) 3) 4) 5)]

CCITT Recommendation Q.315, *PCM Line Signal Sender (Transmitter)*, pp. 11-12[2) 3) 4) 5)]

CCITT Recommendation Q.400, *Definitions and Functions of Signals*, pp. 37-40[2) 3) 4) 5)]

CCITT Recommendation Q.411, *Line Signaling Code*, pp. 41-42[2) 3) 4) 5)]

CCITT Recommendation Q.412, *Clauses for Exchange Line Signaling Equipment*, pp. 42-48[2) 3) 4) 5)]

CCITT Recommendation Q.414, *Signal Sender*, pp. 48-50[2) 3) 4) 5)]

CCITT Recommendation Q.421, *Digital Line Signaling Code*, pp. 57-58[2) 3) 4) 5)]

CCITT Recommendation Q.442, *Pulse Transmission of Backward Signals A-3, A-4, A-6 or A-15.* pp. 97-99[2) 3) 4) 5)]

CCITT Recommendation Q.703, *Signaling Link*, pp. 51-123[2) 3) 4) 5)]

CCITT Recommendation Q.921/I.441, *ISDN user network interface - Data link layer specification*, pp. 19-141[2) 3) 4) 5)]

CCITT Recommendation T.4, *Standardization of group 3 facsimile apparatus for document transmission*, pp. 21-47[2) 3) 4) 5)]

CCITT Recommendation T.30, *Procedures for document facsimile transmission in the general switched telephone network*, pp. 77-174[2) 3) 4) 5)]

CCITT Recommendation V.2, *Power levels for data transmission over telephone lines*, pp. 7-8[2) 3) 4) 5)]

CCITT Recommendation V.21, *300 bits per second duplex modem standardized for use in the general switched telephone network, with V-series type interfaces with provision for statistical multiplexing*, pp. 65-69[2) 3) 4) 5)]

CCITT Recommendation V.29, *9600 bits per second modem standardized for use on point-to-point 4-wire leased telephone-type circuits*, pp. 215-228[2) 3) 4) 5)]

CCITT Recommendation V.120, *Support by an ISDN of data terminal equipment with V-series*

---

5) CCITT is now known as ITU-T, International Telecommunications Union-Telecommunications Standardization.

type interfaces with provision for statistical multiplexing, pp. 453-478[2) 3) 4) 5)]

CCITT Recommendation Q.922, *ISDN Data Link Layer Specification for Frame Node Bearer Services*[2) 3) 4) 5)]

ISO-3309-1984, *Information processing systems – Data communications – High-level data link control procedures – Frame structure*[2) 3) 4)]

3 Definitions and abbreviations

This standard includes the following definitions and abbreviations.

3.1 Definitions

This standard includes the following definitions:

3.1.1 adaptive differential pulse code modulation (ADPCM): ADPCM algorithms are compression algorithms that achieve bit rate reduction through the use of adaptive prediction and adaptive quantization.

3.1.2 block: A specific group of octets within a voice packet that is made up of bits of the same significance.

3.1.3 block dropping: A process by which one or more of the less significant bits of all the samples stored in a packet are dropped to alleviate congestion.

3.1.4 block dropping indicator (BDI): The field of the voice packet header that indicates the number of blocks that have been dropped and the maximum number that can be dropped.

3.1.5 build-out delay: The maximum variable transmission and processing delay that is permitted in a wideband network.

3.1.6 bursts: Periods of high energy content signals present in the access channel of a wideband network.

3.1.7 check sequence (CS): A 16-bit sequence in the last two octets of a frame (excluding flags) that offers a cyclic redundancy check (CRC). The CRC is derived over either the header in unnumbered information with header check (UIH) format frames or over the entire packet frame for unnumbered information (UI) frames (excluding flags).

3.1.8 coding type (CT) field: In a voice/voiceband data packet, a 5-bit sequence in the packet header that indicates the method of coding speech samples used at the originating endpoint before packetization.

3.1.9 congestion: The condition that exists in a network if the bandwidth needed for the instantaneous traffic exceeds the bandwidth available in the network.

3.1.10 data link connection identifier (DLCI): A 13-bit field that defines the destination address of a frame on a physical-link-by-physical-link basis.

3.1.11 digital circuit emulation (DICE) protocol: A wideband protocol used to transport digital data that arrive on the channelized side through a specific format containing idle codes and repetition of user's data.

3.1.12 digital speech interpolation: A process that takes advantage of inactive periods of a conversation to insert speech from other conversations and to remove silent periods.

3.1.13 embedded adaptive differential pulse code modulation (embedded ADPCM): Embedded ADPCM algorithms are ADPCM algorithms that quantize the difference between the input and the estimated signal into core bits and enhancement bits.

3.1.14 frame check sequence (FCS): A 16-bit Cyclic Redundancy Check sequence that is derived over an entire packet frame (excluding flags) of a UI format packet.

3.1.15 gap: A period of low energy content signals present of a digital speech interpolation device.

3.1.16 header check sequence (HCS): A 16-bit cyclic redundancy check that is derived from the first 8 octets (excluding flags) of a UIH format packet.

3.1.17 idle code A special sequence that indicates that no data are being sent on the channelized side.

3.1.18 more (M) bit: A bit used to indicate that more packets in sequence are to be expected by the terminating endpoint.

3.1.19 originating endpoint: In a wideband packet node, the point that receives channelized traffic, packetizes it, and sends it into the wideband packet network.

3.1.20 packet header: Consists of octets 4 to 8 (inclusive) of the frame (flags excluded from the octet numbers).

3.1.21 packetization interval: The duration of the channel-oriented traffic that has been packetized.

3.1.22 packetized circuit multiplication equipment (PCME): A general class of equipment that compresses and integrates voice, voiceband data, digital data, signaling, image, facsimile, and network control into packets of common formats at the wideband range of bit rates greater than 64 kbit/s and less than the broadband rates of 150 Mbits/s.

3.1.23 packetized circuit multiplication system: A telecommunications network comprising two or more PCME nodes.

3.1.24 packet stream: A collection of logical links multiplexed together onto one physical channel between two endpoints of the wideband packet network.

3.1.25 protocol discriminator (PD) field The first octet of the packet header that identifies the protocol used to transport the frame.

3.1.26 scheduled play-out time: The time at which a received packet is to be played out.

3.1.27 sequence number (SEQ): In a packet, a field of the packet header that is used by the terminating endpoint to determine if the packets arrive in sequence.

3.1.28 signaling transition: In channel-associated signaling, a change in state of the A bit for 2-state signaling, A and/or B bit in 4-state signaling, or the A, B, C, and/or D bit for 16-state signaling.

3.1.29 terminating endpoint: In a wideband packet node, the part of the node that receives packetized traffic, depacketizes it, and then plays it back as channelized traffic.

3.1.30 time stamp (TS): A field that records the cumulative variable queuing delay experienced by a packet in transversing the network with a resolution of 1 ms.

3.1.31 unnumbered information (UI) frames: A frame used to transfer unacknowledged information between two link layer entities. The format and encoding are the same as specified in Recommendation Q.921/I.441. The CRC is derived over the entire frame.

3.1.32 unnumbered information with header check (UIH) frame: Similar to the UI frame, except that the CRC sequence is derived over the frame and packet headers (the first 8 octets excluding flags) rather than over the entire frame.

3.1.33 virtual data link capability (VDLC) protocol: A wideband protocol that is used to transport digital data packets arriving from the channelized side in HDLC frames.

3.2 Abbreviations

This standard includes the following abbreviations:

ADPCM	adaptive differential pulse code modulation
AIS	alarm indication signal
BDI	block dropping indicator
BER	bit error ratio
BILO	bits in last octet
CFR	Confirmation to Receive
CIR	Committed Information Rate
CRC	cyclic redundancy check
CRP	Command Repeat
CS	check sequence
CT	coding type
CTC	Continue to Correct
CTR	Response to Continue to Correct
DCS	Digital Command Signal
DICE	Digital Circuit Emulation
DIS	Digital Identification Signal
DLCI	data link connection identifier
DMC	digital modem class
DTC	Digital Transmit Command
EOM	End of Message
EOP	End of Procedure
EPT	Echo Protection Tone

EQ	equalization
ERR	End of Retransmission
ES	errored seconds
FADCOMP	Facsimile Demodulation and Compression Protocol
FCS	frame check sequence
FTT	Failure to Train
HCS	header check sequence
HDLC	High-Level Data-Link Control
IBT	idle background type
ISC	International Switching Center
LAPD	Link Access Procedure D-Channel
MCF	Message Confirmation
MPS	Multipage Signal
NSC	Nonstandard Facilities Command
NSS	Nonstandard Setup
OA&M	Operations, Administration, and Maintenance
OFF	inhibit information transfer
OOF	out of frame
PCM	Pulse Code Modulation
PCME	packetized circuit multiplication equipment
PCMS	Packetized Circuit Multiplication System
PIN	Procedure Interrupt Negative
PIP	Procedure Interrupt Positive
PPR	Partial Page Request
PPS	Partial Page Signal
PRI	Procedure Interrupt
PVC	permanent virtual circuit
PD	protocol discriminator
PVP	Packetized Voice Protocol
RTN	Retrain Negative
RTP	Retrain Positive
SC	sub-class
SDH	Synchronous Digital Hierarchy
SEQ	sequence number
SES	severely errored seconds
SONET	Synchronous Optical Network
SSEQ	send sequence variable
STM	Synchronous Transport Module
TCF	training check
TS	time stamp
UI	unnumbered information
UIH	unnumbered information with header check
VDLC	virtual data link capability

4 General considerations of packetized circuit multiplication equipment

PCME provides for the compression and packetization of several types of traffic A PCME converts speech, voiceband data, facsimile, channel-associated (i.e., in-band) signaling, common-channel signaling, video, and digital data information from primary rate channel formats or Synchronous Optical Network (SONET) formats to Link Access Procedure D-Channel (LAPD)-like frame format. The LAPD-like link layer protocol is used with unacknowledged operation to limit delay in the network. The LAPD-like frames are transported as packet streams in a wideband packet network over a full or fractional primary channel, or a SONET virtual tributary

Wideband packet technology, as used herein, refers to packet systems requiring transmission channels capable of supporting rates above 64 kbit/s up to 150 Mbit/s Application-specific protocols at layers 3 and above are used to transport the various types of traffic. A functional representation of a PCME node is shown in figure 1.

On the channel-oriented (full-rate) side, 1544 kbit/s and/or 2048 kbit/s, or SONET STS-1 interfaces are provided. A time slot interconnect function can connect any time slot or group of time slots to the proper processing function as required to packetize the incoming channel-oriented traffic. A frame cross-connect function transfers the layer 2 frames produced by the processing functions to the appropriate packetized side interface. On the packetized side, 1544 kbit/s and/or 2048 kbit/s or SONET STS-1 interfaces are provided to carry the bit-serial packet streams. Interfaces for higher optical synchronous rates are for further study The compatibility with the rates of the Synchronous Digital Hierarchy (SDH) Level 1 is also for further study

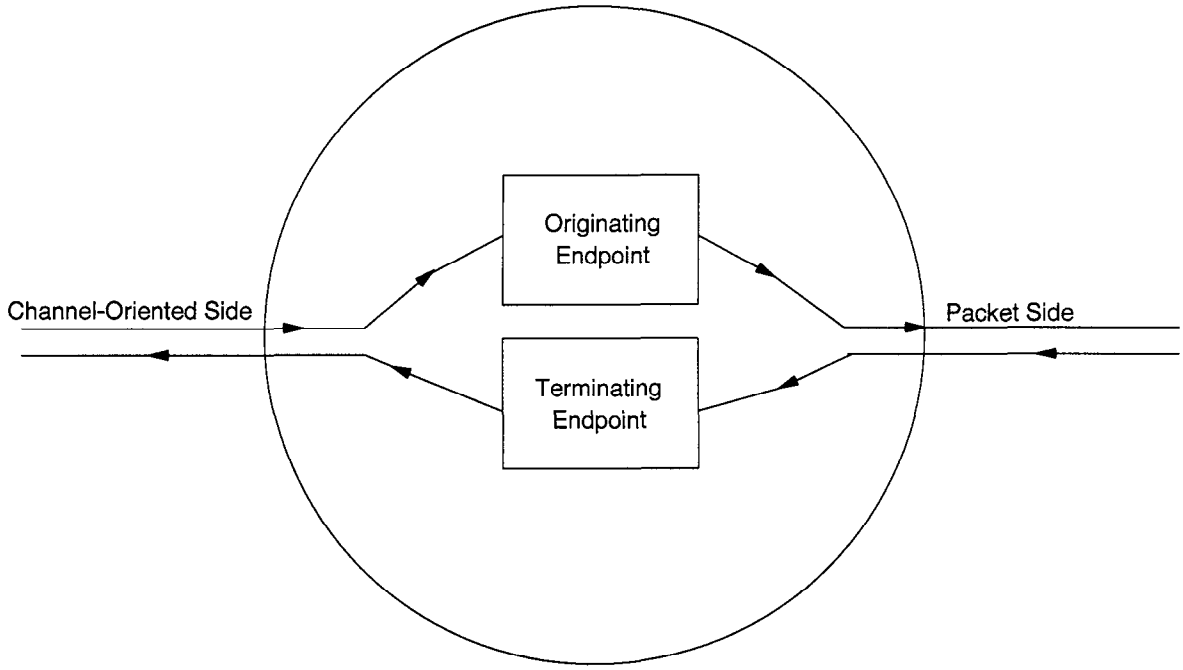


Figure 1 – Endpoint node

PCME allows networking in both the circuit and packet domains, offers bandwidth on demand, and achieves graceful degradation of voice quality during congestion. A reference model for a PCME network is shown in figure 2.

4.1 Speech processing

For speech traffic, the input speech samples may be coded at the originating endpoint of the access node before packetization by one of several coding methods. The stream of coded speech is transformed into packets with the format specified in ANSI T1.312. The samples are collected over a period of 16 ms and divided into blocks, as defined in ANSI T1.312. The blocks are arranged to facilitate block dropping, as explained below.

Periods of activity and inactivity are respectively called "bursts" and "gaps." It is not necessary to transmit packets during gaps and silent intervals may be removed.

The terminating endpoint at the far side reconstructs a continuous stream of speech from the incoming packets using the information in the packet header: the time stamp value and the sequence number. The Time Stamp field stores the value of the total accumulated variable delay that a packet has experienced. Each node adds to the time stamp value, the time it took to serve the packet using its local clock as a reference.

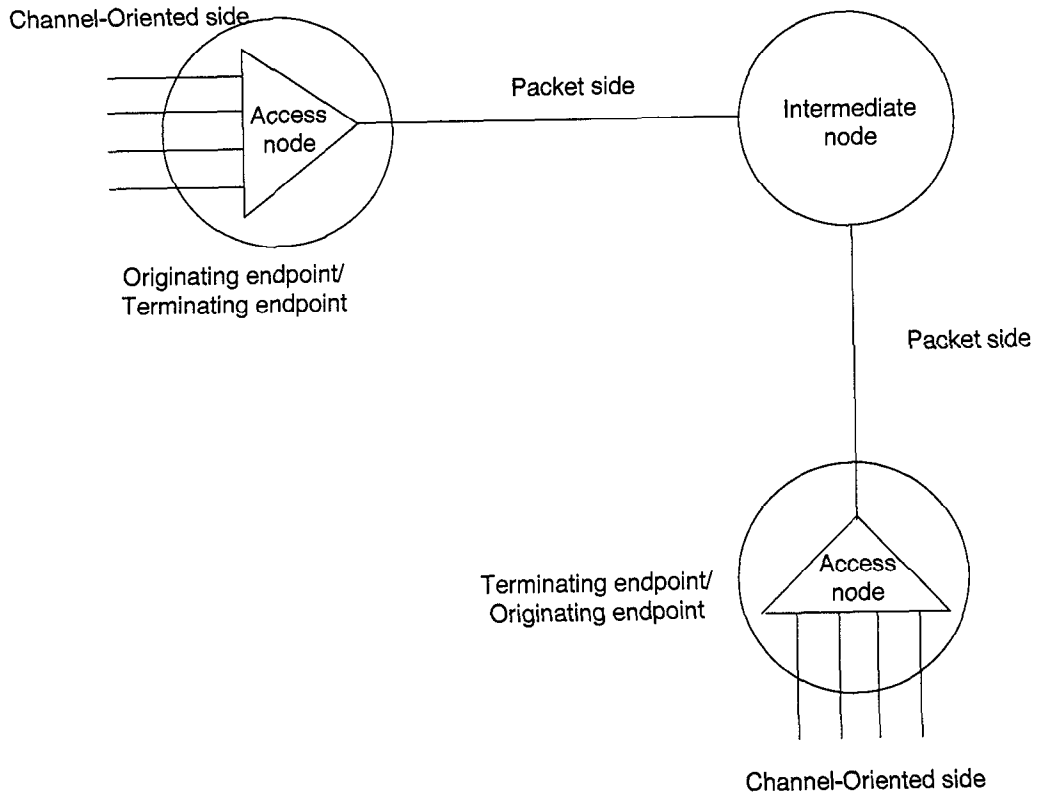


Figure 2 – Reference model for a PCME network

Build-out procedures compensate for the variable delay that packets may experience within the network. These procedures apply for the first packet of a voiceband spurt, for all signaling packets, and for the first packet after a missing packet is detected. When any of the above packets arrive with the time stamp value less than the value of the build-out, they are stored for the following duration:

Time before play-out = Build-out - Time Stamp Value

If they arrive with a time stamp value that exceeds the Build-out, they are discarded. Other packets are placed in the play-out queue in the order of their arrival and are played without interruption after the preceding packet (6.3.3.2 of ANSI T1.312).

The value of the build-out is a trade-off between accepting excessively large delay and having a large number of discarded packets (see 6.3 - Note 2).

The voice packet header contains information about the level of noise that was measured by the originating endpoint. The terminating endpoint uses this information to play out a matching noise level.

PCME has the additional ability of dropping blocks from a speech packet as a congestion control mechanism. The *n*th block consists of the *n*th bit from each sample collected during the sampling interval. The packet header indicates the number of droppable blocks contained in the packet. Congested nodes may use this information to drop the least significant blocks from packets to abate the congested state.

In general, the speech processing functions include echo cancellation, speech detection, signal classification, embedded Adaptive Differential Pulse-Code Modulation (ADPCM) coding, speech packetization, speech depacketization, variable rate ADPCM decoding, and noise fill. Echo cancellation will normally be required because the speech packet length is 16 ms. Speech detection and noise fill improve transmission efficiency by allowing the transmission of speech packets only when speech signals are present. Signal classification provides for the detection of voiceband data signals and the estimation of their speed, so that an appropriate coding algorithm can be chosen. Speech packetization and depacketization provide for the proper time stamping and delay build-out of packets within a talk spurt, so that gaps do not occur. These processes also control the effects of missing or excessively delayed packets.

4.2 Channel-associated signaling

The signaling associated with each voiceband connection is conveyed in signaling packets. Signaling packets are sent separately on a different logical channel.

A signaling transition is defined for channel-associated signaling as a change in state of the A bit for 2-state signaling, A and/or B bit in 4-state signaling, or the A, B, C, and/or D bit for 16-state signaling. The channel-associated signaling process detects changes in the state of A, AB, and ABCD signaling, and creates packets to convey the changed state to the distant PCME (see 14.2).

4.3 Common-channel signaling

Common channel signaling may be physically or logically out of band.

4.3.1 Common-channel signaling (out-of-band physical)

The signaling associated with each incoming connection is transported on a separate physical channel as defined in CCITT Recommendation G.704. The signaling information is conveyed in signaling frames flowing on a separate logical address (see 14.3).

4.3.2 Common-channel signaling (out-of-band logical)

The signaling associated with each incoming connection is transported via a separate logical channel, as in some of the Additional Packet Mode protocol bearer services. In this case, the PCME shall frame relay the information over a separate logical channel.

4.4 Digital data

Digital data can be transported by using any of the following:

- a Digital Circuit Emulation (DICE) protocol for the transport of special circuits in a bit transparent manner;
- a High-Level Data-Link Control (HDLC) specific procedure;

– an LAPD specific procedure;

– a V.120 specific procedure.

Interconnection with digital data on 64-kbit/s clear channels may involve ways to increase the compression gain, such as removing HDLC flags and/or network idle codes.

4.5 Facsimile

For Group 3 facsimile transport, the V.21 handshake signals may be transported as voiceband data in voice packets defined in ANSI T1.312. The coded page information is demodulated to extract the baseband signals, which are transmitted in the facsimile frames described in clause 14.

There are three types of facsimile frames:

– facsimile capability indication frames;

– spurt header frames that contain modem control information;

– facsimile page information frames that contain the T.4 coded image information in unscrambled format.

The PCME at the terminating end recombines the facsimile frames to reconstruct the original facsimile signal.

4.6 Video transport

CCITT Recommendation H.320 covers the technical requirements for narrow-band visual telephone services. It includes use of Recommendation H.261 for video coding, the Recommendation H.221 frame structure, and Recommendations G.711, G.722, and G.728 for audio coding. It is conceivable that the above recommendations, or a subset of these recommendations, could be applied in a packetized mode and transmitted over a wideband packetized network. Packetizing the video would allow improved picture quality by enabling the transmission of more or fewer bits as desired to maintain quality. Alternatively, a new set of video compression algorithms may be necessary, designed with packetized transport in mind. Furthermore, capabilities for interfacing packetized video with existing applications using 56-bit/s services must be contemplated.

The transmission of video and audio/videotelephony services in packetized networks is for further study.

5 Interfaces

5.1 1544 kbit/s interfaces

5.1.1 Physical interface

The physical interface conforms to the specifications in ANSI T1.102 (clause 2 of G.703).

5.1.2 Frame structure

The basic frame structure is shown in 2.1 of CCITT Recommendation G.704. The characteristics of the frame structure carrying channels at various bit rates in 1544 kbit/s is given in clause 3 of CCITT Recommendation G.704.

5.2 2048 kbit/s interface

5.2.1 Physical interface

The physical interface conforms to the specifications in ANSI T1.102 (clause 6 of CCITT Recommendation G.703).

5.2.2 Frame structure

The basic frame structure is shown in 2.3 of CCITT Recommendation G.704. The characteristics of the frame structure carrying channels at various bit rates in 2048 kbit/s is given in clause 5 of CCITT Recommendation G.704. Bit 1 of the frame can be used in accordance with 2.3.3 of CCITT Recommendation G.704 for a cyclic redundancy check (CRC) procedure.

5.3 SONET STS-1

SONET STS-1 may be supported. The primary level signals shall be mapped as virtual tributaries (VT-15 for 1564 kbit/s or VC-2 for 2048 kbit/s) into an STS-1 (51.840 Mbit/s) as described in ANSI T1.105. The electrical interface characteristics for STS-1 shall conform with ANSI T1.102.

5.4 Channel-oriented side interconnection with the packetized side

By service administration, it shall be possible to treat any time slot, or contiguous group of time slots, on the channel-oriented interface as a single channel-oriented circuit, and to connect that circuit to functions that provide packetization for:

– speech and voiceband data;

– signaling;

– facsimile;

– video;

– data.

5.5 Packetized-side interconnection

It shall be possible to direct any flow of packets on a channel-oriented side circuit to any packet stream on the packetized side.

Any packet stream from the packetized side may be interconnected to any other packet stream to achieve a packet cross-connect function.

6 Voiceband processing

6.1 Echo cancellation

An echo canceller shall be available for each speech and voiceband data circuit. The echo canceller shall conform to the requirements of CCITT Recommendation G.165.

As an objective, the echo canceller processing range may be a per-circuit administerable parameter.

6.2 Signal classification

The PCME shall provide for signal classification as a per-circuit administerable parameter. For example, the PCME may include an automatic signal classifier to classify the voiceband signal as:

– voiceband data at 7200 bit/s or higher rates;

– voiceband data at less than 4800 bit/s;

– voiceband data at 1200 to 2400 bit/s;

– voiceband data at less than 1200 bit/s;

– speech.

6.3 Speech packetization

Speech packetization shall be done according to ANSI T1.312. As an example in a representative application, voiceband data at rates exceeding 9600 bit/s shall be carried in Pulse-Code Modulation (PCM). Voiceband data at rates greater than 7200 bit/s and less than or equal to 9600 bit/s shall be encoded using the 40-kbit/s fixed rate algorithm of ANSI T1.303. Voiceband data at 1200 to 4800 bit/s shall be encoded using the 32-kbit/s fixed rate algorithm of ANSI T1.303. Voiceband data at less than 1200 bit/s shall be encoded as for speech, using the 24-kbit/s fixed rate algorithm of ANSI T1.303.

Decoding shall be done using the algorithm indicated in the ANSI T1.312 header.

NOTE 1 – In a national network, the time stamp (TS) and the build-out procedures of ANSI T1.312 may be replaced by a fixed-delay for the first packet. The fixed additional delay for the first packet of a speech burst shall be administerable to accommodate transit times through networks or through interconnection nodes not implementing TS updating. At the originating endpoint, the TS will be set to zero. In an intermediate node, the TS field will not be updated. However, the build-out procedure will be used always on network-network and user-network interfaces.

NOTE 2 – To meet the allowable transmission delay given in CCITT Recommendation G.114, the build-out shall be selected such that the total delay (propagation + build-out) shall not exceed the 400 ms allowance for voice traffic. The build-out should not exceed 70 ms in normal conditions.

6.3.1 ADPCM/PCM transcoding

A specific incoming channel may be defined by administration to correspond to information containing μ-law or A-law voiceband samples.

6.3.2 PCM operation

ANSI T1.312 specifies that the ADPCM coder on both sides shall be reset at the beginning of every speech burst. This is the case when all samples for a speech burst are coded with the same ADPCM algorithm specified in the coding type field. However, ANSI T1.312 does not specify the action of the ADPCM coders when the part of the traffic is coded according to CCITT Recommendation G.711.

When an originating PCME transports the incoming channel-oriented traffic in PCM format, it shall arrange the PCM coded speech as described in ANSI T1.312. The resultant voice packets shall have the format described in figure 2 of ANSI T1.312 and shall be transmitted on the packetized side of the endpoint.

The ADPCM encoder at the originating endpoint shall operate on the incoming PCM signal using the 40-kbit/s fixed rate ADPCM algorithm of ANSI T1.303, even though its ADPCM output is not transmitted. The ADPCM codec at the terminating endpoint shall operate as an encoder in an identical manner as the ADPCM codec of the originating endpoint, and its ADPCM output shall not be transmitted. Thus, when there are no line errors, the state variables of both ADPCM coders that are described in table 5 of ANSI T1.303 and table 7 of ANSI T1.310 shall be identical for both

endpoints. Furthermore, both ADPCM codecs shall be tracking the input PCM signal and would be ready to switch to their respective normal operations without introducing adaptation gaps.

When the originating endpoint PCME begins coding the incoming traffic, using the ADPCM algorithm indicated in the coding type field of the packet header, the ADPCM coded speech shall be arranged as described in figure 7 of ANSI T1.312 and then transmitted in voice packets with the formats shown in figure 2 of ANSI T1.312. At the terminating endpoint, the ADPCM codec shall decode the speech, using the ADPCM algorithm indicated in the coding type field to reconstruct the original signal, and then convert it to PCM, as indicated in ANSI T1.303 and T1.310. In the case of the embedded ADPCM algorithms of ANSI T1.310, the decoder at the terminating endpoint shall also use the information coded in the block dropping indicator field to select the decoding algorithm to use.

7 Video processing

The PCME shall implement packetization of fixed-rate and variable-rate video for teleconferencing and videotelephony. The specific procedures are for further study.

8 Interface with cellular speech packetization networks

This item is for further study.

9 Digital data interface

The PCME shall be equipped to receive, packetize and transmit the following digital data traffic arriving on the channel-oriented side as:

- special traffic, using the DICE protocol and the Virtual Data Link Capability (VDLC) for bit transparent transport;
- asynchronous and synchronous traffic, using V.120;
- digital data traffic, using frame relay or LAPF.

In some national applications, the PCME shall be equipped to receive, packetize and transmit digital data from the channel-oriented side to operate at either 56 or 64 kbit/s, or at the subrates of 2400, 4800, or 9600 bit/s. The polarity of the data bit can be either normal or inverted, as determined by administration.

As an objective, the PCME may be equipped to packetize and transmit digital data systems for other national applications.

As an objective, the PCME may be equipped to receive, packetize, and transmit digital data from groups of time slots on the channel-oriented side at rates of n × 64 kbit/s. The method of packetization in this case is for further study.

10 Digital circuit emulation (DICE) protocol

The Digital Circuit Emulation (DICE) protocol may be used to transport digital data that arrive on the channel-oriented side. DICE builds upon the same physical, link, and packet layers of ANSI T1.312 to increase efficiency of use of available bandwidth by (a) eliminating the redundant transmission of terminal and network idle codes, thereby releasing bandwidth for utilization, and (b) eliminating the transmission of redundant copies from subrate channels.

The DICE protocol consists of a physical layer, a link layer, a packet layer, and a high layer. Subclauses 10.1, 10.2, and 10.3 describe the physical, link, and packet layers by referring to the description of ANSI T1.312. Subclause 10.4 describes DICE procedures for the high, packet, and link layers of the originating endpoint. Subclause 10.5 describes the procedures at the intermediate nodes, while 10.6 describes the procedures at the terminating endpoint.

10.1 Physical layer

The physical layer is the same as for ANSI T1.312 (see 4.1 of ANSI T1.312).

10.2 Link layer

The link layer is the same as described in 4.2 of ANSI T1.312. In particular, the address field is the same as described in 4.2.1 of ANSI T1.312. Figure 3 depicts the format of a DICE frame. The control field of the unnumbered information (UI) frame is described in CCITT Recommendation

Q.921/I.441. The control field of the UIH frame is described in 4.2.3.2 of ANSI T1.312.

10.3 Packet layer

The packet layer procedures apply to the information transfer phase only. Call control procedures are outside the scope of this standard.

There are two types of DICE packets: DICE information packets and DICE idle update packets. DICE idle update packets update the idle code that the terminating endpoint should play to the channel-oriented side in the absence of information packets. Both packets have the format shown in figure 3 except that idle update packets shall have zero-length DICE information field and shall have the sequence number (SEQ) set to 0.

10.3.1 Protocol discriminator (PD)

The format and encoding of the protocol discriminator (PD) field are the same as the voice packet format (see 4.3.1.1 of ANSI T1.312).

10.3.2 Block dropping indicator (BDI)

The format of the block dropping indicator (BDI) field is the same as in the voice packet (see 4.3.1.2 of ANSI T1.312). Because the DICE packets are not block droppable, both the C-subfield and the M-subfield are set to 0.

10.3.3 Time stamp (TS)

The format of the TS field is the same as the voice packet format (see 4.3.1.3 of ANSI T1.312).

10.3.4 M-bit

The more (M) bit shall always be set to 1 (see 4.3.1.5 of ANSI T1.312).

10.3.5 Sub-class (SC) field

The sub-class (SC) field is used to indicate that the packet is a digital data packet. The SC field is coded as 11 for digital data.

10.3.6 Control (C) bit

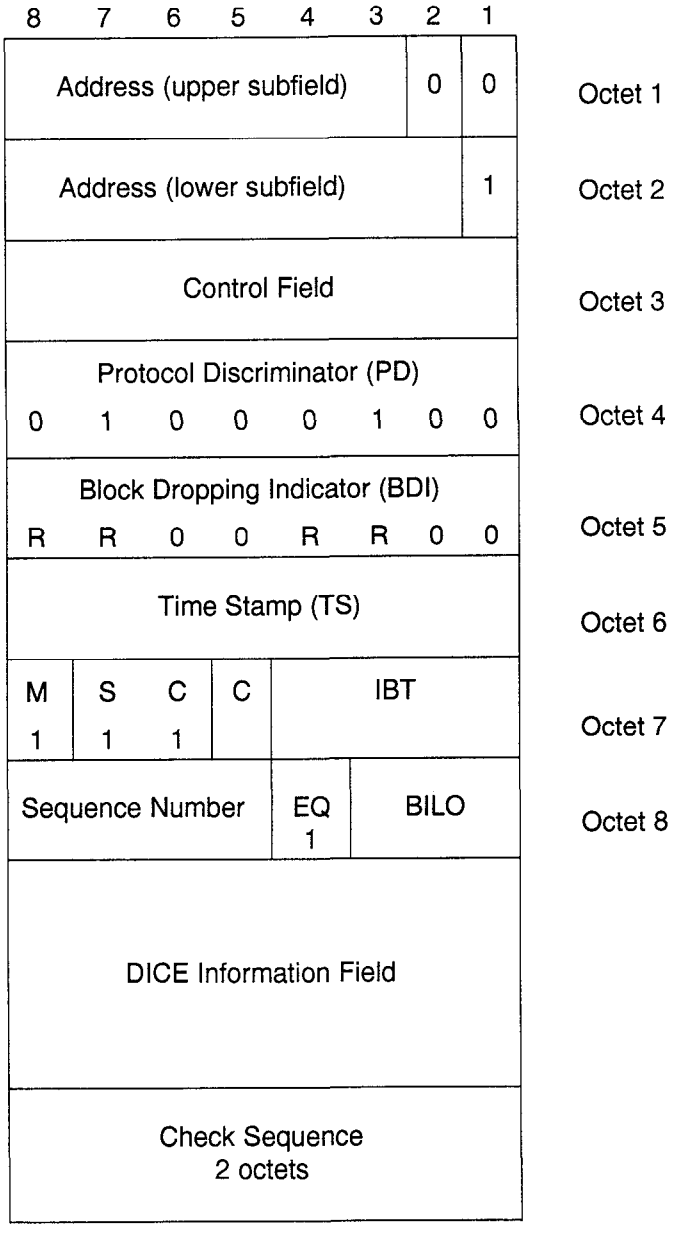
The control (C) bit is set by the originating end to indicate whether the last bit to arrive for each octet has been removed. In some national applications, this bit is called the control bit and is used for synchronization, status and remote testing on the channel-oriented side. C is set to 0 if the control bit is removed; otherwise C is set to 1.

10.3.7 Idle background type (IBT)

The idle background type (IBT) indicates the type of idle code that has been removed from the original data stream by the originating endpoint. An idle code is a sequence that indicates that no data are being sent on the channel-oriented side. The type of idle codes to be recognized is determined by service administration.

In the absence of packets to be played out, the IBT value of the last received packet shall be used to determine the idle codes to be played out on the channel-oriented side.

The terminal idle code is the all 1 octet. For 64 kbit/s, the terminal idle code may be inverted to the all 0 octet. The network idle code depends on whether the channel-oriented circuit is used in a point-to-point connection or in a multipoint connection. In a multipoint connection, two idle codes are possible: 1111 1111 or 1111 1110. Table 1 shows the relation between the idle code and the coding of the IBT field at the originating endpoint.



M = M-bit
R = Reserved for future use and set to 0

Figure 3 – DICE frame format

Table 1 – Correspondence between the channel-oriented idle code and the IBT field at the originating endpoint

Bit Rate (kbit/s)	Idle Code		IBT		
	Bit Pattern	Meaning	Point-to-Point	Multipoint	
	Bit 1 ... Bit 8			Option 1	Option 2
64	0000 0000	Normal	1000	1000	1000
	1111 1111		1111	1111	1111
	1111 1110		1110	1110	1111
56	1111 1111	Normal	1111	1111	1111
	1111 1110		1110	1110	1111
	X001 1110	Failure	1101	1110	1111
	X001 1000		1100	1110	1111
	X001 1010		1010	1110	1111
	X000 0000		1000	1110	1111
Subrate	0111 1111	Normal	1111	1111	1111
	0111 1110		1110	1110	1111
	0001 1110	Failure	1101	1110	1111
	0001 1000		1100	1110	1111
	0001 1010		1010	1110	1111
	0000 0000		1000	1110	1111
All Rates		No idle	0111	1110	1111

NOTE – X may be 0 or 1

The first bit of the idle code arriving on the channel-oriented side is the leftmost bit of the idle code bit pattern. It is represented as bit 1 in table 1 and is the most significant bit of the idle code bit pattern. In some national networks, bit 1 is called the "subrate framing bit" while bit 8 is called the "control bit."

The failure network idle codes are defined in table 2.

Table 2 – Failure network idle codes

Code	Meaning
Bit 1 ... Bit 8	
X001 1110	Abnormal station code
X001 1010	Multiplexer out of sync
X001 1000	Unassigned multiplexer channel
X000 0000	Zero code suppression

The abnormal station code indicates one of the following:

– a failure in the user's subrate or 56-kbit/s equipment;

– a failure in the local loop in the direction of transmission towards the network;

– removal of the far end equipment.

The multiplexer out of sync and unassigned multiplexer channel codes indicate failure within the network. The "No idle" IBT represents the case of a user that is not subscribed to any of the corresponding idle codes.

The initial value of the IBT field is 1111 (decimal 15). The IBT value is updated as indicated in 10.4.2.

At the terminating endpoint, the IBT field is translated into a bit pattern on the channel-oriented side, as indicated in table 3. In this table, options 1 and 2 of the multipoint connections have been combined.

Table 3 – Correspondence between the IBT field and the channel-oriented idle code at the terminating endpoint

Bit Rate (kbit/s)	Meaning Normal	IBT Code Point-to-Point	Channel-Oriented Bit Pattern	IBT Code Multipoint	Channel-Oriented Bit Pattern
			Bit 1 ... Bit 8		Bit 1 ... Bit 8
64	Normal	1000 1111 1110	0000 0000 1111 1111 1111 1110	1000 1111 1110	0000 0000 1111 1111 1111 1110
56	Normal	1111 1110	1111 1111 1111 1110	1111 1110	1111 1111 1111 1110
	Failure	1101 1100 1010 1000	1001 1110 1001 1000 1001 1010 1000 0000	1111 1111 1110 1110	1111 1111 1111 1111 1111 1110 1111 1110
Subrate	Normal	1111 1110	0111 1111 0111 1110	1111 1110	1111 1111 1111 1110
	Failure	1101 1100 1010 1000	0001 1110 0001 1000 0001 1010 0000 0000	1111 1110 1111 1110	1111 1111 1111 1110 1111 1111 1111 1110
All Rates	No idle	0111	0010 0001	1110 1111	1111 1110 1111 1111

NOTE – X may be 0 or 1

The first bit to be played out on the channel-oriented side is the leftmost bit for idle code bit patterns. It is represented as bit 1 in table 3 and is the most significant bit of the idle code pattern. In some national networks, bit 1 is called the "subrate framing bit" while bit 8 is called the "control bit."

10.3.8 Sequence number (SEQ)

The format of the SEQ field for the DICE packet is the same as for the voice packet (see 4.3.1.6 of ANSI T1.312).

10.3.9 Delay equalization bit (EQ)

The delay equalization (EQ) bit is set by the originating endpoint to enable or disable the build-out delay procedures at the terminating endpoint. When the EQ bit is set to 1, build-out procedures are enabled; otherwise, they are disabled. It is always set to 1 for DICE.

10.3.10 Bits in last octet (BILO)

LAPD specifies that frames should be octet-aligned, and therefore, the last octet of information shall be padded if necessary by a number of 1's to ensure this alignment. The bits in last octet (BILO) field contains the number of usable bits in the last information octet (i.e., that are valid data bits) to separate them from those that have been padded to the last octet. The valid BILO values are given in table 4.

Table 4 – BILO field format

Bit Number 321	Number of Valid Bits in Last Octet	Number of Padded 1's to Last Octet
001	1	7
010	2	6
011	3	5
100	4	4
101	5	3
110	6	2
111	7	1
000	8	0

The BILO field is 000 in DICE update packets.

10.3.11 DICE information field

The number of octets in the information field shall be dependent on the bit rate of the incoming channel-oriented circuit. ANSI T1.312 specifies that the maximum size for the information field is 482 octets. To carry the traffic for certain national applications, the maximum number of octets in the DICE information field, NMAX, that constitute the information field, shall be set by service administration as a multiple of 7. This protects against the non-synchronization between the data bits and the corresponding control bit if the control bits are not removed and some packets are lost. The following values of NMAX (shown in table 5) correspond to a packetization period of about 20 ms, which is the packetization interval for non-voice packets such as facsimile and video.

Table 5 – Size of the DICE information field

Bit Rate bit/s	NMAX in Octets
2400	7
4800	14
9600	28
56000	133
64000	133

The octets of the DICE information field are transmitted in ascending numerical order. Inside an octet, bits are transmitted in an increasing order, i.e., bit 1 is transmitted first (see figure 3). Therefore, the bits shall be arranged in the DICE information field so that the first bit to be transmitted on the packet side is the first arriving bit on the channel-oriented side. As explained in 10.3.7, in some national applications, the first bit of an octet to arrive from the channel-oriented side is called the subrate framing bit, while the last bit is called the control bit. Arithmetically, bit 1 is the most significant bit of an octet from the channel-oriented side while bit 8 is the least significant bit of an octet on the packetized side.

10.4 DICE originating endpoint procedures

10.4.1 Preprocessing

The preprocessor shall operate on the bit stream arriving from the channel-oriented side. Subrate traffic consists of CMAX duplicated copies of the user traffic. CMAX is 1 for 56 kbit/s, 5 for 9600 bit/s, 10 for 4800 bit/s, and 20 for 2400 bit/s. The preprocessor shall retain only one copy of the subrate information and shall drop the redundant copies. It shall send this copy to the idle code detector.

10.4.2 Operation of the idle code detector

The idle code detector shall monitor the pre-processed traffic to determine if it contains idle codes or user data.

If an idle code is present, it shall discriminate between the non-failure idle codes (e.g., terminal idle or normal network idle) or failure codes. The patterns for the various idle codes are specified in table 1.

When the detector detects the arrival of an idle code, it shall proceed as follows:

a) for a normal idle code, it shall start two counters, IDLE_LAT_CNT and NORM_IDLE_CNT, to count the number of idle codes and the normal idle codes received, respectively;

b) for a failure idle code, it shall start the counters, IDLE_LAT_CNT and FAIL_IDLE_CNT, to count the number of idle codes and failure idle codes received, respectively.

Subsequently, if the same idle code continues to arrive on the channel-oriented side, the idle code detector shall increment the counter IDLE_LAT_CNT. Depending on the type of idle code, either NORM_IDLE_CNT or FAIL_IDLE_CNT shall be incremented.

If a data octet arrives on the channel-oriented side, the counters IDLE_LAT_CNT and either NORM_IDLE_CNT or FAIL_IDLE_CNT (depending on the case) shall be reset to 0.

If a new idle code arrives, the counters IDLE_LAT_CNT, NORM_IDLE_CNT, and FAIL_IDLE_CNT shall be reset to 0 and shall be updated to reflect the new idle code.

A protocol parameter, denoted as IDLE_LAT-_MAX, controls the update of the system variable IBT_IDLE. When the same idle code (whether normal or failure) arrives consecutively for IDLE-_LAT_MAX times, the idle code detector shall update the value of the system variable IBT-_IDLE according to table 1.

The value of IDLE_LAT_MAX can range from 2 to 15. It shall be selected to minimize the probability that data octets with bit errors are not interpreted erroneously as idle codes.

Even if NORM_IDLE_CNT and FAIL_IDLE_CNT have not reached their respective maximum, the idle latency counter IDLE_LAT_CNT may attain its maximum value IDLE_LAT_MAX. In this case, the idle code detector shall update the value of the IBT_IDLE system variable to correspond to the new idle code, as defined in table 1.

If the idle code detector receives the same pattern of a non-failure (normal) idle code for NORM_IDLE_MAX times, it shall stop the parallel-to-serial converter until the idle pattern changes or user data arrive. It shall reset the NORM_IDLE_CNT and the IDLE_LAT_CNT counters.

Similarly, if the idle code detector receives the same pattern of a failure idle code for FAIL_IDLE_MAX times, it shall stop the parallel-to-serial converter until the idle code pattern changes or user data arrive. It shall reset the FAIL_IDLE_CNT and the IDLE_LAT_CNT counters.

To allow the change of value of the IBT_IDLE system variable before the parallel-to-serial converter switches its state from active to one of the IDLE states, the following relation shall be satisfied:

$$\text{IDLE\_LAT\_MAX} \leq \text{NORM\_IDLE\_MAX} \ll \text{FAIL\_IDLE\_MAX}.$$

This relation is based on previous experience with some national circuits.

10.4.3 Operation of the parallel-to-serial converter

The parallel-to-serial converter shall convert the retained octet into a serial bit stream supplied to the data detector. The serialization shall be done in the order of increasing bit numbers, i.e., from bit 1 to bit 8, with the bit numbering done according to the nomenclature of the channel-oriented pattern of table 1. This is also the order of bit arrival on the channel-oriented side. As previously explained in 10.3.7, in some national networks the first bit of an arriving octet is called the subrate framing bit while the last bit is called the control bit.

For subrate traffic (bit rates less than 56 kbit/s), the parallel-to-serial converter shall remove the first bit of an octet (the "subrate framing" bit). If the circuit is provisioned for a multipoint configuration, it shall also remove the last bit of each octet (the "control" bit) for non-64-kbit/s channels. If the circuit is not provisioned for multipoint operation, the parallel-to-serial converter shall remove the "control" bit if this is specified by service provisioning. Whenever the control bit is removed, it shall set the variable C to 0; otherwise, it shall set it to 1.

Operation of the parallel-to-serial converter is under the control of the idle code detector. Thus the originating endpoint parallel-to-series converter has three global states: the ACTIVE state, the IDLE-NORMAL state, and the IDLE-FAILURE state. The transitions among the various states are depicted in figure 4.

NOTE – The actual time required to start the transition from the ACTIVE state to either the IDLE_NORMAL or IDLE_FAILURE state depends on the incoming subrate traffic.

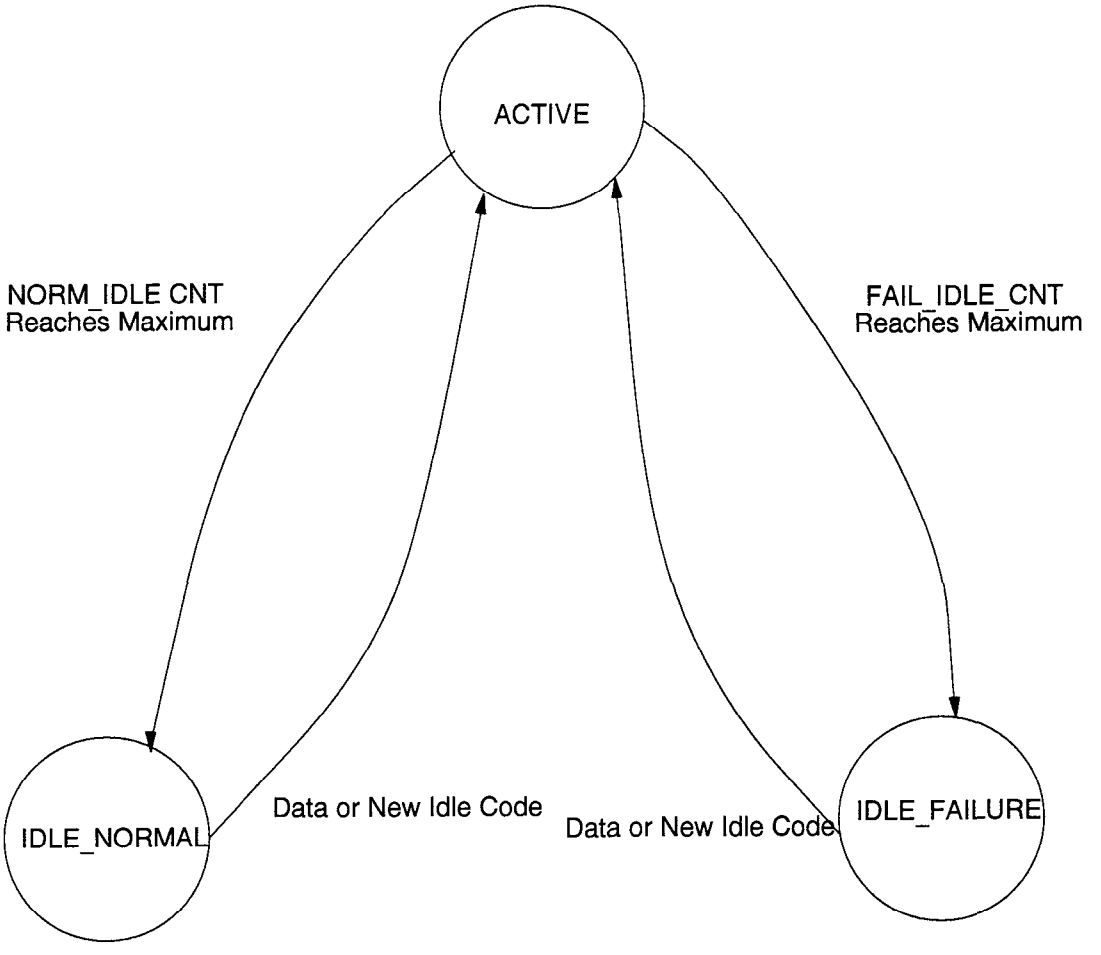


Figure 4 – Global DICE state transitions at the parallel-to-serial converter of the originating endpoint

The parallel-to-serial converter shall remain in the active state as long as the idle code detector has not signalled that the same pattern of idle code has arrived consecutively for a specified number of times. This number is a protocol parameter defined by service administration and is denoted as NORM_IDLE_MAX for non-failure (normal) codes and as FAIL_IDLE_MAX for failure codes.

When the parallel-to-serial converter changes its state from ACTIVE to one of the IDLE states, it shall send a message to the data collector so that the latter may cease collecting data. When the state changes from IDLE to ACTIVE, the parallel-to-serial converter informs the data collector to start collecting data.

Note that there is no direct transition between the idle failure and idle normal failure states, but that such a transition is mediated by the active state.

10.4.4 Operation of the data collector

The data detector shall peel the successive data bits from the incoming bit stream and shall arrange them in successive octets so that they can be put in the DICE information field in a way that preserves the order of transmission on each side. It shall take the bits in the order of their arrival and then stack them in the DICE information field, shown in figure 3, starting from bit 1, to bit 8 of each octet. When an octet is formed, it shall signal to the high layer entity that a DICE data octet is available.

If the parallel-to-serial converter signals to the data collector to stop collecting additional data, the data collector shall complete the current octet by putting 1's in the higher order bits. When all bits of the octet have been formed, it shall signal to the high layer entity that the last DICE octet is being sent and shall indicate to the high layer entity the number of valid bits in the last octet.

10.4.5 High layer procedures

The high layer can be in one of two states, the PACKETIZE state and the IDLE state.

10.4.5.1 Procedures in the IDLE state

In the IDLE state, the high layer entity of the originating endpoint shall cease packetization. If the circuit is provisioned to send idle update packets, the high layer entity shall restart the timer T_IDLE whenever it expires and then shall send the PL_DICE_IDLE_REQUEST(C,IBT) primitive to the packet layer to request the transmission of an idle update packet. T_IDLE determines the period between the successive transmission of idle update packets.

The idle update packet updates the idle code octet that the terminating endpoint at the far side should play on its channel-oriented side in the absence of information packets. If the data collector indicates that it has started to collect new data arriving on the channel-oriented side, the high layer entity shall stop the T_IDLE timer and shall move back to the PACKETIZE state.

10.4.5.2 Procedures in the PACKETIZE state

In this state, the high layer entity shall operate on an octet-by-octet basis to collect up to NMAX octets of information bits and shall construct the information field of a DICE frame. Thus, a DICE data spurt begins when the high layer entity enters the PACKETIZE state and ends when the high layer entity moves into the IDLE state.

The high layer entity shall update the value of the send sequence variable (SSEQ) and shall then inform the layer 3 of the originating endpoint to transmit a packet using the PL_DICE_DATA_REQUEST(SSEQ,BILO,C,IBT) primitive. The value of SSEQ is 0 for the first packet of each DICE data spurt. Subsequent SSEQ numbers are from 1 to 15 with a rollover to 1. The value of C is set to 1 for 64 kbit/s, or as indicated above for other rates. As long as the data collector has not informed that it is not sending the last octet, the BILO field shall always be set to 0. Finally, the value of the IBT field shall correspond to the current value of the IBT system variable, as defined in 10.4.2.

The high layer entity shall continue to packetize the incoming channel-oriented traffic until the data collector has informed it that it was sending the last octet of user data. The high layer entity shall then stop packetization and shall use the number of valid data bits in the last octet communicated by the data collector to select the value of the system variable BILO that represents the number of valid data bits as shown in table 4. It shall then request from the layer 3 entity to send the current packet in a DICE frame even if the size of its information field is less than NMAX octets using the primitive PL_DICE_DATA_REQUEST(SSEQ,BILO,C,IBT). The value of IBT in this primitive shall be equal to the current value of the system variable IBT_IDLE.

If the packet has a sequence number of 0, the packet layer shall invoke the build-out procedures of 6.3.3.2 of ANSI T1.312. At the scheduled play-out time, the packet layer shall send to the higher layer entity either the PL_DICE_DATA_INDICATION(BILO, C, IBT) primitive or the PL_DICE_IDLE_INDICATION(C,IBT) primitive, depending on whether the packet is a DICE information packet or a DICE idle update packet. In the case of an information packet, it shall set RSEQ to 1.

If the packet has a sequence number > 0 and arrives in sequence, the packet layer shall send to the higher layer entity the PL_DICE_DATA_INDICATION (BILO,C, IBT) primitive, and shall increment RSEQ.

If the packet arrives out-of-sequence with a non-zero sequence number, the packet shall be retained for the interval of (BUILDOUT – Time Stamp) ms.

At the scheduled play-out time, the packet layer shall inform the higher layer entity with the PL_DICE_DATA_INDICATION(BILO,C,IBT) primitive in the case of a DICE information packet and increment RSEQ, or with the PL_DICE_IDLE_INDICATION(C,IBT) primitive for an update packet and set RSEQ to 0.

10.6.3 High layer entity procedures

The high layer entity at the terminating end shall request from the packet layer to play out additional data whenever all data have been played out on the channel-oriented side.

The high layer entity will take each packet and then rearrange the DICE information field in a serial fashion. The serialization of the bits of each octet shall be performed in ascending order from bit 1 to bit 8, with the bit numbering done according to the numbering convention of figure 3. The high layer entity shall remove (8-BILO) 1's from the end of the data stream.

10.6.4 Procedures of the serial-to-parallel converter

The serial-to-parallel converter shall multiplex the serial bit stream arriving from the higher layer entity with specific bits it generates, and reorganize the multiplexed stream into a parallel format according to tables 1 and 3.

10.6.4.1 Procedures for bit rates below 56 kbit/s

The serial-to-parallel converter shall fill the remaining positions of the current octet, if any, with ones. It shall then retrieve the octet representing the idle code pattern that corresponds to the value stored in the system variable IBT_LAST, using table 1, and send it to the channel-oriented side.

For subrate operation, the serial-to-parallel converter shall reinsert the "subrate framing" bit (the first bit of an octet to be sent on the channel-oriented side), the "control" bit (the last bit to be sent) into the bit stream if C is 0 in each octet that it forms. The serial-to-parallel converter shall transmit CMAX consecutive copies of each octet.

10.6.4.2 Procedures at 56 kbit/s

The serial-to-parallel converter shall fill the remaining positions of the current octet, if any, with ones. It shall then retrieve the octet representing the idle code pattern that corresponds to the value stored in the system variable IBT_LAST, using table 1, and send it to the channel-oriented side.

For 56-kbit/s operation, the serial-to-parallel converter shall reinsert the "subrate framing" bit (the first bit of an octet to be sent on the channel-oriented side), the "control" bit (the last bit to be sent) into the bit stream if C is 0, and shall transmit one copy of each octet that it forms.

10.6.4.3 Procedures at 64 kbit/s

The serial-to-parallel code converter shall retrieve the octet representing the idle pattern that corresponds to the value stored in the system variable IBT_LAST, using table 1, and send it to the channel-oriented side.

For 64-kbit/s operation, the serial-to-parallel converter shall transmit one copy of each octet that it forms.

10.7 System variables

10.7.1 Send sequence state variable

See 8.1 in ANSI T1.312.

10.7.2 Receive sequence state variable

See 8.2 in ANSI T1.312.

10.7.3 IBT_IDLE

At the originating endpoint, IBT_IDLE stores the value that corresponds to the most recent idle code arriving on the channel-oriented side that has caused the IDLE_LAT_CNT to reach its threshold of IDLE_LAT_MAX.

10.7.4 IBT_LAST

At the terminating endpoint, IBT_LAST stores the value of the IBT in the last received packet.

10.7.5 T_IDLE

T_IDLE is the idle update timer. T_IDLE determines the period between the successive transmission of idle update packets. Its value is set to 60 seconds.

10.7.6 TVDELAY

This timer is used to measure the variable queueing delay that a packet experiences in a node that is used to update the TS field of a voiceband packet.

10.8 Protocol parameters

10.8.1 Build-out delay

See 9.1 in ANSI T1.312.

10.8.2 CMAX

CMAX is the number of duplicate copies of the subrate user traffic. Its value is 1 for 56 kbit/s, 5 for 9600 bit/s, 10 for 4800 bit/s, and 20 for 2400 bit/s.

10.8.3 FAIL_IDLE_MAX

FAIL_IDLE_MAX defines the maximum number of consecutive identical failure idle octets that can arrive at the channel-oriented side before the high layer entity enters the idle failure state and stops packetization. For point-to-point connections, the number is one of the values 2, 3, 6, 12, 15, 30, 60, and 500. The default value is 500. For multipoint connections, the value is set to 2 and cannot be changed.

10.8.4 IDLE_LAT_MAX

IDLE_LAT_MAX defines the maximum number of consecutive identical idle octets that can arrive on the channel-oriented side before updating the value of the IBT_IDLE system variable. This prevents the erroneous updating of IBT_IDLE due to bit errors. The value of IDLE_LAT_MAX ranges from 2 to 15. The default value is 2.

10.8.5 NMAX

NMAX defines the maximum number of octets in the DICE information field of a DICE packet (see 10.3.11).

10.8.6 NORM_IDLE_MAX

NORM_IDLE_MAX defines the maximum number of consecutive identical normal (non-failure) idle octets that can arrive on the channel-oriented side before the high layer entity enters the idle normal state. For point-to-point connections, the allowable values are 2, 3, 6, 12, 15, 30, 60, and 500. The default value is 3 to allow for the transmission of two consecutive idle codes on the secondary channel. The value of 6 corresponds to 4.8 kbit/s lines provisioned at 9.6 kbit/s. The value of 12 corresponds to 2.4-kbit/s lines provisioned at 9.6 kbit/s. The values of 15, 30, and 60 correspond respectively to 9.6-kbit/s, 4.8-kbit/s, and 2.4-kbit/s lines provisioned at 56 kbit/s. For multipoint connections, the value is set to 2 and cannot be changed.

10.9 Summary of primitives

10.9.1 Primitives for the interfaces between layers 2 and 3

The primitives for the interfaces between layers 2 and 3 have the same definition as in clause 10 of ANSI T1.312.

10.9.1.1 DL_L1_READY_INDICATION

See 10.1.1 of ANSI T1.312.

10.9.1.2 DL_UNIT_DATA_REQUEST

See 10.1.2 of ANSI T1.312.

10.9.1.3 DL_UNIT_DATA_INDICATION

See 10.1.3 of ANSI T1.312.

10.9.1.4 DL_UNIT_H_DATA_REQUEST

See 10.1.4 of ANSI T1.312.

10.9.1.5 DL_UNIT_H_DATA_INDICATION

See 10.1.5 of ANSI T1.312.

10.9.1.6 DL_PVP_H_DATA_INDICATION

See 10.1.6 of ANSI T1.312.

10.9.1.7 DL_PVP_DATA_INDICATION

See 10.1.7 of ANSI T1.312.

10.9.1.8 DL_PVP_H_DATA_REQUEST

See 10.1.8 of ANSI T1.312.

10.9.1.9 DL_PVP_DATA_REQUEST

See 10.1.9 of ANSI T1.312.

10.9.2 Primitives for the interface between layer 3 and the higher layer entity

10.9.2.1 PL_DICE_DATA_INDICATION(BILO,C, IBT)

PL_DICE_DATA_INDICATION(BILO,C,IBT) primitive is used by the packet layer of the terminating endpoint to indicate to its high layer entity the arrival of a DICE information packet with the parameter BILO, C and IBT.

10.9.2.2 PL_DICE_DATA_REQUEST The PL_DICE_DATA_REQUEST primitive is used by the high layer entity of the terminating endpoint to request the data from the layer 3 entity.

10.9.2.3 PL_DICE_DATA_REQUEST(SSEQ,BILO, C,IBT)

The PL_DICE_DATA_REQUEST(SSEQ,BILO,C, IBT) primitive is used by the originating endpoint high layer entity to request the layer 3 entity to transmit DICE information packets. SSEQ is 0 for the first packet of a DICE data spurt. Subsequent primitives have the SSEQ numbers 1 to 15 with a rollover to 1. The value of BILO is set to 0 for 64 kbit/s and to its appropriate value for the other rates. The value of C is set to 1 for 64 kbit/s and to 0 for other bit rates. Finally, the value of the IBT corresponds to the current idle code, as defined in 10.4.2.

10.9.2.4 PL_DICE_IDLE_REQUEST (C,IBT)

PL_DICE_IDLE_REQUEST(C,IBT) primitive is used by the originating endpoint high layer entity to request the layer 3 entity to transmit an idle update packet. The value of C is set to 1 for 64 kbit/s, and to 0 for other rates. The value of the IBT corresponds to the current idle code, as defined in 10.4.2.

10.9.2.5 PL_DICE_IDLE_INDICATION (C,IBT)

PL_DICE_IDLE_REQUEST(C,IBT) primitive is used by the packet layer of the terminating endpoint to indicate to its high layer entity the arrival of a DICE idle update packet.

11 Virtual data link capability (VDLC)

The Virtual Data Link Capability (VDLC) protocol may be used to transport digital data that arrive on the channel-oriented side. VDLC builds upon the same physical, link, and packet layers of ANSI T1.312 and extends DICE to cover the bit-oriented procedures of HDLC. Thus, VDLC removes specific bits, idle codes, and HDLC flags from the channel-oriented traffic.

VDLC allows frame partitioning and rejoining, and the inversion of the polarity of the bit stream. In place of lost information packets, VDLC allows a provisionable number of HDLC flags to be played out on the channel-oriented side.

The reception, packetization, and transport of DS0-B multiplexed subrates, 56 kbit/s with error correction on two 64-kbit/s time slots, and 19200 bit/s are for further study.

Subclauses 11.1, 11.2, and 11.3 describe the physical, link, and packet layers by referring to the description of ANSI T1.312. Subclause 11.4 describes VDLC procedures for the high, packet, and link layers of the originating endpoint. Subclause 11.5 describes the procedures at the intermediate nodes, while 11.6 describes the procedures at the terminating endpoint.

11.1 Physical layer

The physical layer is the same for ANSI T1.312 (see 4.1 of ANSI T1.312).

11.2 Link layer

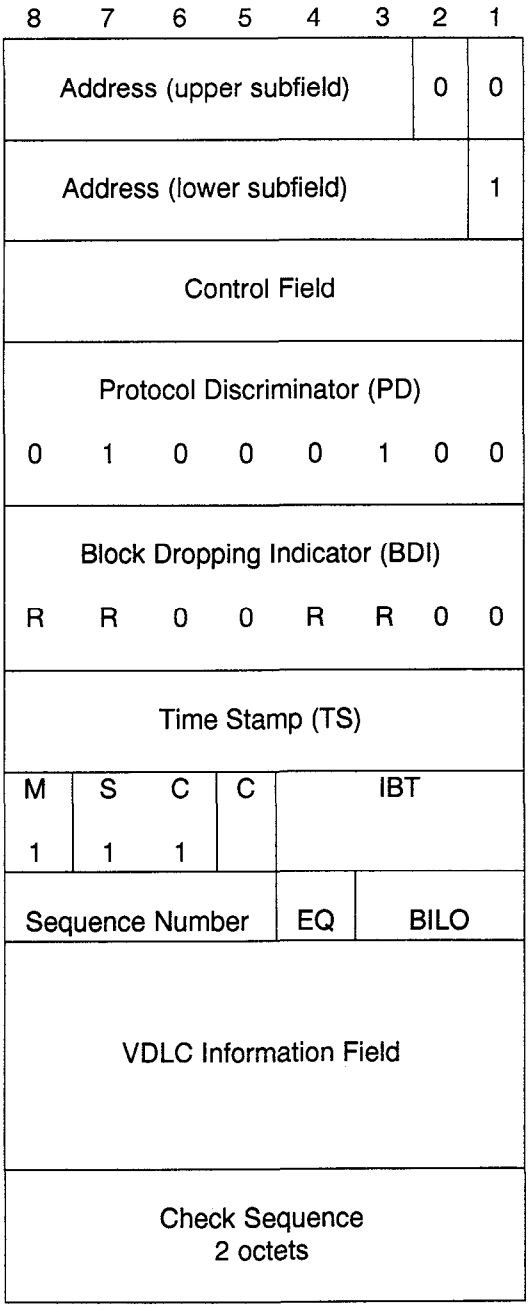
The link layer is the same for DICE (see 10.2).

11.3 Packet layer

The packet layer procedures apply to the information transfer phase only. Call control procedures are outside the scope of this standard.

VDLC packets are of the format shown in figure 5. There are three types of VDLC packets:

- VDLC information packets;
- VDLC update packets;
- VDLC flag packets.



M = M-bit
R = Reserved for future use and set to 0

Figure 5 – VDLC frame format

VDLC update packets have zero-length information field, a SEQ value of 0, and an EQ value of 0. They update the bit pattern that the terminating endpoint should play to the channel-oriented side in the absence of information packets. VDLC flag packets contain HDLC flags in their VDLC information field.

11.3.1 Protocol discriminator (PD)

Same as for DICE (see 10.3.1).

11.3.2 Block dropping indicator (BDI)

Same as for DICE (see 10.3.2).

11.3.3 Time stamp (TS)

Same as for DICE (see 10.3.3).

11.3.4 M-bit

Same as for DICE (see 10.3.4).

11.3.5 Sub-class (SC) field

Same as for DICE (see 10.3.5).

11.3.6 Control (C) bit

The control (C) bit is set by the originating endpoint to indicate whether the last bit to arrive for each octet has been removed. In some national applications, this bit is called the control bit and is used for synchronization, status, and remote testing on the channel-oriented side. C is set to 0 if the control bit is removed; otherwise C is set to 1. For non-64-kbit/s channels, C shall be set to 0, because the control bit is always removed.

11.3.7 Idle background type (IBT)

As in DICE, the IBT indicates the type of idle code that has been removed from the original data stream by the originating endpoint. In addition, VDLC defines three new entries that are used to indicate to the terminating endpoint what action to take in case of HDLC frames. The type of idle codes to be recognized is determined by service administration.

In addition to the IBT codes described in table 1, VDLC uses the IBT coding types of table 6 at the originating endpoint for point-to-point and multipoint connections.

The initial value of the IBT field is 1111. This value shall be updated according to the traffic characteristics, as indicated in 11.4.5.1.

At the terminating endpoint, the IBT coding types shown in table 7 are used for both point-to-point and multipoint connections, in addition to those already listed in table 3.

Table 6 – Meaning of the additional coding types at the originating endpoint

Bit rate (kbit/s)	IBT code	Meaning
All	0000	Complete HDLC frame in VDLC frame
	0000	Partitioned HDLC frame (last)
	0001	Partitioned HDLC frame (not last)
	0010	VDLC frame contains HDLC flags (flag packet)
	0010	VDLC frame contains an update packet in the FLAG_IDLE state

Table 7 – Meaning of the additional coding types at the terminating endpoint

Bit rate (kbit/s)	IBT code	Meaning
All	0000	Stuff bits for transparency and append an HDLC flag
	0001	Stuff bits for transparency and no HDLC flag appended
	0010	No bit stuffing (VDLC flag packet and VDLC update packet)

11.3.8 Sequence number

Same as for DICE (see 10.3.8).

11.3.9 Delay equalization bit

The EQ bit is set by the originating endpoint to enable or to disable the build-out delay procedures at the terminating endpoint. When EQ is set to 1, build-out procedures are enabled; otherwise, they are disabled. The originating endpoint shall set EQ to 0 except for the following cases, where EQ is set to 1:

– the first VDLC packet that contains a partial HDLC frame of user information when frame partitioning and rejoining is done;

– subsequent packets that contain partial HDLC frame information, with the exception of the last packet.

11.3.10 Bits in last octet

The BILO field contains the number of bits in the last information octet that are valid data bits.

11.3.11 VDLC information field

The minimum length of a frame arriving on the channel-oriented side is 4 octets between HDLC flags. Smaller frames shall be dropped at the originating endpoint.

The size for the VDLC information field, VMAX, shall not exceed 482 octets, as required by ANSI T1.312. However, a smaller size can be selected by service administration, according to table 5.

If the size of an HDLC frame on the channel-oriented side exceeds VMAX octets, the frame shall either be dropped or shall be broken into consecutive VDLC frames, depending on operator's provisioning.

The octets of the VDLC information field are transmitted in ascending numerical order. Inside an octet, bits are transmitted in an increasing order, i.e., bit 1 is transmitted first (see figure 5). Therefore, the bits shall be arranged in the VDLC information field so that the first bit to be transmitted on the packet side is the first arriving bit on the channel-oriented side. As explained in 10.3.7, in some national applications, the first bit of an octet to arrive from the channel-oriented side is called the subrate framing bit, while the last bit is called the control bit. Arithmetically, bit 1 is the most significant bit of an octet on the channel-oriented side while bit 8 is the least significant bit of an octet on the packetized side.

11.4 VDLC procedures at the originating endpoint

The channel-oriented bit stream is first treated according to 11.4.1. Two high layer entity processors monitor the pre-processed bit stream. The idle code detector, whose action is described in 11.4.2, operates on the pre-processed traffic. The parallel-to-serial conversion of the traffic is the function of the parallel-to-serial converter, which is described in 11.4.3. The HDLC detector operates on the serial bit stream, as shown in 11.4.4.

11.4.1 Preprocessing

The subrate traffic consists of CMAX duplicated copies of the user traffic. CMAX is 1 for 56 kbit/s, 5 for 9600 bit/s, 10 for 4800 bit/s, and 20 for 2400 bit/s, and the subrate framing pattern consists of CMAX bits. For traffic at rates less than 56 kbit/s, the preprocessor shall retain one copy of the subrate information and remove the redundant octets.

11.4.2 Operation of the idle code detector

The idle code detector operates on the pre-processed traffic to distinguish idle codes from other octets. Idle codes are defined by service administration, as indicated in table 1.

When the idle code detector detects an idle code specified by service administration, it shall start the counter IDLE_LAT_CNT to count the number of idle codes received. If the same idle code continues to arrive on the channel-oriented side, the counter IDLE_LAT_CNT shall be incremented for each newly arrived octet.

If the idle detector detects a non-idle octet, it shall reset the counter IDLE_LAT_CNT.

If the idle code detector detects a new idle, it shall set the counter IDLE_LAT_CNT to 1 and shall increment it as the octets of the new idle code continue to arrive.

If the IDLE_LAT_CNT counter reaches the value of IDLE_LAT_MAX, it shall send a message to the high layer entity to signal that the threshold IDLE_LAT_MAX has been reached and to inform it of the corresponding idle code.

If the IDLE_LAT_CNT counter reaches the value of VDLC_IDLE_MAX, it shall send a message to the high layer entity to signal that the threshold VDLC_IDLE_MAX has been reached and to inform it of the corresponding idle code.

The values of IDLE_LAT_MAX and VDLC_IDLE_MAX shall be set by service provisioning.

11.4.3 Operation of the parallel-to-serial converter

The parallel-to-serial converter shall convert the retained octet into a serial bit stream supplied to the HDLC detector. The serialization shall be done in the order of increasing bit numbers, i.e., from bit 1 to bit 8, with the bit numbering done according to the nomenclature of the channel-oriented pattern of table 1. This is also the order of bit arrival on the channel-oriented side. As previously explained in 10.3.7, in some national applications, the first bit of an arriving octet is

called the subrate framing bit while the last bit is called the control bit.

The parallel-to-serial converter shall remove the first bit of an octet (the "subrate framing" bit) for subrate traffic at rates of less than 56 kbit/s. It shall also remove the last bit of an octet (the "control" bit) for non-64-kbit/s channels. If so provisioned by service administration, it shall invert the remaining bits.

11.4.4 Operation of the HDLC detector

The HDLC detector operates on the serial bit stream to determine if the traffic contains an HDLC flag, an abort (at least seven consecutive 1's), or user data.

HDLC octets are not necessarily octet-aligned with the octets on the channel-oriented side. For example, in some applications, the HDLC detector may require the examination of up to three octets of channel-oriented traffic to recognize the presence of an HDLC flag.

The HDLC detector shall peel the successive data bits from the incoming bit stream and shall arrange them in successive octets so that they can be put in the VDLC information field in a way that preserves the order of transmission on each side. It shall take the bits in the order of their arrival and shall stack them in the VDLC information field (shown in figure 5), starting from bit 1 to bit 8 of each octet. It shall remove the bits stuffed for transparency from the data stream. When an octet is formed, it shall signal to the high layer entity that an HDLC data octet is available. Also, it shall signal to the high layer entity to drop the data bits collected between two flags, if these data bits are not octet aligned.

11.4.5 High layer procedures

The high layer entity can be in one of six global states: The AWAIT state, the PACKETIZE state, the IDLE state, the FLAG_IDLE state, the FLAG_WAIT state, and the ABORT state. The initial state of the high layer entity is the IDLE State. A simplified state transition diagram of the high layer entity is shown in figure 6. The system variables, timers and counters used in the following description are:

– IBT_IDLE, which corresponds to the most recent idle code that has caused the idle latency counter IDLE_LAT_CNT of the originating endpoint to reach one of the two thresholds, IDLE LAT MAX or VDLC_IDLE_MAX. The initial value of IBT_IDLE is 1111 (15 decimal);

– IBT_LAST, which stores the value of the IBT in the last received packet at the terminating endpoint. The initial value of IBT_LAST is 1111 (15 decimal);

– IDLE_LAT_CNT, which is the idle latency counter for idle codes before updating IBT_IDLE;

– NFLAG, which is the flag latency counter for HDLC flags. NFLAG counts the number of times that HDLC flags have been received consecutively;

– RSEQ, the receive sequence number variable, gives the SEQ of the next packet to be received. It is updated by incrementing the previous RSEQ from 1 to 15 with a rollover to 1;

– SSEQ, the send sequence number variable, gives the SEQ of the next packet to be sent. It is updated by incrementing the previous SSEQ from 1 to 15 with a rollover to 1;

– T_IDLE, which is a timer that determines the period between the successive transmission of update packets;

– VDLC_IDLE_CNT, which is the idle code latency counter for the number of idle codes before the high layer entity enters the IDLE state.

11.4.5.1 Procedures in the IDLE state

Upon entering this state, the high layer entity of the originating endpoint shall cease packetization of the incoming channel-oriented traffic. It shall restart the timer T_IDLE whenever it expires and then shall send the PL_VDLC_IDLE_REQUEST(IBT) primitive to the packet layer to transmit a VDLC update packet. IBT has the value stored in the system variable IBT_IDLE. (The layer 3 entity shall set the SEQ to 0 and the EQ to 0 in the packet header, as explained in 11.4.6.)

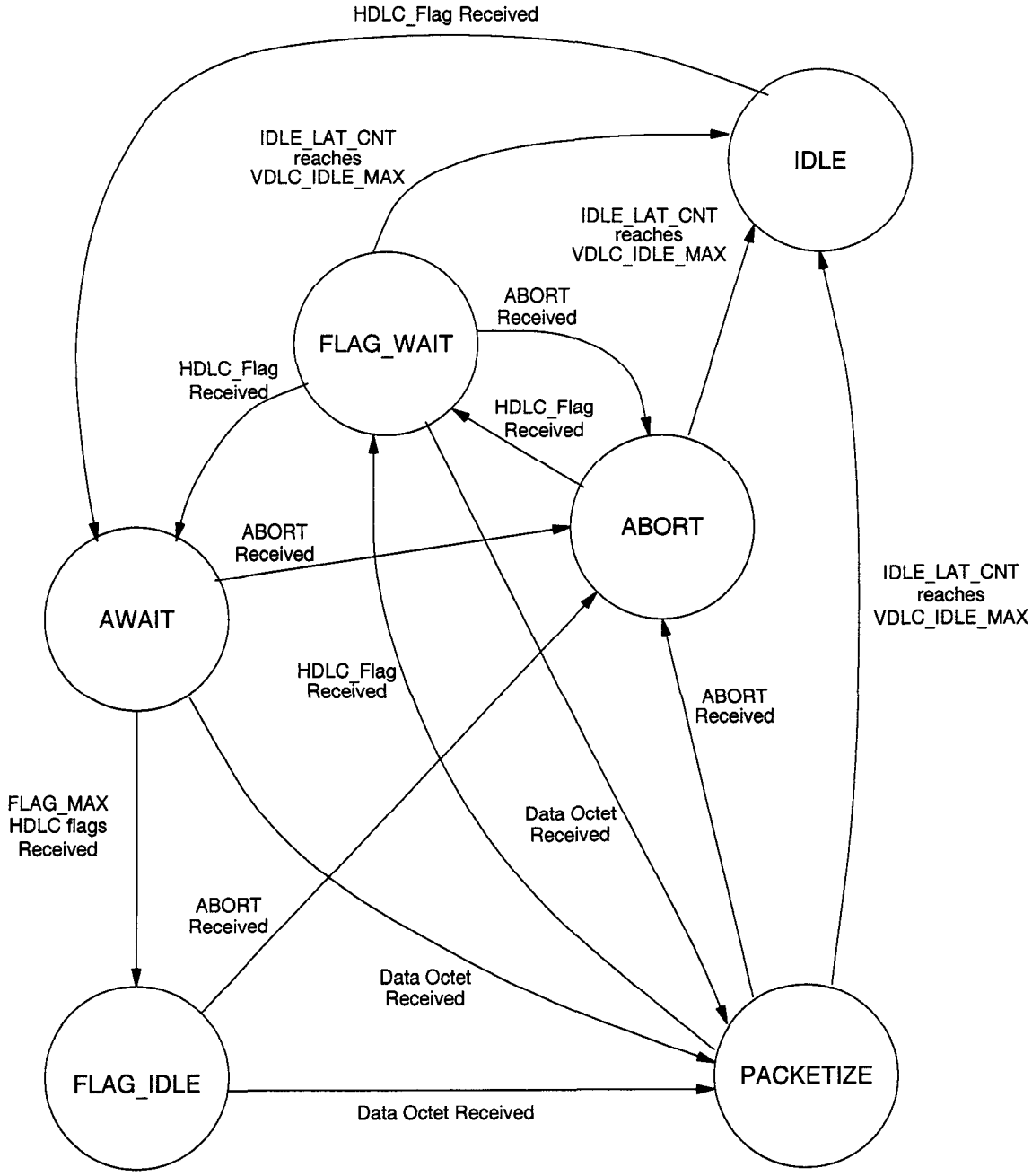


Figure 6 – Global VDLC state transitions at the high layer entity of the originating endpoint

If a message arrives from the idle detector indicating that the IDLE_LAT_CNT counter has reached its threshold IDLE_LAT_MAX, the high layer entity shall update the value of IBT_IDLE to correspond to the idle code in question. It shall restart the timer T_IDLE and shall send the PL_VDLC_IDLE_REQUEST(IBT) primitive to the packet layer to transmit a VDLC update packet with the IBT equal to the new value of the IBT_IDLE system variable.

If the HDLC detector indicates the reception of an HDLC flag, the high layer shall stop the timer T_IDLE. It shall reset the flag counter NFLAG to 0 and the SSEQ to 0 and shall move to the AWAIT state.

11.4.5.2 Procedures during the AWAIT state

In the AWAIT state, the high layer entity shall increment NFLAG whenever the HDLC detector indicates that an HDLC flag has arrived on the channel-oriented side.

If the value of NFLAG becomes FLAG_MAX, the high layer entity shall put NFLAG HDLC flags in the VDLC information field. It shall then request the transmission of a VDLC packet using the primitive PL_VDLC_FLAG_REQUEST(SSEQ). SSEQ is incremented within the range 1 to 15 with a rollover to 1. The packet layer shall set IBT to 0010 and EQ to 0. It shall leave the AWAIT state and enter the FLAG_IDLE state. FLAG_MAX is selected by service administration.

If the HDLC detector indicates the reception of an abort, the high layer entity shall put NFLAG HDLC flags in the VDLC information field. It shall request the transmission of a VDLC flag packet using the primitive PL_VDLC_FLAG_REQUEST(SSEQ). The high layer entity shall then move to the ABORT state.

If the HDLC detector indicates the reception of HDLC data, the high layer entity shall put NFLAG HDLC flags in the VDLC information field. It shall then request the transmission of a VDLC flag packet using the primitive PL_VDLC_FLAG_REQUEST(SSEQ). The high layer entity shall put the data octet in the first octet of the information field and then move to the PACKETIZE state.

11.4.5.3 Procedures during the PACKETIZE state

The high layer entity shall remain in the PACKETIZE state as long as (a) the idle code detector has not indicated that the counter IDLE_LAT_CNT has reached VDLC_IDLE_MAX, and (b) the HDLC detector has not indicated that it has received an HDLC flag or an abort.

In the PACKETIZE state, the high layer entity shall operate on an octet-by-octet basis to collect up to VMAX octets of information bits and shall construct the information field of a VDLC frame.

The high layer entity shall continue preparing the VDLC information field until either the HDLC detector has detected a new HDLC flag or an abort, or the idle code detector has indicated that the IDLE_LAT_CNT has reached its threshold VDLC_IDLE_MAX.

If the HDLC detector indicates reception of a new flag, the subsequent action is as follows:

a) If the VDLC information field contains fewer than four octets, the VDLC packet shall be dropped and the high layer entity shall move to the FLAG_WAIT state;

b) If the HDLC detector signals that the VDLC information field is not octet aligned, the high layer entity shall drop the packet and shall move to the FLAG_WAIT state;

c) If the VDLC information field is ≤ VMAX, the high layer entity shall send the primitive PL_VDLC_DATA_REQUEST(SSEQ,IBT,EQ) with IBT = 0000 and EQ = 0 to the packet layer. It shall increment the SSEQ within the range 1 to 15 with a rollover to 1 and shall go to the FLAG_WAIT state;

d) If the VDLC information field is larger than VMAX octets, then depending upon the provisioning, the incoming HDLC frames shall either be partioned into successive VDLC frames or shall be dropped. SSEQ shall not be updated if the frame is dropped. In both cases, the high layer entity shall then move to the FLAG_WAIT state.

If an HDLC frame is partitioned, the high layer entity shall set the SSEQ to 0 and shall send the primitive PL_VDLC_DATA_REQUEST(SSEQ, IBT,EQ) with SSEQ = 0, IBT = 0001, and EQ = 1 for first frame. For subsequent VDLC frames, it shall set IBT to 0001, and EQ to 1. The high

layer entity shall increment the SSEQ with a roll-over to 1 after sending each primitive. The value of EQ = 1 ensures that the terminating endpoint builds-out the VDLC frame before play-out. The value of IBT = 0001 ensures that the terminating endpoint does not append flags. The last packet shall have IBT = 0000 and EQ = 0 to add flags after play-out and avoid the build-out procedures and improve the overall end-to-end delay.

If a message arrives from the idle detector to indicate that the value of VDLC_IDLE_MAX has been reached, the high layer entity shall update the value of the system variable IBT_IDLE to correspond to the idle code in question. It shall send the PL_VDLC_IDLE_REQUEST(IBT) primitive to the packet layer to request the transmission of an update packet with IBT equal to the new value of system variable IBT_IDLE. (The layer 3 entity shall set the value of EQ to 0 and SEQ to 0.) It shall start the timer T_IDLE and then move to the IDLE State.

If the HDLC detector indicates that an abort has arrived, the high layer entity shall drop the current packet without updating SSEQ, and shall move to the ABORT state.

11.4.5.4 Procedures in the FLAG_IDLE state

The high layer entity shall restart the timer T_IDLE whenever it expires and then shall send the PL_VDLC_IDLE_REQUEST(IBT) primitive to the packet layer to transmit a VDLC update packet with IBT = 0010. The packet layer shall set EQ to 0 and SEQ to 0 in the packet header.

The high layer shall stop the timer T_IDLE if the HDLC detector indicates reception of a data octet. The high layer entity shall set the SSEQ to zero; it shall put the data octet in the first octet of the information field and then shall move to the PACKETIZE state.

If the HDLC detector indicates reception of an abort, the high layer entity shall stop the timer T_IDLE and shall move to the ABORT state.

11.4.6 Procedures in the ABORT state

In the ABORT state, the high layer entity shall cease packetization.

When the idle code detector indicates that the counter IDLE_LAT_CNT has reached the threshold VDLC_IDLE_MAX, the high layer entity shall update the IBT_IDLE system variable to correspond to the idle code in question. It shall send the PL_VDLC_IDLE_REQUEST(IBT) primitive to the packet layer to request the transmission of an update packet, with IBT equal to the new value of system variable IBT_IDLE. (The layer 3 entity shall set EQ to 0 and SEQ to 0.) It shall start the timer T_IDLE and then move to the IDLE State.

If the HDLC detector indicates reception of an HDLC flag, the high layer entity shall move to the FLAG_WAIT state.

11.4.7 Procedures in the FLAG_WAIT state

In this state, the high layer entity shall cease packetization.

When the idle code detector indicates that the counter IDLE_LAT_CNT has reached the threshold VDLC_IDLE_MAX, the high layer entity shall update the IBT_IDLE system variable to correspond to the idle code in question. It shall send the PL_VDLC_IDLE_REQUEST(IBT) primitive to the packet layer to request the transmission of an update packet, with IBT equal to the new value of system variable IBT_IDLE. (The layer 3 entity shall set EQ to 0 and SEQ to 0.) It shall start the timer T_IDLE and then move to the IDLE State.

The high layer entity shall set the flag counter NFLAG to 1 and shall move to the AWAIT state if the HDLC detector indicates reception of an HDLC flag.

If the HDLC detector indicates reception of an abort, the high layer entity shall put the data octet in the first octet of the information field and then shall move to the ABORT state. SSEQ shall not be updated.

If the HDLC detector indicates reception of an octet of data, the high layer entity shall put the data octet in the first octet of the information field and then move to the PACKETIZE State.

11.4.8 Layer 3 procedures for the originating endpoint

When the high layer entity informs the layer 3 entity to transmit a packet using the PL_VDLC_DATA_REQUEST(SSEQ,IBT,EQ) the layer 3 entity shall start the timer TVDELAY and insert the values IBT and EQ in their respective fields. The SEQ shall be set equal to the value of the SSEQ.

If the high layer entity indicates to the layer 3 entity to send update packets using the PL_VDLC_IDLE_REQUEST(IBT) primitive, the layer 3 entity shall start the timer TVDELAY and set the IBT field to the value indicated in the primitive. It shall set the SEQ to 0, the BILO to 0000 and the EQ to 0.

If the high layer entity indicates to the layer 3 entity to send flag packets using the PL_VDLC_FLAG_REQUEST(SSEQ) primitive, the layer 3 entity shall start the timer TVDELAY and set the SEQ field to the value indicated by SSEQ in the primitive. It shall set the EQ to 0, the BILO to 0000, and the IBT to 0010.

The layer 3 entity shall await the arrival of the DL_L1_READY_INDICATION primitive from layer 2. Upon receipt of this primitive, layer 3 shall stop the timer TVDELAY and its value shall be copied to the TS field. The value of the TS shall not exceed 200 ms. If the variable delay exceeds 200 ms, the value is set to 200 ms.

The layer 3 entity shall pass the VDLC packet to the layer 2 entity for transport using the DL_UNIT_H_DATA_REQUEST primitive (if the control field is UIH) or the DL_UNIT_DATA_REQUEST (if the control field is UI).

11.4.8.1 VDLC link layer procedures

The link layer procedures are the same as those defined in 5.2.1 and 5.2.2 of ANSI T1.312.

11.5 Intermediate node procedures

Same as for DICE (see 10.5).

11.6 VDLC terminating endpoint procedures

11.6.1 VDLC link layer procedures

The link layer procedures are the same as specified in 5.2.3 and 5.2.4 of ANSI T1.312.

11.6.2 Layer 3 procedures

Upon receipt of the DL_UNIT_H_DATA_INDICATION primitive from layer 2 for a UIH frame (or DL_UNIT_DATA_INDICATION for a UI frame), the layer 3 entity at the terminating endpoint shall examine the value encoded in the PD field. If this value matches that for PVP, the layer 3 entity shall proceed as below. Otherwise, it shall drop the packet. The layer 3 entity shall examine the value coded in the SC field. If this value is decimal 3, the layer 3 entity shall proceed as below. Otherwise, it shall drop the packet.

When the high layer entity requests the play-out of data using the PL_VDLC_DATA_REQUEST primitive, the packet layer shall check its queue.

If the queue is empty, the packet layer shall send PL_VDLC_IDLE_INDICATION(IBT) to the high layer entity.

If a packet is on the queue, the packet layer shall proceed according to the value of the EQ field. If EQ is = 0, no build-out procedures shall be invoked; if EQ is = 1, the build-out procedures of 6.3.3.2 of ANSI T1.312 shall be invoked. In all cases, when RSEQ is incremented, it shall be incremented in the range of 1 to 15 with a roll-over to 1. Also, the value of the system variable IBT_LAST shall be updated to contain the value of the IBT field in the received packet if and only if this IBT value corresponds to an idle code defined in table 1.

11.6.2.1 Procedures without build-out

When a packet arrives in sequence with a non-zero sequence number, the packet layer shall play it immediately. The packet layer shall inform the high layer entity with the PL_VDLC_DATA_INDICATION(IBT) primitive and increment the value of RSEQ.

If the packet arrives in sequence and has a sequence number of 0, the packet layer shall inform the high layer entity with the PL_VDLC_DATA_INDICATION(IBT) primitive, and set RSEQ to 1 if the packet is a VDLC information packet, or with a PL_VDLC_IDLE_INDICATION(IBT) for an update packet.

If the packet arrives out-of-sequence and has a sequence number > 0, the packet layer shall examine the value of the IBT field. If the value of the IBT field is either 0 or 1, the packet layer shall send the PL_VDLC_FLAGS_INDICATION primitive to the higher layer entity. Otherwise, it shall send the PL_VDLC_ONE_FLAG_INDICATION primitive.

The packet layer shall inform the high layer entity with the PL_VDLC_DATA_INDICATION(IBT) primitive and increment RSEQ if the packet is a VDLC information packet, or with a PL_VDLC_IDLE_INDICATION(IBT) and set RSEQ to 0 for an update packet.

11.6.2.2 Procedures with build-out

If the packet has a sequence number of 0, the packet layer shall invoke the build-out procedures of 6.3.3.2 of ANSI T1.312. At the scheduled playout-time, the packet layer shall send to the higher layer entity either the PL_VDLC_DATA_INDICATION primitive or the PL_VDLC_IDLE_INDICATION(IBT) primitive, depending on whether the packet is a VDLC information packet or a VDLC update packet. In the case of an information packet, it shall set RSEQ to 1.

If the packet has a sequence number > 0 and arrives in sequence, the packet layer shall send to the higher layer entity the PL_VDLC_DATA-_INDICATION primitive, and shall increment RSEQ.

If the packet arrives out-of-sequence with a non-zero sequence number, the packet layer shall examine the value of the IBT field. If the value of the IBT field is either 0 or 1, the packet layer shall send the PL_VDLC_FLAGS_INDICATION primitive to the higher layer entity. Otherwise, it shall send the PL_VDLC_ONE_FLAG_INDICATION primitive. The packet layer shall then schedule the packet for playout according to the build-out procedures of 6.3.3.2 of ANSI T1.312. Thus the packet shall be retained for the interval of (BUILDOUT – Time Stamp) ms.

At the scheduled play-out time, the packet layer shall inform the higher layer entity with the PL_VDLC_DATA_INDICATION(IBT) primitive in the case of a VDLC information packet and increment RSEQ, or with the PL_VDLC_IDLE-_INDICATION(IBT) primitive for an update packet and set RSEQ to 0.

11.6.3 High layer entity procedures

The high layer entity at the terminating endpoint shall request from the packet layer to play out additional data whenever all data have been played out on the channel-oriented side.

The high layer entity will take each packet and then rearrange the VDLC information field in a serial fashion. The serialization of the bits of each octet shall be performed in ascending order from bit 1 to bit 8, with the bit numbering done according to the numbering convention of figure 5.

The high layer entity shall also inform the serial-to-parallel converter whether to operate in the IDLE mode or in the HDLC mode.

11.6.3.1 Procedures after receiving PL_VDLC_ONE_FLAG_INDICATION

The high layer entity shall inform the serial-to-parallel converter to operate in the HDLC mode. It shall then send the bit pattern representing one HDLC flag to the serial-to-parallel converter.

11.6.3.2 Procedures after receiving PL_VDLC_FLAGS_INDICATION

The high layer entity shall inform the serial-to-parallel converter to operate in the HDLC mode. It shall then send the bit pattern representing FLAG_MIN HDLC flags to the serial-to-parallel converter.

11.6.3.3 Procedures after receiving PL_VDLC_IDLE_INDICATION(IBT)

The action depends on whether the IBT code corresponds to the values in tables 1 and 3, or in tables 6 and 7.

If IBT > 2, the high layer entity shall put the serial-to-parallel converter in the IDLE mode.

If IBT ≤ 2, the high layer entity shall put the serial-to-parallel converter into the HDLC mode and shall send to it the bit pattern representing one HDLC flag in a serial fashion.

11.6.3.4 Procedures after receiving PL_VDLC_DATA_INDICATION(IBT)

The high layer entity shall put the serial-to-parallel converter into the HDLC mode. Subsequent actions depend on whether the IBT code corresponds to the values in tables 1 and 3, or in tables 6 and 7.

a) If IBT = 0, the high layer entity shall send the VDLC information field serially to the serial-to-parallel converter. The serialization of each octet shall be done in the order of increasing bit number, as shown in figure 5. For data transparency, it shall insert a 0 after each sequence of five consecutive ones in the data stream. After pumping out the data bits, the high layer entity shall send the bit pattern of an HDLC flag serially. It shall then request additional data from the packet layer, using the PL_VDLC_DATA_INDICATION.

b) If IBT = 1, the high layer entity shall send the VDLC information field serially to the serial-to-parallel converter. The serialization of each octet shall be done in the order of increasing bit number, as shown in figure 5. For data transparency, it shall insert a 0 after each sequence of five consecutive ones in the data stream. After pumping out the data bits, the high layer entity shall request additional data from the packet layer, using the PL_VDLC_DATA_INDICATION.

c) If IBT = 2, the high layer entity shall send the VDLC information field serially to the serial-to-parallel converter. The serialization of each octet shall be done in the order of increasing bit number, as shown in figure 5. After pumping out the data bits, the high layer entity shall request additional data from the packet layer, using the PL_VDLC_DATA-_INDICATION.

If the IBT value is different than 0, 1 or 2 the high layer entity shall indicate an error to the management entity.

11.6.4 Procedures of the serial-to-parallel converter

The serial-to-parallel converter shall multiplex the serial bit stream arriving from the higher layer entity with specific bits it generates, and reorganize the multiplexed stream into a parallel format according to tables 1 and 3.

The serial-to-parallel converter has two modes of operation: the HDLC mode of operation and the IDLE mode of operation. The high layer entity determines the mode of operation of the serial-to-parallel converter, as described in 11.6.3. The procedures used in each mode depends on the speed, as described below.

11.6.4.1 Procedures for bit rates below 56 kbit/s

a) In the HDLC mode, the serial-to-parallel converter shall put the incoming serial bits in the bit positions 2 through 7 of each octet that it shall form. In each octet, it shall set the values of bits 1 and 8 to 1. Bit 1 is the first bit of an octet to be transmitted on the channel-oriented side, and bit 8 is the last bit to be transmitted. Arithmetically, bit 1 is the most significant bit of an octet on the channel-oriented side, while bit 8 is the least significant bit. As explained in 10.3.7, in some national applications, the first bit of an octet on the channel-oriented side is called the subrate framing bit, while the last bit is called the control bit. The nomenclature of the number of the bits of the channel-oriented pattern is as shown in tables 1 and 3.

The serial-to-parallel converter shall transmit CMAX consecutive copies of each octet that it forms.

b) In the IDLE mode of operation, the serial-to-parallel converter shall fill the remaining position of the current octet, if any, with ones. It shall then retrieve the octet representing the idle pattern that corresponds to the value stored in the system variable IBT_LAST, using table 1, and send it to the channel-oriented side.

The serial-to-parallel converter shall transmit CMAX consecutive copies of each octet that it forms.

11.6.4.2 Procedures at 56 kbit/s

a) In the HDLC mode, the serial-to-parallel converter shall put the incoming serial bits in the bit positions 1 through 7 of each octet that it shall form. In each octet, it shall set the values of bit 8 to 1. Bit 8 is the first bit of an octet to be transmitted on the channel-oriented side. Arithmetically, bit 8 is the least significant bit of an octet on the channel-oriented side. As explained in 10.3.7, in some national applications, the last bit of an octet on the channel-oriented side is called the control bit. The nomenclature of the number of the bits of the channel-oriented pattern is as shown in tables 1 and 3.

b) In the IDLE mode of operation, the serial-to-parallel converter shall fill the remaining position of the current octet, if any, with ones. It shall then retrieve the octet representing the idle pattern that corresponds to the value stored in the system variable IBT_LAST, using table 1, and send it to the channel-oriented side.

For 56-kbit/s operation, the serial-to-parallel converter shall transmit one copy of each octet that it forms.

11.6.4.3 Procedures at 64 kbit/s

a) In the HDLC mode, the serial-to-parallel converter shall put the incoming serial bits in the bit positions 1 though 8 of each octet that it shall form. Bit 1 is the first bit of an octet to be transmitted on the channel-oriented side, and bit 8 is the last bit to be transmitted. Arithmetically, bit 1 is the most significant bit of an octet on the channel-oriented side, while bit 8 is the least significant bit. The nomenclature of the number of the bits of the channel-oriented pattern is as shown in tables 1 and 3.

b) In the IDLE mode of operation, the serial-to-parallel converter shall fill the remaining position of the current octet, if any, with ones. It shall then retrieve the octet representing the idle pattern that corresponds to the value stored in the system variable IBT_LAST, using table 1, and send it to the channel-oriented side.

For 64-kbit/s operation, the serial-to-parallel converter shall transmit one copy of each octet that it forms.

11.7 System variables

11.7.1 IBT_IDLE

At the originating endpoint, the IBT_IDLE system variable corresponds to the most recent idle code that has caused the idle latency counter IDLE_LAT_CNT of the originating endpoint to reach one of the two thresholds, IDLE_LAT_MAX or VDLC_IDLE_MAX. The initial value of IBT_IDLE is 1111 (15 decimal).

11.7.2 IBT_LAST

At the terminating endpoint, IBT_LAST stores the value of the IBT in the last received packet. The initial value of IBT_LAST is 1111 (15 decimal).

11.7.3 Send sequence state variable

See 8.1 in ANSI T1.312.

11.7.4 Receive sequence state variable

See 8.2 in ANSI T1.312.

11.7.5 T_IDLE

Same as 10.7.5 for DICE.

11.7.6 TVDELAY

Same as 10.7.6 for DICE.

11.8 Protocol parameters

11.8.1 Build-out delay

See 9.1 in ANSI T1.312.

11.8.2 CMAX

CMAX is the number of duplicate copies of the subrate user traffic. Its value is 1 for 56 kbit/s, 5 for 9600 bit/s, 10 for 4800 bit/s, and 20 for 2400 bit/s.

11.8.3 FLAG_MAX

FLAG_MAX is the maximum size of a VDLC flag packet.

11.8.4 FLAG_MIN

FLAG_MIN is the number of HDLC flags that must be played on the channel-oriented side instead of a lost packet. The value of FLAG_MIN may be set to 1, 2, 16, or 32. The default value is 1. It is recommended that FLAG_MIN = FLAG_MAX.

11.8.5 IDLE_LAT_MAX

IDLE_LAT_MAX is the maximum number of times the same idle octet must be received consecutively on the channel-oriented side before the IBT_IDLE system variable can be changed. Its default value is 2.

11.8.6 VDLC_IDLE_MAX

VDLC_IDLE_MAX is the maximum number of times that the same idle octet must be received consecutively on the channel-oriented side before the high layer entity moves to the IDLE state. The value of VDLC_IDLE_MAX is set to 8.

11.8.7 VMAX

VMAX defines the maximum number of octets that shall be collected from the channel-oriented side to form a VDLC frame. The value of VMAX is determined by service administration, and shall be less than or equal to 482 octets.

11.9 Summary of primitives

11.9.1 Primitives for the interfaces between layers 2 and 3

The primitives for the interfaces between layers 2 and 3 have the same definition as in clause 10 of ANSI T1.312 and 10.9.1 for DICE.

11.9.1.1 DL_L1_READY_INDICATION

See 10.1.1 of ANSI T1.312.

11.9.1.2 DL_UNIT_DATA_REQUEST

See 10.1.2 of ANSI T1.312.

11.9.1.3 DL_UNIT_DATA_INDICATION

See 10.1.3 of ANSI T1.312.

11.9.1.4 DL_UNIT_H_DATA_REQUEST

See 10.1.4 of ANSI T1.312.

11.9.1.5 DL_UNIT_H_DATA_INDICATION

See 10.1.5 of ANSI T1.312.

11.9.1.6 DL_PVP_H_DATA_INDICATION

See 10.1.6 of ANSI T1.312.

11.9.1.7 DL_PVP_DATA_INDICATION

See 10.1.7 of ANSI T1.312.

11.9.1.8 DL_PVP_H_DATA_REQUEST

See 10.1.8 of ANSI T1.312.

11.9.1.9 DL_PVP_DATA_REQUEST

See 10.1.9 of ANSI T1.312.

11.9.2 Primitives for the interface between layer 3 and the higher level entity

11.9.2.1 PL_VDLC_DATA_INDICATION

The PL_VDLC_DATA_INDICATION primitive is used by the packet layer of the terminating endpoint to indicate to the high layer entity the arrival of VDLC information packet.

11.9.2.2 PL_VDLC_DATA_REQUEST

The PL_VDLC_DATA_REQUEST primitive is used by the high layer entity of the terminating endpoint to request data from the layer 3 entity.

11.9.2.3 PL_VDLC_DATA_REQUEST (SSEQ,IBT,EQ)

The PL_VDLC_DATA_REQUEST(SSEQ,IBT,EQ) primitive is used by the originating endpoint high layer entity to indicate to the layer 3 entity to transmit VDLC information packets with the corresponding values of IBT and EQ and with the sequence number SEQ = SSEQ. The value of C is set to 0 in the packet.

11.9.2.4 PL_VDLC_FLAGS_INDICATION

The PL_VDLC_FLAGS_INDICATION primitive indicates to the high layer entity of the terminating endpoint to play FLAG_MIN HDLC flags to replace a lost packet.

11.9.2.5 PL_VDLC_FLAG_REQUEST(SSEQ)

The PL_VDLC_FLAG_REQUEST(SSEQ) primitive is used by the originating endpoint high layer entity to indicate to the layer 3 entity to transmit VDLC flag packets with the sequence number SEQ = SSEQ. The layer 3 entity sets IBT to 0010 and EQ to 0 and C to 0.

11.9.2.6 PL_VDLC_IDLE_INDICATION(IBT)

The PL_VDLC_IDLE_INDICATION(IBT) primitive is used by the packet layer of the terminating endpoint to indicate to the high layer entity the arrival of VDLC update packet with the IBT value indicated in the primitive.

11.9.2.7 PL_VDLC_IDLE_REQUEST(IBT)

The PL_VDLC_IDLE_REQUEST(IBT) primitive is used to update the idle code that the far end should play at the channel-oriented side. The value of the IBT corresponds to the value of the IBT_IDLE system variable.

11.9.2.8 PL_VDLC_ONE_FLAG_INDICATION

The PL_VDLC_ONE_FLAG_INDICATION primitive indicates to the high layer entity of the terminating endpoint to play one HDLC flag to replace a lost packet.

12 Interface with LAPD

If the digital data protocol is native LAPD, then the LAPD frame relay option can be used. For primary rate applications that require code restrictions to maintain one's density, bit inversion is necessary so that bit stuffing and bit inversion prevent the all 0 octet and satisfy the one's density requirements of restricted transmission facilities.

13 Interface with V.120

CCITT (or ITU-T) Recommendation V.120 may be used to rate adapt and submultiplex lower speed data into higher speed data. Two situations may apply:

– the digital data on a channel-oriented circuit may be formatted using a synchronous HDLC-like protocol;

– the digital data may be asynchronous.

13.1 V.120 Frame format

The CCITT Recommendation V.120 specifies the packetization of data into LAPD-like frames. It includes three modes of operation:

– an asynchronous mode;

– a synchronous mode, which envelopes HDLC frames and achieves data compression by removing redundant flags that would normally be transmitted on fixed sub-rate circuits to fill the time between frames;

– a bit-transparent mode.

In frame relay networks, V.120 restricts the modes to the asynchronous and the synchronous modes.

The V.120 frame format is similar to the layer 2 frame format specified in ANSI T1 312 It consists of an HDLC flag, address octets (default is 2), control octets (HDLC format), optional V 120 header octets to communicate terminal status and controls, information octets, FCS octets, and a HDLC flag.

In the asynchronous mode, no segmentation capability is provided for; therefore, octet 3 of the V.120 frame will have the form:

X	X	X	X	X	X	1	1

X = 1 or 0

In this case, therefore, this pattern will be different from the PD of PVP.

In the synchronous mode, a segmentation capability is provided and the segmentation bits (bits 1 and 2) may be either 0 or 1. Bit 1 is set to 1 to indicate that the frame contains the final portion of the message. Bit 2 is set to 1 to indicate that the frame is the first one in a series of messages Both are set for a message corresponding to a single frame.

Bits 3 and 4 are used for error indication. Both are set to 1 when the data terminal equipment overruns the V.120 terminal adaptor. Therefore, octet 3 takes the following form:

X	X	X	X	1	1	X	X

X = 1 or 0

Bit 3 is set to 1 when the message is aborted. Also, bit 1 is set to 1 to indicate the end of segmentation. Accordingly, octet 3 becomes:

X	X	X	X	0	1	0	1

X = 1 or 0

In either case, the V.120 header does not coincide with the PVP protocol discriminator.

13.1.1 V.120 Asynchronous mode operation

The PCME shall implement a V.120 synchronous mode operation, as defined in CCITT recommendation V.120 3.3.1, for use in asynchronous applications.

13.2 V.120 Synchronous mode operation

The PCME shall implement V.120 synchronous mode operation, as defined in Recommendation V.120 3.3 2, for use when administration indicates that the data is formatted using HDLC protocols, in inverted polarity. The logical addresses for V.120 shall be established by service administration, and the V.120 signaling frames and procedures shall not be used.

13.3 V.42 and V.42 Bis

The use of V.42 and V.42 bis for error control and data compression is for further study.

14 Signaling transport

14.1 General

The following subclauses address the transport of signaling information arriving at the channel-oriented side of a PCME node on 1544-kbit/s and 2048-kbit/s primary multiplex rate systems. The two types of signaling information addressed are·

– channel-associated signaling;

– common channel signaling

Refer to clause 3 of CCITT Recommendation G 704 regarding the signaling methods for the 1544-kbit/s interface and to clause 5 of CCITT Recommendation G.704 regarding the signaling methods for 2048-kbit/s interfaces.

Signaling information arriving in designated signaling channels shall be packetized and transported over a packet connection separate from that of packetized speech or voiceband data in accord with clause 5 of ANSI T1.312. Signaling tones in voice channels that arrive as PCM

sampled tones shall be packetized and transported over a packet connection as packetized voiceband signals. (See 14.2.1.)

14.2 Channel-associated signaling

The PCME shall support the transport of channel-associated signaling. By service administration, a voice circuit shall be identifiable as having 2-state (A), 4-state (AB), or 16-state (ABCD) inband signaling, or none of these. When signals are transported as PCM sampled analogue signaling tones, they should be transported in voiceband signal packets. Originating endpoint PCME nodes shall transport signaling information arriving on the channel-oriented side according to either the 1544-kbit/s or the 2048-kbit/s primary multiplex rate.

14.2.1 PCM sampled analog signaling tones

In systems in which signaling information is transferred as a 64-kbit/s PCM sampled analogue tone, the originating endpoint PCME node shall interpret the PCM signal arriving on its channel-oriented side as a voiceband signal, which it shall packetize using UIH voice frames, as defined in 4.3.1 of ANSI T1.312. It shall then transmit those frames to the terminating endpoint PCME node, which shall transfer the data from the voice frames to its channel-oriented side, in accordance with ANSI T1.312.

14.2.2 2-, 4-, and 16-state channel-associated signaling

Signaling information in the form of 2-, 4-, or 16-state channel-associated signaling on the channel-oriented side of a PCME node shall be transported on the packetized side using UI signaling frames, as specified in 4.3.2 of ANSI T1.312. The signaling packets shall be carried via a logical address different from the logical address for speech or voiceband data information. The system parameters required to control the transport of this signaling information are TSIG_REF and TSIG_KA, which are defined in clause 9 of ANSI T1.312.

When an originating or terminating endpoint node is provisioned, signaling bits on the channel-oriented side unused by a signaling system should be mapped to specific values, as noted in CCITT Recommendation G.704.

14.2.2.1 1544-kbit/s signaling

For the 1544-kbit/s primary multiplex rate described in CCITT Recommendation G.733, either a 12-frame or 24-frame multiframe format is present on the channel-oriented side. The allocation of signaling bits for the channel-oriented side is described in 3.1 of CCITT Recommendation G.704.

For 2-state signaling, the A bit shall be mapped directly to the A bit field of the T1.312 signaling frame. The B, C, and D bits of the T1.312 signaling packet shall be mapped so that B = A, C = A, and D = B. Signaling frames on the packet side of the originating endpoint shall be generated only when transitions of the A bit occur on the channel-oriented side of the originating endpoint, or when TSIG_REF expires. The terminating endpoint node shall pass the A bit only from the packet side to the channel-oriented side interface. The B, C, and D bits shall be ignored. They shall not be used for testing or maintenance purposes.

For 4-state signaling, the PCME originating endpoint shall accept the A and B bits, and shall map them into the A and B bit fields in the UI signaling frame, as specified in ANSI T1.312. The unused C and D bits on the channel-oriented side shall be ignored. The C and D bit fields in the signaling frame should be set with C = A and D = B. Transition of either the A or B bit on the channel-oriented side, or the expiration of TSIG_REF, shall cause a signaling frame to be generated on the packet side of the originating endpoint. The terminating endpoint node shall pass the A and B bits only to the channel-oriented side interface.

For 16-state signaling, the originating endpoint node shall map the channel-oriented side ABCD bits directly to the ABCD bit field of the signaling frame. Transition of any one or more of the ABCD bits on the channel-oriented side of the originating endpoint, or expiration of TSIG_REF, shall cause a T1.312 signaling frame to be generated. The terminating endpoint node shall pass the ABCD bit field from the signaling frame to the terminating endpoint channel-oriented side directly.

14.2.2.2 2048-kbit/s signaling

The 2048-kbit/s primary multiplex rate, described in CCITT Recommendation G.733, supports channel-associated signaling as specified in CCITT Recommendation G.704. The

arrangement of the 64-kbit/s channel time slot 16 provides signaling channels designated *a, b, c,* and *d*.

When 2-state signaling is used, only transitions of the *a* bit on the channel-oriented side of the originating endpoint, or expiration of TSIG_REF, shall cause a ANSI T1.312 signaling frame to be generated and *a* to be mapped to the A bit field on the packet side. As per 5.1.3.2.2 of CCITT Recommendation G.704, one option is to set the unused *b, c,* and *d* signaling bits on the channel-oriented side to the values *b* = 1, *c* = 0, and *d* = 1. Therefore, the B, C, and D bits of the ANSI T1.312 signaling packet should be set to B = 1, C = 0, and D = 1. The user may specify other values as well, such as (a) each bit is not changed, (b) each bit is set to 0, (c) each bit is set to 1, (d) each bit is inverted to ensure end-to-end signaling compatibility. The terminating endpoint shall transfer the *a, b, c,* and *d* bits from the packet side to the channel-oriented side. It is the user's responsibility to ensure end-to-end signaling compatibility.

When 4-state signaling is used, transitions of either the *a* or *b* bit on the channel-oriented side of the originating endpoint, or expiration of TSIG_REF, shall generate ANSI T1.312 signaling packets with A = *a* and B = *b*. As per CCITT Recommendation G.704, one option is to map the C and D bit fields in the signaling frame to C = 0 and D = 1. The user may specify other values as above. The terminating endpoint shall transfer the *a, b, c,* and *d* bits from the packet side to the channel-oriented side. The C and D bits shall be ignored. They shall not be used for testing or maintenance purposes.

For 16-state signaling, the originating endpoint node shall accept from the channel-oriented side and map to the signaling frame the *a* bit to A field, *b* to B, *c* to C, and *d* bit to the D field. When the full 16 states are provisioned, a change in state of the *a, b, c,* or *d* bit on the channel-oriented side, or the expiration of TSIG_REF, shall generate an ANSI T1.312 signaling frame on the packet side. The terminating endpoint shall transfer the *a, b, c,* and *d* bits from the packet side to the channel-oriented side.

14.2.3 Interface between 2-, 4-, and 16-state signaling

When different signaling formats (2-, 4-, or 16-state signaling) in the same primary rate multiplex are provisioned on the channel-oriented sides of a PCME network, translation from one signaling format to the other shall be accomplished on the channel-oriented side of either endpoint so that the PCME network interfaces the same signaling format at both endpoints. The associations shall be set as indicated in 14.2.2.1 and 14.2.2.2. It is the user's responsibility to ensure end-to-end signaling compatibility.

14.2.4 Interface between 1544-kbit/s and 2048-kbit/s primary rate signaling systems

In the case where 1544-kbit/s and 2048-kbit/s primary rate channel-associated signaling systems are connected via a PCME network, translation of signaling from one format to the other, including bit inversion, shall be accomplished at the channel-oriented side of the 2048-kbit/s interface. The associations shall be set as indicated in 14.2.2.1 and 14.2.2.2. It is the user's responsibility to ensure end-to-end signaling compatibility.

14.2.5 Trunk conditioning for channel-associated signaling

In the presence of facility and maintenance alarms, the PCME node shall be able to recognize alarms affecting the channel-oriented and packet side of the originating and terminating endpoints. In national networks, the user may provision the trunk conditioning action to be taken for the affected channels on the channel-oriented side.

The procedures taken by a PCME node when a RED, YELLOW, AIS, or out of frame (OOF) alarm is established or cleared on the channel-oriented or packet side are described in clause 7 of ANSI T1.312.

14.2.6 Support of present channel-associated signaling systems

14.2.6.1 Signaling system R1

The PCME network shall support the R1 signaling system defined in CCITT Recommendation Q.310. Signaling information includes line signaling for line or supervisory signals and register signaling for address signals, all of which are addressed in the following sections.

14.2.6.1.1 2600 Hz line signaling

When analogue circuits are cascaded with PCM systems, a 2600 Hz line signaling tone, as described in CCITT Recommendation Q.311, is

coded and transferred in the PCM system as a 64-kbit/s PCM sampled tone. This shall be carried in the PCME network as voiceband signals in the manner specified in 14.2.1.

14.2.6.1.2 PCM line signaling

In digital PCM systems, R1 signaling provides for individual channel PCM line signaling in the 1544-kbit/s primary multiplex rate (CCITT Recommendation Q.314). The PCM line signaling is provided in a 12-frame multiframe using 4-state signaling in which the same signaling information is sent on both the A and B signaling channels. The PCME network shall carry this signaling information in the manner specified for 4-state signaling in 14.2.2.1.

14.2.6.1.3 Register signaling

Register signaling uses pulses of two-out-of-six multifrequency inband tones, as specified in CCITT Recommendation Q.315. Appearing on PCM systems as PCM sampled tones, these pulses shall be transported over the PCME network in the manner specified in 14.2.1.

14.2.6.2 Signaling system R2

Signaling system R2, defined in CCITT Recommendation Q.400, includes analogue and digital versions of line signaling, and interregister signaling for address signals. The PCME network shall support the transport of system R2 signaling information, as specified in the following subclauses.

14.2.6.2.1 Line signaling – analogue version

The analogue version of line signaling (CCITT Recommendations Q.411, Q.412, Q.414) uses a 3825 Hz out-of-band tone for signaling link-by-link. The 3825 Hz tone shall be processed into a digital signal before this interface and shall arrive on the channel-oriented side of a PCME node as a 64-kbit/s bit stream. This tone shall be transported in the manner specified in 14.2.1.

14.2.6.2.2 Line signaling – digital version

The digital version of line signaling for system R2 is transmitted link-by-link using 4-state signaling (two signaling channels) in each direction, as described in CCITT Recommendation Q.421. The PCME network shall transport this signaling information in the manner specified for 4-state signaling in 14.2.2.2.

14.2.6.2.3 Interregister signaling

System R2 interregister signaling is performed end-to-end (or by end-to-end sections) using multifrequency two-out-of-six inband tones in a compelled signaling procedure. When a relatively long period may elapse between reception of the last digit and detection of the condition of the called subscriber's line, such as when a satellite link is included, fully compelled signaling may be suspended by using pulsed backward signals (CCITT Recommendation Q.442). Appearing as PCM sampled tones on the channel-oriented side of a PCME node, these continuous or pulsed tones shall be transported in the manner of 14.2.1.

14.2.6.3 Signaling system No. 5

Line signaling for Signaling System No. 5 (SS5) is accomplished link-by-link using 2400 Hz and 2600 Hz tones transmitted individually or in combination, as specified in CCITT Recommendations Q.140 and Q.141. Register signaling for SS5, specified in CCITT Recommendation Q.151, is accomplished link-by-link using *en bloc* multifrequency pulsed combinations of two-out-of-six inband tones.

The PCME node shall support the transport of analogue SS5 signal tones as 64-kbit/s PCM sampled tones, as specified in 14.2.1.

14.3 Common channel signaling

14.3.1 Common channel signaling (out-of-band physical)

Prevalent types of common channel signaling systems include Signaling System No. 6 (SS6) and Signaling System No. 7 (SS7).

14.3.1.1 Common channel signaling for 1544-kbit/s primary multiplex rate

As provided in CCITT Recommendation G.704, for both 24-frame and 12-frame multiframes, one octet time slot is used to provide common channel signaling at a rate of 64 kbit/s. In the case of a 12-frame multiframe, the S-bits (the first bit of even numbered frames) may be arranged to carry common channel signaling at a rate of 4 kbit/s or a submultiple of this rate. These channels are capable of transporting SS6 information. SS7 is optimized for 56-kbit/s or 64-kbit/s digital channels, but is also suitable for operation at lower speeds.

14.3.1.2 Common channel signaling for 2048-kbit/s primary multiplex rate

As provided in CCITT Recommendation G.704, channel time slot 16 may be used for common channel signaling up to a rate of 64 kbit/s to transport both SS6 and SS7. The method of obtaining signal alignment will form part of the particular common channel signaling specification.

14.3.1.3 Interface for signaling between 1544-kbit/s and 2048-kbit/s primary rates

A PCME network transporting common channel signaling information between 1544-kbit/s and 2048-kbit/s primary multiplex rate systems shall carry that information transparently on the packet stream. Details are for further study.

14.3.1.4 Support of present common channel signaling systems

14.3.1.4.1 Signaling system No. 6 (SS6)

The transport of SS6 is left for further study.

14.3.1.4.2 Signaling system No. 7 (SS7)

SS7 is optimized for operation on 56- or 64-kbit/s channels. It is suitable for use on point-to-point links and provides error detection and correction for each individual signaling link.

A network of PCMEs shall support transport of the SS7 signal units that are carried on the signaling links comprising a SS7 network.

Signal units arrive on the channel-oriented side of the PCME endpoint in the basic formats given in clause 2 of CCITT Recommendation Q.703. Each signal unit uses opening and closing HDLC flags consisting of single octets with the bit pattern 01111110. The originating endpoint of the SS7 signaling link uses HDLC bit stuffing procedures to preclude imitation of the flag code.

SS7 signal units arriving at the channel-oriented side of an originating endpoint shall be transported on the packet side using the VDLC frame format and procedures given in clause 10.

The maximum length of a SS7 signaling unit is 280 octets in North America (clause 4 of ANSI T1.111) and 70 octets for international networks (clause 4 of CCITT Recommendation Q.703). These lengths are well within the maximum length requirement of an ANSI T1.312 information field. If it is desired that SS7 signal units be transported entirely within single VDLC frames, the maximum length VMAX of the VDLC frame information field shall be provisioned large enough to accommodate the appropriate maximum SS7 signal unit lengths. In this case the EQ bit (bit 4 of octet 8) and the M-bit (bit 8 of octet 7) shall be set to 0.

For the purpose of congestion control, SS7 traffic is categorized as administrative traffic. VDLC frames transporting such traffic over the PCME network shall be so marked using the logical address assignment. See 14.6 for congestion control procedures.

14.3.2 Common channel signaling (out-of-band logical)

A prevalent type of common-channel signaling (out-of-band logical) is D-channel signaling. D-channel signaling can be efficiently transported using LAPD. Other types of signaling are left for further study.

14.4 International reserved signaling bits

The use of international reserved signaling bits (S_i bits in Table 1a/G.704) is to be defined.

14.5 National reserved signaling bits

The use of the spare bits reserved for national use (S_n bits in Table 1a/G.704) is to be defined.

14.6 Procedures

The procedures to convey per-call requests and dynamic load control are to be defined.

15 Facsimile demodulation and compression protocol

The Facsimile Demodulation and Compression Protocol (FADCOMP) describes procedures to compress Group III (G3) facsimile traffic by PCME. Group III facsimile traffic includes two types of information transfer: (a) call control information, and (b) image data. The call control information flows in both directions while the transfer of image data is unidirectional. The image data include the T.30 training check (TCF).

FADCOMP procedures specify that:

- end-to-end capability indication procedures shall follow facsimile call set-up;
- the handshake and call control information shall flow on a logical link provisioned for

voice, following the procedures described ANSI T1.312;

– if the indication procedures show that both PCME endpoints are compatible, the originating (or demodulating) endpoint shall extract the baseband image signal and transmit it at its baseband rate. The terminating (or remodulating) endpoint shall regenerate the sampled voiceband signal of the image data;

– if the indication procedures show that both ends are not compatible, the image traffic shall continue to proceed on the voice path.

The facsimile demodulation and compression *protocol is a T.30 protocol analysis approach.* It relies on a continuous analysis of the T.30 protocol exchanges from both directions to determine the call parameters, the demodulating, and the remodulating sides, and to follow the progress of the facsimile call. As such, this protocol specification only supports facsimile equipment complying with Group III (G3) specified in T.30.

Figure 7 shows the various protocol layers and peer-to-peer entities involved with FADCOMP. The protocol uses the same physical and link layers as ANSI T1 312, and builds upon the packet layer of ANSI T1.312 using two new layers: the Modulation Layer and the Call-State Layer

While this specification provides detailed procedures for V.29 and V 27 ter facsimile calls, it does not preclude other modulation methods (e.g., V.17). Detailed procedures for these other methods are left for further study.

The organization of this specification is as follows: 15.1, 15.2, and 15.3 describe the formats of the physical, link and packet layers, in part by referring to the description found in ANSI T1.312, and 15.4 describes the Call-State Layer, whose operation is the same for both endpoints Subclause 15.5 describes the protocol procedures at the originating (demodulating) endpoint Subclause 15.6 describes the procedures at the intermediate nodes, while 15.7 describes the procedures at the terminating (remodulating) endpoint. Subclause 15.8 summarizes the system variables and protocol parameters. Subclause 15.9 lists the interface primitives used in this protocol.

15.1 Physical layer

The Physical Layer is the same as for ANSI T1.312 (see 4.1 of ANSI T1.312).

15.2 Link layer

The Link Layer is the same as for ANSI T1.312. In particular, the address field is the same as described in 4.2.1 of ANSI T1.312.

There are three types of facsimile frames: (a) facsimile capability indication frames, (b) facsimile spurt header frames, and (c) facsimile page information frames. All frames flow on the same virtual circuit, which is the same virtual circuit used for the voiceband path that carries the ANSI T1.312 frames of the permanent virtual connection. Figure 8 depicts the format of a facsimile capability indication frame, figure 9 depicts the format of a facsimile spurt header frame, and figure 10 describes the format of a facsimile page information frame The facsimile capability indication and spurt header frames may be either a UI frame or a UIH frame. The facsimile page information is a UIH frame. The control field of the UI frame is described in CCITT Recommendation Q.921/I.441. The control field of the UIH frame is described in 4.2.3.2 of ANSI T1.312.

15.3 Packet layer

The following fields are common to all packet types.

15.3.1 Protocol discriminator

Same as for ANSI T1.312 (see 4.3 2 1 of ANSI T1 312).

15.3.2 Block dropping indicator

Same as for DICE (see 10.3.2 of this standard).

15.3.3 Time stamp

Same as for DICE (see 10.3.3 of this standard).

15.3.4 M-bit

The M-bit is set to 0, except for page information packets. In this case, the M-bit is set to 1 for all page information packets, except for the last packet of a page spurt, where it is set to 0.

15.3.5 Sub-class field

The Sub-Class (SC) field is used to indicate that the packet is a digital modem packet. The SC field is coded as 01.

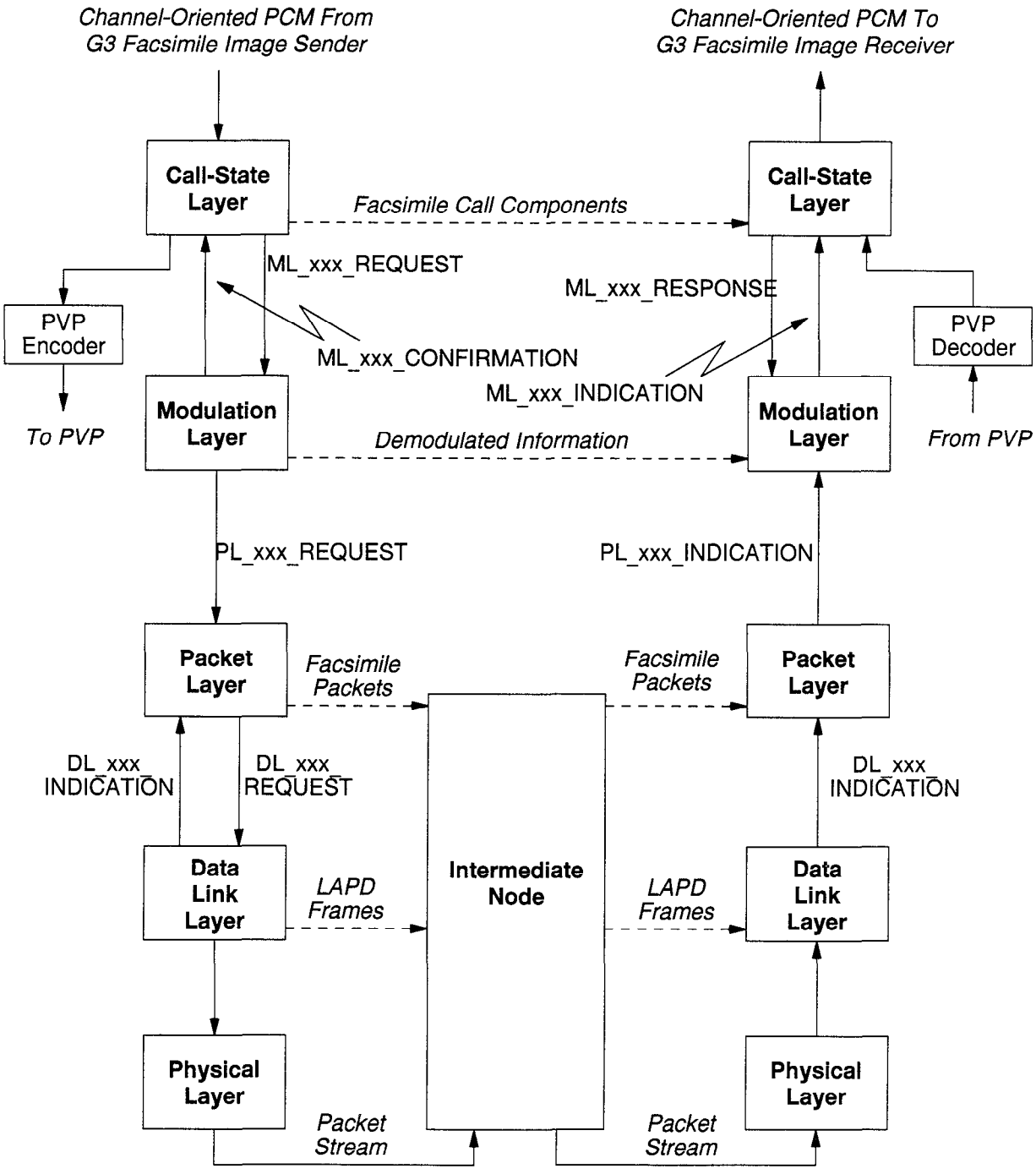
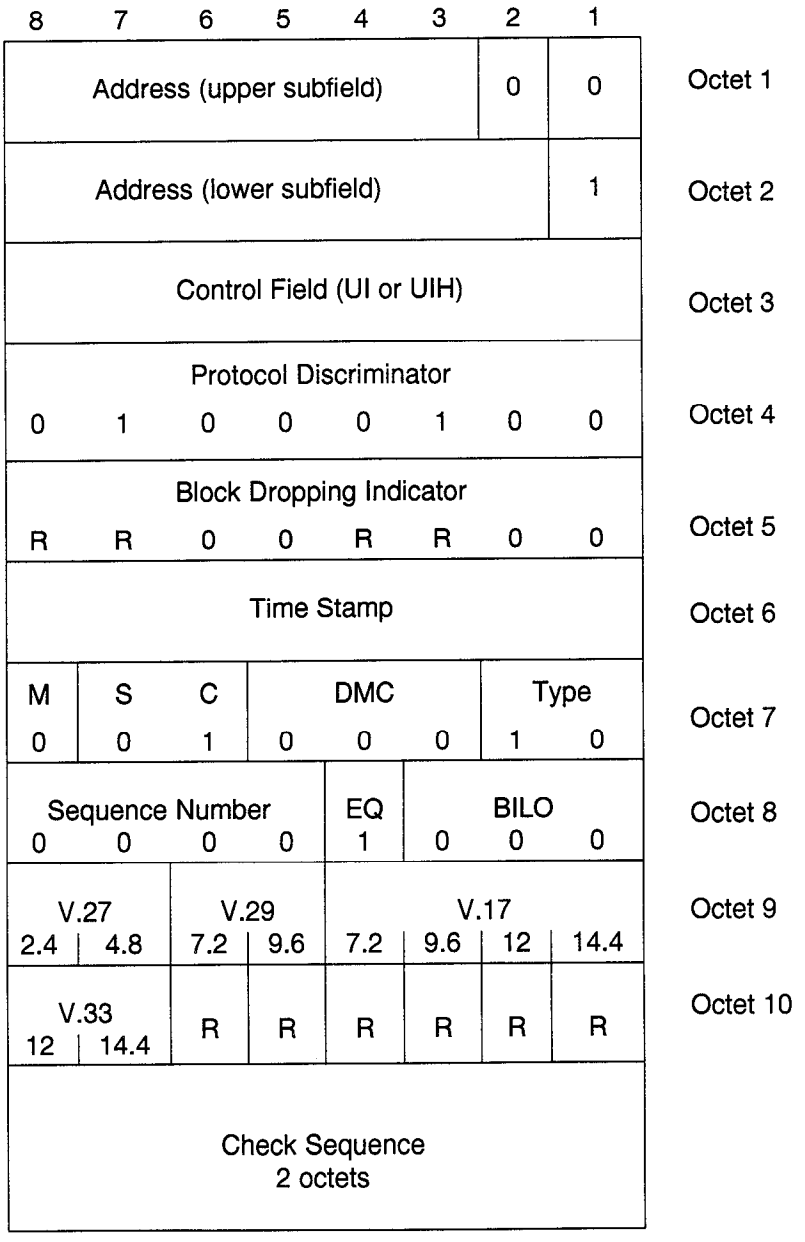
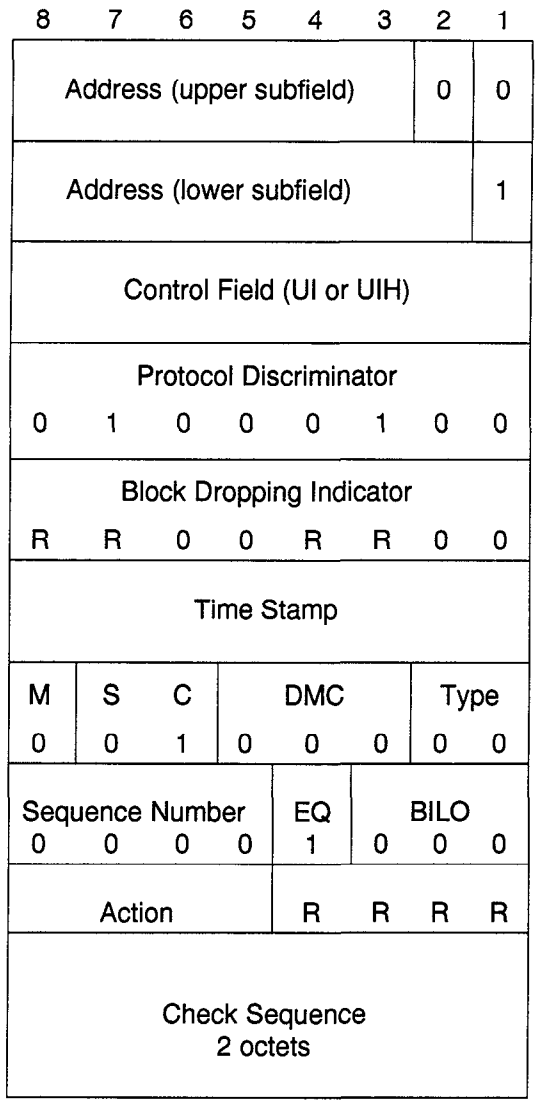


Figure 7 – Facsimile protocol layer model



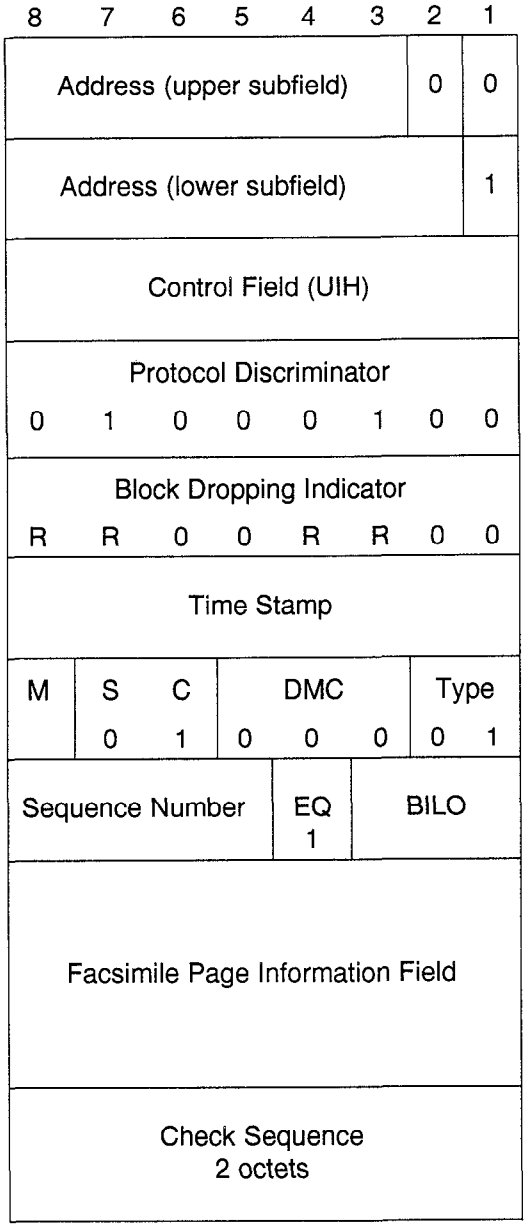
M = M-bit (see 15.3.4)
R = Reserved for future use and set to 0

Figure 8 – Facsimile capability indication frame



M = M-bit (see 15.3.4)
R = Reserved for future use and set to 0

Figure 9 – Facsimile spurt header frame



M = M-bit (see 15.3.4)
R = Reserved for future use and set to 0

Figure 10 – Facsimile page information frame format

15.3.6 Digital modem class

The digital modem class (DMC) field indicates the type of digital modem used for digital modem packets (SC = 01). The following codes are currently used:

Table 8 – Digital modem class field codes

Code	Meaning
000	Facsimile
001	Reserved for future use
010	Reserved for future use
011	Reserved for future use
100	Reserved for future use
101	Reserved for future use
110	Reserved for future use
111	Prohibited

Undefined codes are reserved for future use.

15.3.7 Type

The type field is used to identify the type of packet for a given digital modem class For DMC = 000, the following types are used

Table 9 – Type field codes

Code	Meaning
00	Spurt header packet
01	Page information packet
10	Capability indication packet
11	Reserved for future use

As explained in 15.3 11, the information field of the capability indication and of the spurt header frames (octet 9) contains information that is not available in the V.21 component of the Group III facsimile call.

Page information frames contain demodulated facsimile page information.

15.3.8 Sequence number

Same as for DICE (see 10 3 8)

15.3.9 Delay equalization bit

Same as for DICE (see 10 3 9)

15.3.10 Bits in last octet (BILO)

This field indicates the valid number of bits in the last octet of a Facsimile Page Information frame For V.29 and V.27 ter modulation signals, the definition of its values are as shown in table 10.

Table 10 – BILO definitions for V.29 and V.27 ter

Code	Meaning
000	Information field has even number of symbols
100	Information field has odd number of symbols

For spurt header and capability indication frames, the BILO field is set to zero.

15.3.11 Packet information field

The information carried in the packet information field depends on the packet type, as indicated in the following sections

15.3.11.1 Facsimile capability indication packets

Referring to figure 8, octets 9 and 10 of a facsimile capability indication packet contains the following fields

15.3.11.1.1 V.27

Bits 8 and 7 of octet 9 make up the V 27 field These bits are set to 1 to indicate that the V 27 ter modulation scheme is supported at 2 4 kbit/s and 4 8 kbit/s, respectively Otherwise, they are set to 0

15.3.11.1.2 V.29

Bits 6 and 5 of octet 9 make up the V.29 field. These bits are set to 1 to indicate that the V.29 modulation scheme is supported at 7.2 kbit/s and 9.6 kbit/s, respectively Otherwise, they are set to 0

15.3.11.1.3 V.17

Bits 4 through 1 of octet 9 make up the V.17 field These bits are set to 1 to indicate that the V 17 modulation scheme is supported at 7.2, 9.6, 12, and 14.4 kbit/s, respectively Otherwise, they are set to 0

15.3.11.1.4 V.33

Bits 8 and 7 of octet 10 make up the V.33 field These bits are set to 1 to indicate that the V 33 modulation scheme is supported at 12 and 14.4 kbit/s, respectively Otherwise, they are set to 0

15.3.11.2 Spurt header packets

Referring to figure 9, octet 9 of spurt header packets contains the following fields.

15.3.11.2.1 Action

This field contains the action that the terminating endpoint should perform when the facsimile spurt header packet arrives. See table 11.

Table 11 – Action field codes

Code	Meaning
0001	Generate a training sequence
0010	Abort
0011	Start generating a 1700 Hz Echo Protection Tone (EPT)
0100	Start generating a 1800 Hz Echo Protection Tone (EPT)
0101	Stop generating EPT Tone
1111	Do nothing

The exact response of some of the actions (e.g., generate training sequence) depends on the modulation method of the facsimile call. Modulation techniques other than V.29 and V.27 ter may require different actions. This issue is for further study. Undefined codes are reserved and are left for further study.

15.3.11.3 Page information packets

The format of page information packets is shown in figure 10. In these packets, the facsimile page information field contains the demodulated information. The facsimile page information field shall not exceed 482 octets, as required by ANSI T1.312. The actual size depends on the modulation characteristics

Figure 11 shows the arrangement of the V.29 and V.27 ter symbols in the facsimile page information field.

15.4 Call-state layer procedures

The Call-State Layer tracks the T.30 protocol messages to recognize the beginning of a facsimile call and determine the role of the endpoints for the call (which one is transmitting/demodulating or receiving/remodulating). It also routes the PCM channel-oriented traffic to either the PVP path or the FADCOMP path, and selects the output of either the PVP procedure or FADCOMP remodulation to produce the PCM output. Thus, a single Call-State Layer connects to both the terminating and originating endpoints of a PCME node.

While each of the lower layers has distinct originating and terminating roles, the role of the Call-State Layer dynamically changes. Initially, the Call-State Layer is waiting for a facsimile call to arrive and its role is in the Reset mode. At the very beginning of a facsimile call, the Call-State Layer does not know which endpoint will be the originating (demodulating) endpoint and which will be the terminating (remodulating) endpoint. Furthermore, the Call-State Layer may reverse its role later as the facsimile call progresses.

The service primitives exchanged with both the originating and terminating Modulation Layers are based on the role that the Call-State Layer plays. When the role of the Call-State Layer is Reset, the Call-State Layer does not exchange primitives with either the originating or terminating Modulation Layers. Instead, PCM data is routed to and from PVP.

When the Call-State Layer has the role of a demodulator, it sends and receives primitives to/from the originating endpoint Modulation Layer, and receives facsimile capability primitives from the terminating endpoint Modulation Layer. Depending on the phase of the call, the information arriving from the channel-oriented side is sent to the packetized side either through the voice path or through the demodulation path. Information arriving from the packetized side is sent to the channel-oriented side via the voice path.

When the Call-State Layer has the role of a remodulator, it sends and receives primitives to/from the terminating endpoint Modulation Layer, and sends the facsimile capability primitives to the originating endpoint Modulation Layer. Information from the packetized side is sent to the channel-oriented side either via PVP or through the remodulator, depending on the state of the call. Information from the channel-oriented side is sent to the packetized side via PVP.

15.4.1 Model of the call-state layer

Figure 12, a block diagram model of the Call-State Layer, helps explain the above points. The following subsections describe the individual elements.

Even Number of Symbols

[BILO = 0]

8	7	6	5	4	3	2	1
Symbol 1				Symbol 2			
Symbol 3				Symbol 4			
Symbol n				Symbol n+1			
Symbol n+2				Symbol n+3			

Odd Number of Symbols

[BILO = 4]

8	7	6	5	4	3	2	1
Symbol 1				Symbol 2			
Symbol 3				Symbol 4			
Symbol n				Symbol n+1			
X X X X				Symbol n+2			

Symbol Definitions

Modulation Method	Speed	Symbol Format
V.29	9600	Q4 Q3 Q2 Q1
	7200	0 Q4 Q3 Q2
V.27 ter	4800	0 T3 T2 T1
	2400	0 0 D2 D1

Notes:

1. Q's defined as per CCITT V.29
2. T's defined as tribits as per CCITT V.27 ter. T1 is the left column of Table 1/V.27 ter.
3. D's defined as dibits as per CCITT V.27 ter. D1 is the left column of Table 2/V.27 ter.
4. Q's, T's, and D's as shown after unscrambling
5 X's are Don't Care

Figure 11 – Information field for V.29 and V.27 ter demodulated packets

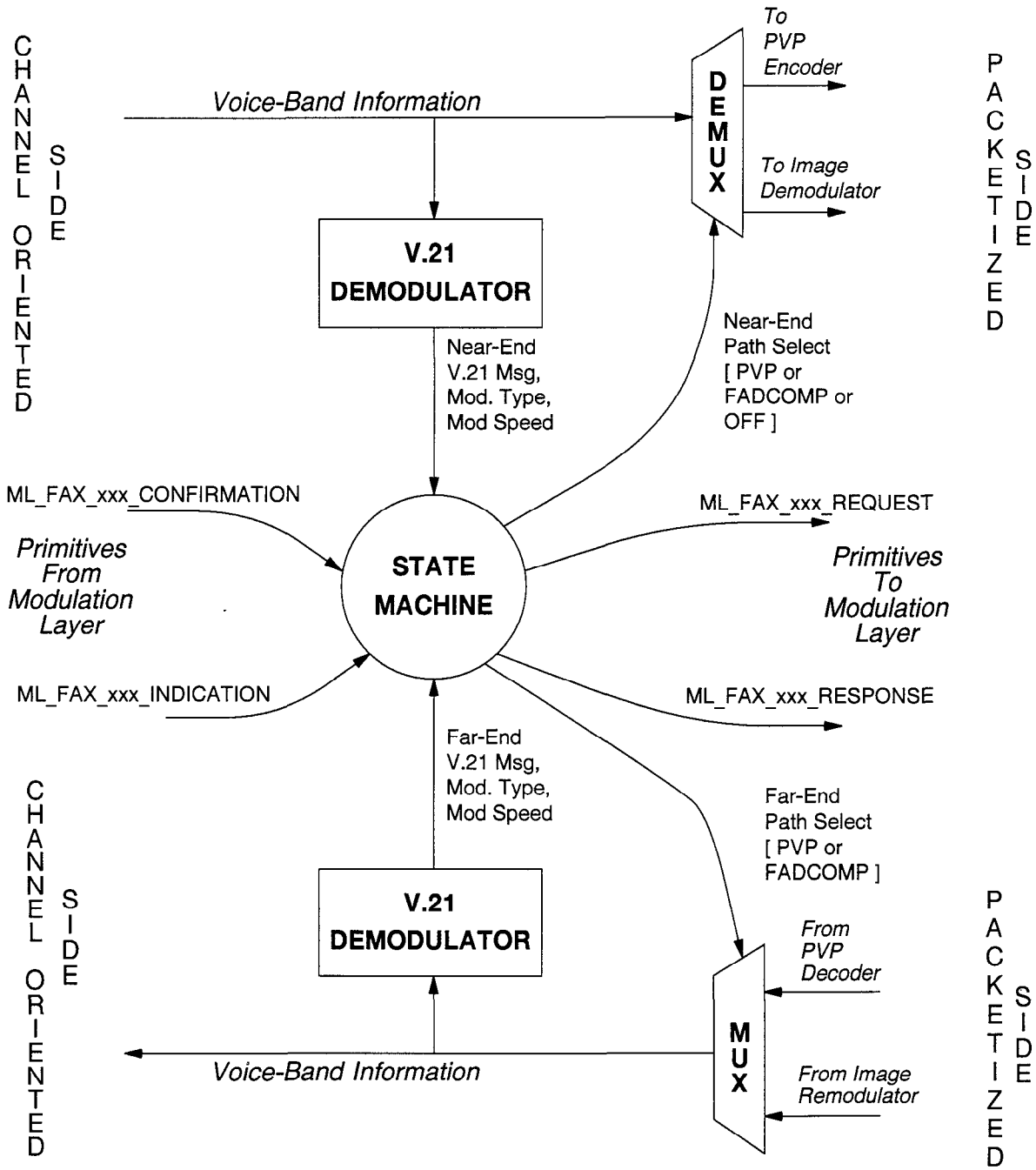


Figure 12 – Call-state layer functional model

15.4.1.1 DEMUX

This block shall route the traffic arriving from the channel-oriented side to either the PVP coder input (PVP) or the Modulation Layer demodulator input (FADCOMP). This block also has capability to inhibit information transfer (OFF), thus providing echo suppression when the Call-State Layer is in the remodulation role. The control signal (PVP, FADCOMP, OFF) is the Near-End Path Select variable generated by the State Machine.

15.4.1.2 MUX

This block is responsible for selecting the appropriate source for information for the channel-oriented side, either the PVP decoder output (PVP) or the Modulation Layer modulator output (FADCOMP). The control signal (PVP, FADCOMP) is the Far-End Path Select variable generated by the State Machine

It is required that MUX have a special fail-safe mechanism to handle the exception case where the terminating end is expecting to remodulate but the originating end is not demodulating. This mechanism is as follows: When the Far-End Path Select is set to FADCOMP, and a PVP packet arrives (which is not normally expected), the resulting PVP decoder output is the source of the channel-oriented information.

15.4.1.3 V.21 demodulators

There is one V 21 demodulator for each PCM direction, monitoring the PCM signal for that direction Each V 21 demodulator is responsible for recognizing the presence of a V 21 modulated signal, extracting legal message frames, and extracting any modem speed and type information that is contained within the message frames Part of the processing includes normal HDLC operations, such as bit-unstuffing and CRC-16 checking. If such operations yield invalid HDLC frames, the message is ignored. The type of message frame and any associated speed or type information is sent to the State Machine. Only signaling with 300 b/s modulation speed is supported Facsimile calls with 2400 b/s signaling shall be processed entirely via PVP

Recommendation T.30 specifies messages that contain modem type and speed information. It is required that the V.21 demodulator extract speed and modem type information from these T.30 messages. However, facsimile calls between machines made by the same manufacturer may have this information embedded in the non-standard facilities T.30 messages. Currently, the extraction of modem speed and type from these non-standard facilities messages is not specified and is left for further study.

15.4.1.4 State machine

The state machine uses message types, modem type and speed indications, and primitives from the originating and terminating modulation layers to track the state of a facsimile call. It sets the Near-End and Far-End Path Select variables and issues primitives to the originating and terminating Modulation Layers Subclause 15.4.2 contains an exact description of the state machine.

15.4.2 Description of the state machine

Figures 13 through 23 show the state transitions in the state machine in response to V.21 messages and primitives from the Modulation Layer The state machine, as described in this section, assumes that both ends have identical resources and that the resources needed are always available Recovery actions when the resources are not available or are not compatible are left for further study

While *Fax Capability* primitives have been incorporated into the Modulation Layer and below, the details of the response of the state Machine to these primitives are under study. The general flow is as follows The remodulating PCME endpoint sends a Capability Indication Frame to the demodulating endpoint after it sees a T.30 Digital Identification Signal (DIS) V.21 frame The demodulating PCME endpoint uses this information, knowledge of its own capabilities, and modulation type and speed information obtained from subsequent V 21 messages to determine whether to demodulate or send the page information through PVP. The fail-safe mechanism in DEMUX allows the remodulating PCME endpoint to properly transport the page information to the receiving facsimile equipment in either case

As indicated in 15 4 1 3, the V 21 demodulators will determine the modulation type and speed of all G3 standard-protocol facsimile calls, including V 29, V 27 ter, V 33, and V.17. Currently, only V 29 and V.27 ter are covered by this protocol. Other modulation types are left for further study and currently undemodulated.

In Figures 13 through 23, the following nomenclature should be observed:

- V.21 messages are noted by capitalized abbreviations, such as NSS or DTC;
- V.21 messages prefixed by a hyphen are messages with the FINAL bit set and are therefore "final frames." As defined in 5.3.5 of CCITT Recommendation T.30, a final frame is the last frame transmitted prior to an expected response from the distant station. In the text itself, it is assumed that all V.21 messages are final frames, unless explicitly stated;
- V.21 messages are shown as coming from the near-end (channel-oriented side) or the far-end (packetized side);
- For ease of reference, the state labels contain two parts:
  - the prefix "t" or "r", representing a transmitting/demodulating or receiving/remodulating endpoint;
  - a state identifier consisting of a letter and a number; the letter identifies a state and the number defines a sub-state within that state.

The state machine is described in four parts:

- global actions that occur in most states;
- the initial state, which is common to both demodulating and remodulating endpoints (state a1);
- a set of states that are followed by the demodulating (transmitting) endpoint (states t.a1 through t.g3);
- a set of states that are followed by the remodulating (receiving) endpoint (states r.a1 through r.g3).

15.4.2.1 Global actions

All states, other than a1, t.b1, t.c1, r.b1, and r.c1, have an associated timer that is set when the state is entered. If a T.30 message is not received before the timer expires, the state machine shall release all resources and transition to the WAIT_FOR_FAX_CALL state (a1). This is a fail-safe mechanism; the value of the timer for each state is under study.

While in all states, other than a1, t.b1, t.c1, r.b1, and r.c1, if a final DCN or DIS T.30 message is received from either side, the state machine shall release all resources and transition to the WAIT_FOR_FAX_CALL state (a1).

15.4.2.2 Actions in the initial state

In the initial state, WAIT_FOR_FAX_CALL (a1), both the Multiplexer and Demultiplexer are in the PVP position so that the traffic follows the rules specified in ANSI T1.312. The state machine shall observe the traffic to detect the presence of the T.30 Digital Identification Signal (DIS). If this signal arrives, then:

a) If the signal arrives from the channel-oriented (near-end) side, the state machine shall check that the remodulating resources are available. If they are available, it allocates the necessary resources. It then assumes the remodulating (receiving) endpoint role and moves to the WAIT_FOR_DCS_FROM_FAR_END state (r.a2);

b) If the signal arrives from the packetized (far-end) side, the state machine shall check that the demodulator resources are available. If they are available, it allocates these resources. It then assumes the demodulating (transmit) endpoint role and moves to the WAIT_FOR_DCS_FROM_NEAR_END state (t.a2).

15.4.2.3 Actions in the demodulating endpoint states

15.4.2.3.1 Actions in the WAIT_FOR_DCS-_FROM_NEAR_END state

In the WAIT_FOR_DCS_FROM_NEAR_END state (t.a2), each endpoint shall check whether traffic coded according to the modulation type and speed that have been provided by the corresponding V.21 demodulator can be demodulated with the available resource. This information is extracted either from the Digital Command Signal (DCS) of T.30 or from the information embedded in the Nonstandard Setup (NSS), both arriving from the V.21 demodulator of the channel-oriented side. If compatibility has been verified, then the state machine shall:

a) store the modem speed and type in the system variables ORIG_SPEED and ORIG_TYPE, respectively;

b) put the Demultiplexer in the FADCOMP position to route the information from the channel-oriented side to FADCOMP. The Call-State Layer shall send the ML_FAX_START_REQUEST(ORIG_SPEED, ORIG_TYPE) primitive to the Modulation Layer of its corresponding originating endpoint.

The Call_State Layer shall set the system variable PART_PAGE_REQ_CNT to 0 and shall move to the WAIT_FOR_TCF_TO_END state (t.b1).

If the modulation type and speed as extracted from the T.30 message is not compatible with the PCME's resources, all resources are released and the state machine moves back to the WAIT_FOR_FAX_CALL state (a1).

If an NSC or DTC T.30 message is received from the channel-oriented side, the machine changes its role to that of a remodulating endpoint and moves to the WAIT_FOR_DCS_FROM-_FAR_END state (r.a2).

15.4.2.3.2 Actions in the WAIT_FOR-_TCF_TO_END state (t.b1)

In this state, the Call-State Layer shall wait for the primitives from the corresponding Modulation Layer. If either V.21 demodulator indicates that V.21 is coming, the Call-State Layer shall put both the Multiplexer and the Demultiplexer in the PVP mode. It shall send the ML_FAX_STOP_REQUEST to indicate to the Modulation Layer of the demodulating endpoint to return to the OFF state. It shall release all resources and return to the WAIT_FOR_FAX_CALL state (a1).

If the ML_FAX_STOP_CONFIRMATION primitive arrives from the Modulation Layer of the originating endpoint, the Call-State Layer shall put both the Multiplexer and the Demultiplexer in the PVP mode. It shall then move to the WAIT_FOR_CFR (t.b2) state.

If the ML_FAX_ABORT_CONFIRMATION arrives from the Modulation Layer of the originating endpoint, the Call-State Layer shall put both the Multiplexer and the Demultiplexer in the PVP mode. It shall release all resources and return to the WAIT_FOR_FAX_CALL (a1) state.

15.4.2.3.3 WAIT_FOR_CFR state (t.b2)

If either DCS or NSS signals arrive from the channel-oriented side, the Call-State Layer shall put both the Multiplexer and the Demultiplexer in the PVP mode. It shall release all resources and return to the WAIT_FOR_FAX_CALL (a1) state.

If the Confirmation to Receive (CFR) arrives from the packetized side, the state machine shall put the Demultiplexer in the FADCOMP position to route the information from the channel-oriented side through FADCOMP. The Call-State Layer shall send the ML_FAX_START_REQUEST-(ORIG_SPEED,ORIG_TYPE) primitive to the Modulation Layer of its corresponding originating endpoint. The state machine shall then move to WAIT_FOR_PAGE_END (t.c1) state.

If the V.21 demodulator indicates the arrival of any of the following messages from the packetized side, the state machine shall return to the WAIT_FOR_DCS_FROM_NEAR_END (t.a2) state: Command Repeat (CRP), Failure to Train (FTT), Nonstandard facilities command (NSC), or Digital Transmit Command (DTC).

15.4.2.3.4 WAIT_FOR_END_OF_PAGE state (t.c1)

In this state, the Call-State Layer shall wait for the primitives from the corresponding Modulation Layer. If either V.21 demodulator indicates that V.21 is coming, the Call-State Layer shall put both the Multiplexer and the Demultiplexer in the PVP mode. It shall send the ML_FAX_STOP_REQUEST to indicate to the Modulation Layer of the demodulating endpoint to return to the OFF state. It shall release all resources and return to the WAIT_FOR_FAX_CALL (a1) state.

If the ML_FAX_STOP_CONFIRMATION primitive arrives from the Modulation Layer of the originating endpoint, the Call-State Layer shall put both the Multiplexer and the Demultiplexer in the PVP mode. It shall then move to the WAIT_FOR_POST_PAGE_CMD (t.c2) state.

If the ML_FAX_ABORT_CONFIRMATION arrives from the Modulation Layer of the originating endpoint, the Call-State Layer shall put both the Multiplexer and the Demultiplexer in the PVP mode. It shall release all resources and return to the WAIT_FOR_FAX_CALL (a1) state.

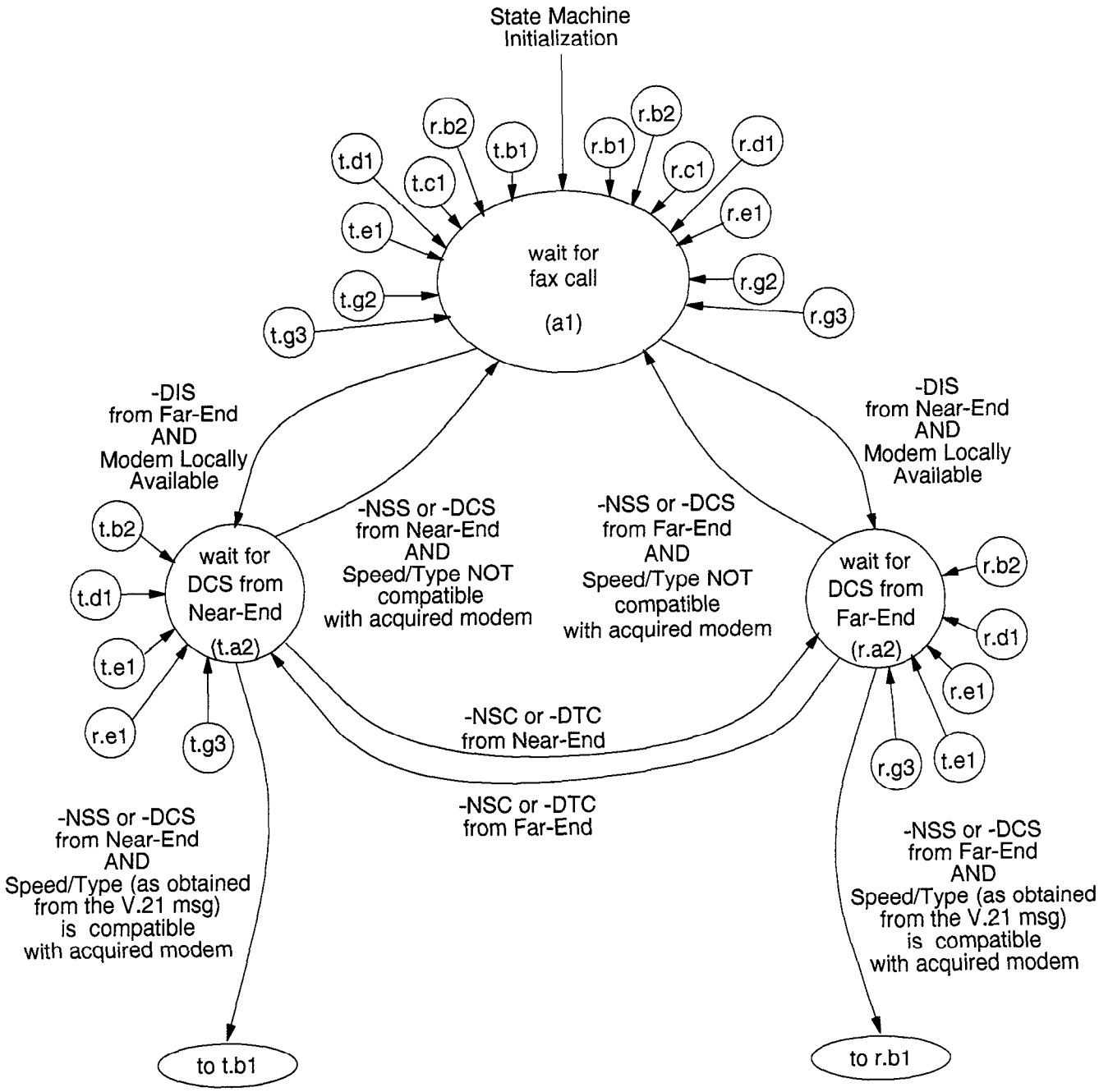


Figure 13 – Call-state layer state machine (A)

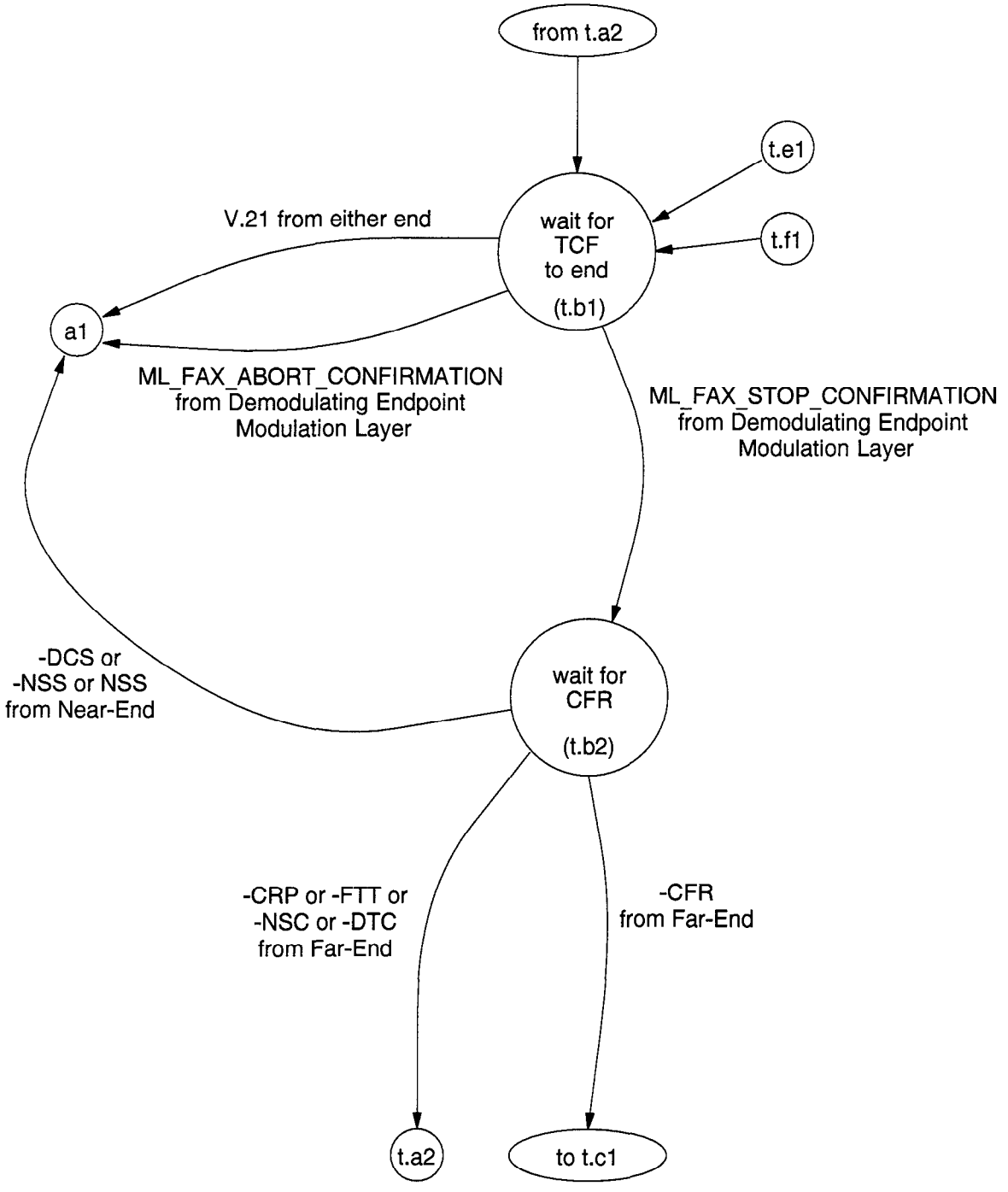


Figure 14 – Call-state layer state machine (State t.b)

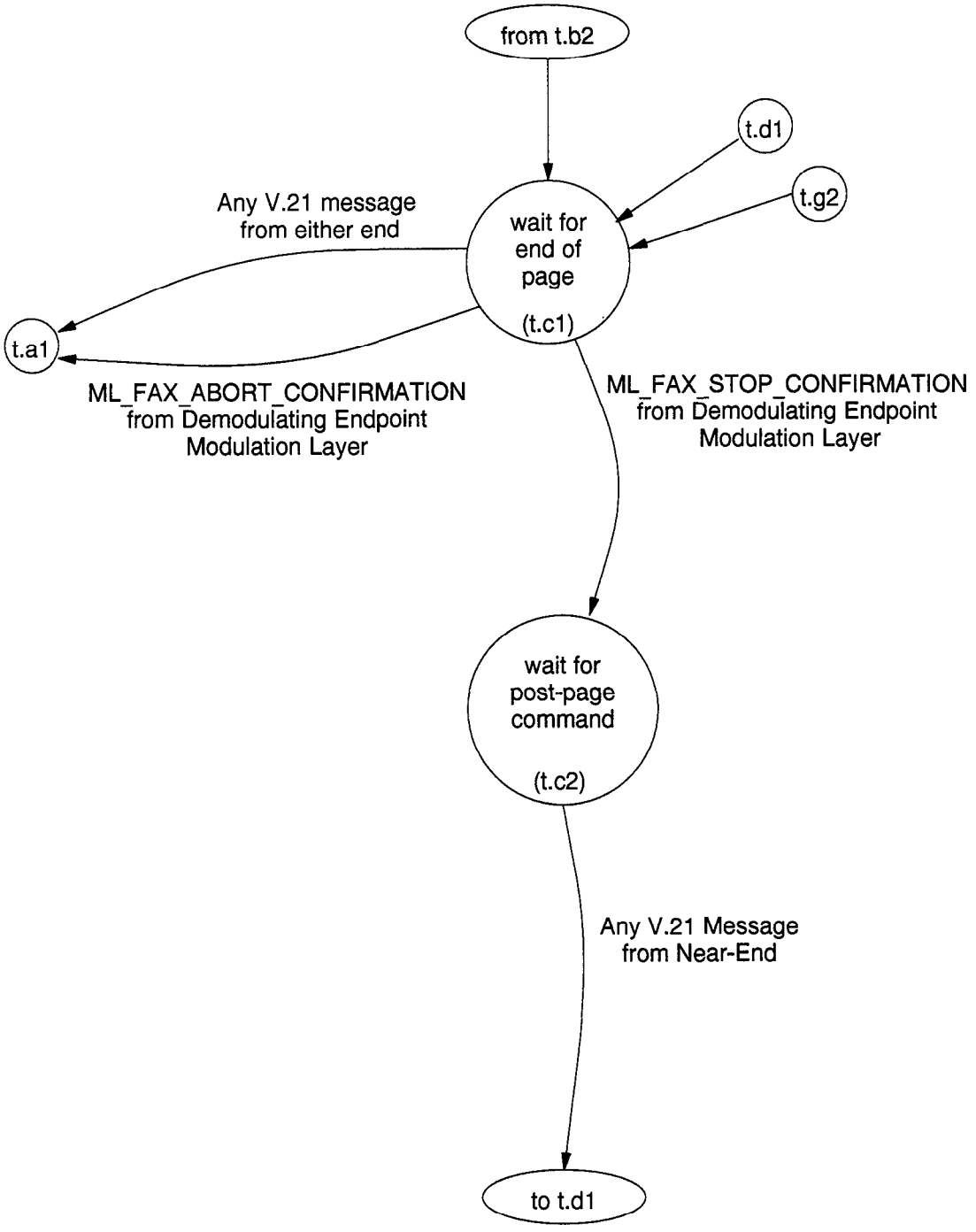


Figure 15 – Call-state layer state machine (State t.c)

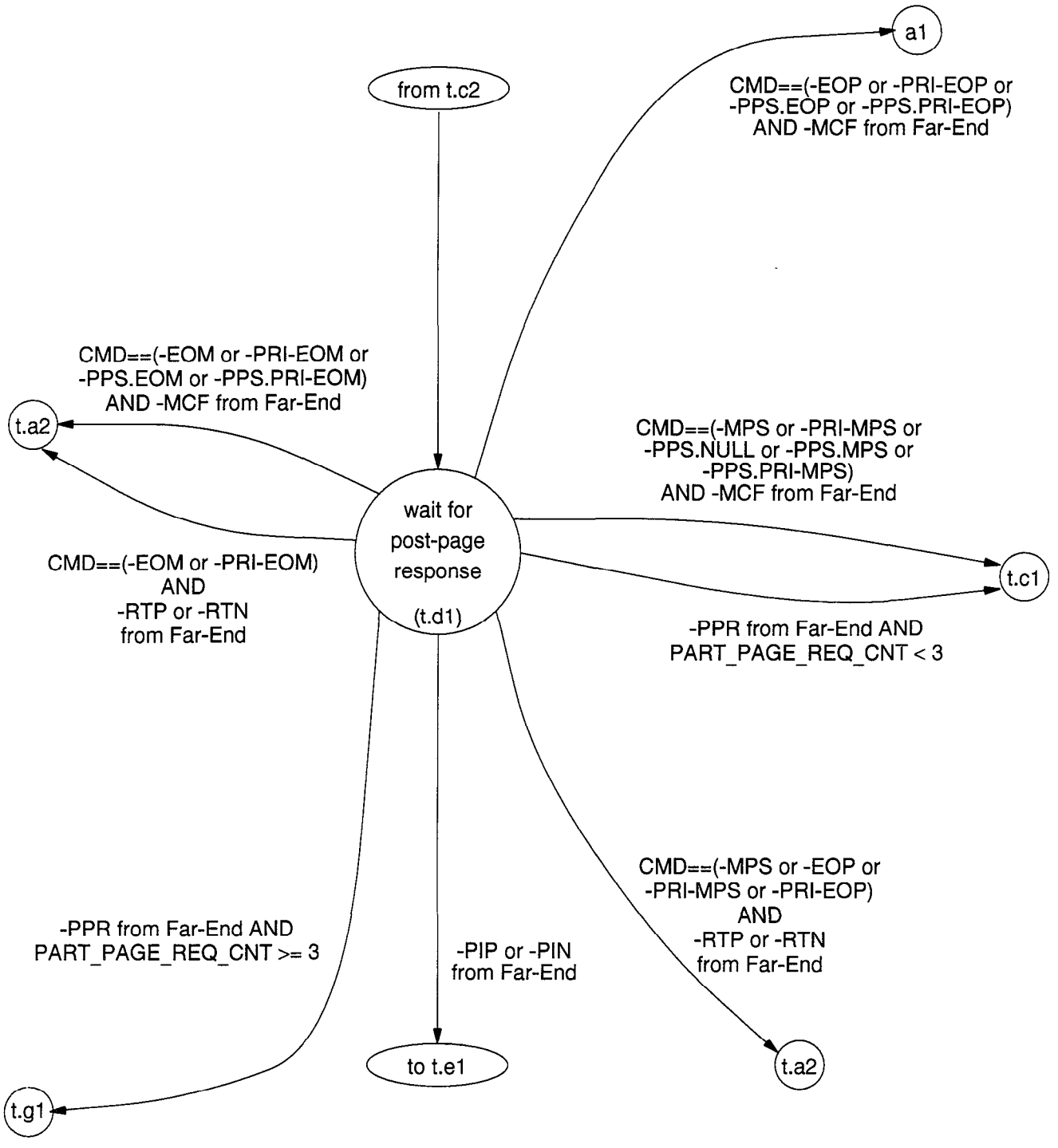


Figure 16 – Call-State Layer State Machine (State t.d)

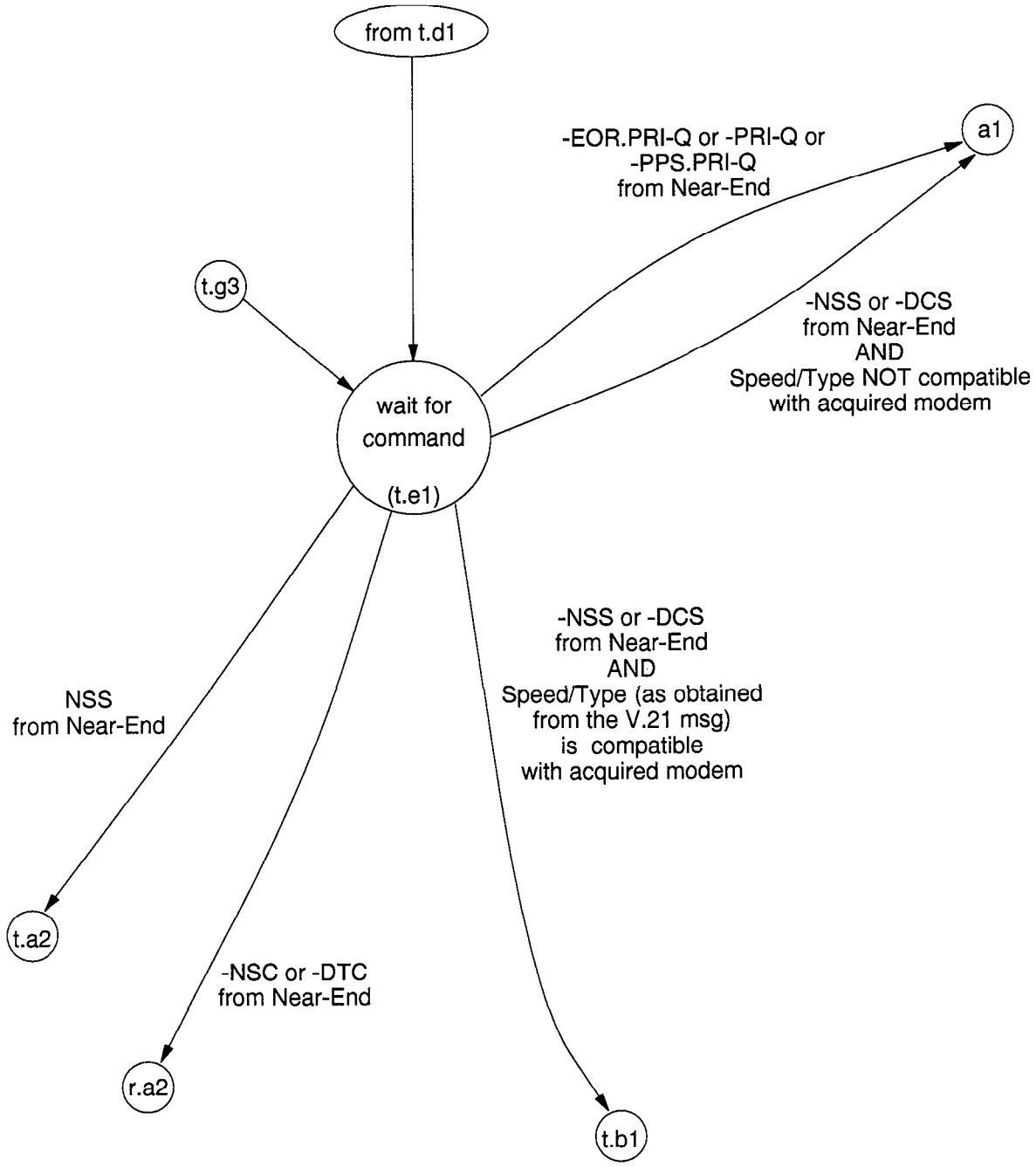


Figure 17 – Call-state layer state machine (State t.e)

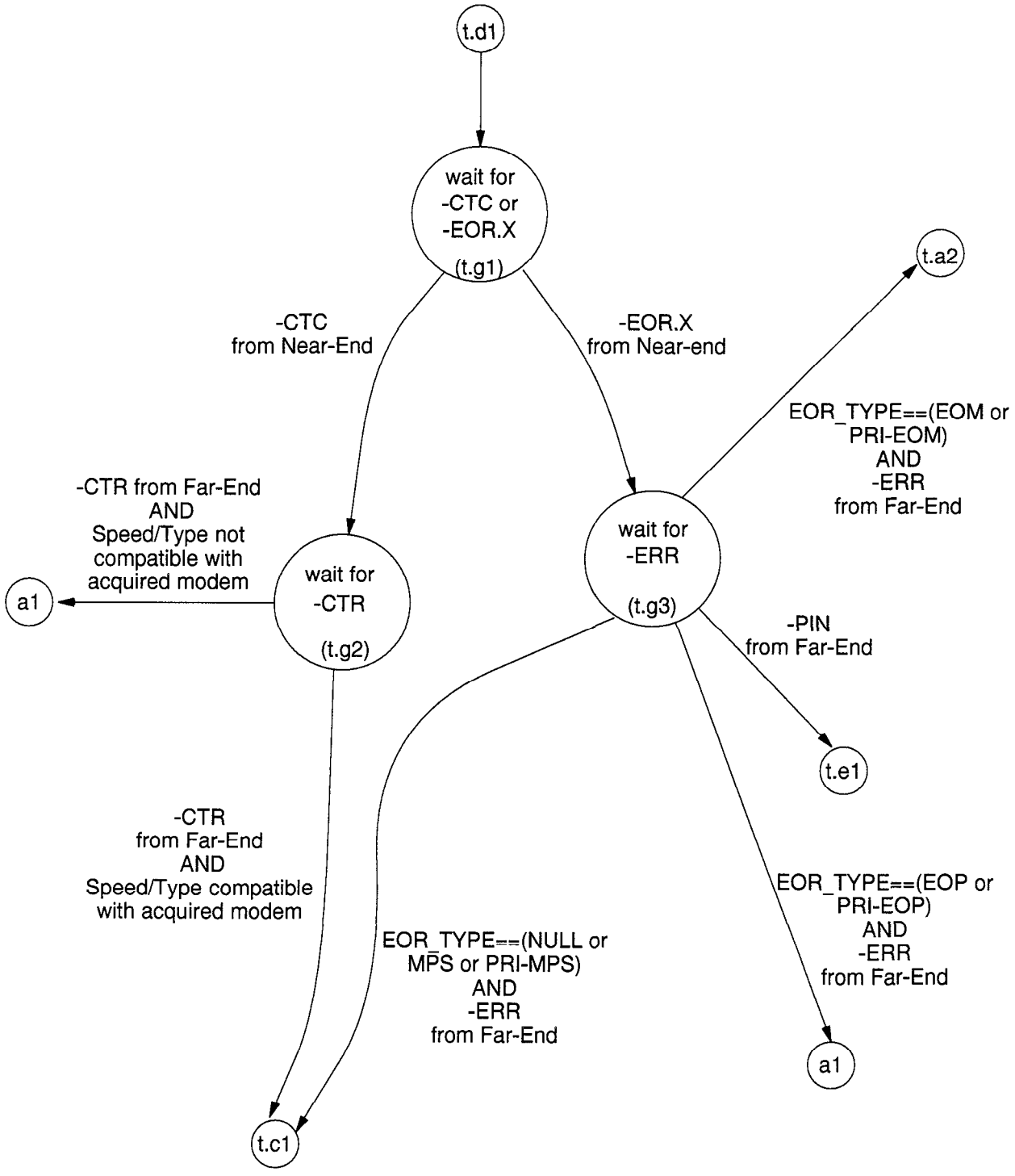


Figure 18 – Call-state layer state machine (State t.g)

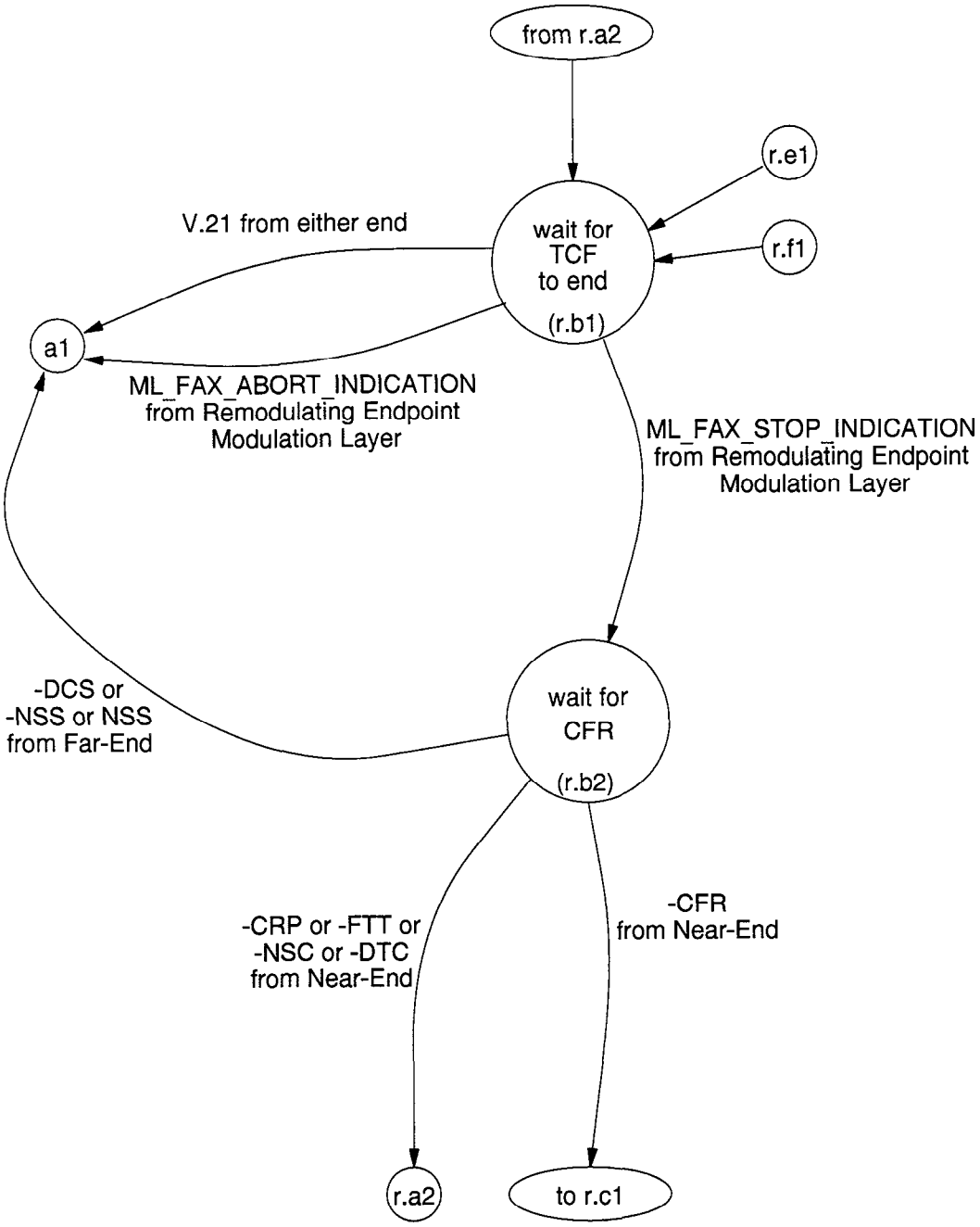


Figure 19 – Call-state layer state machine (State r.b)

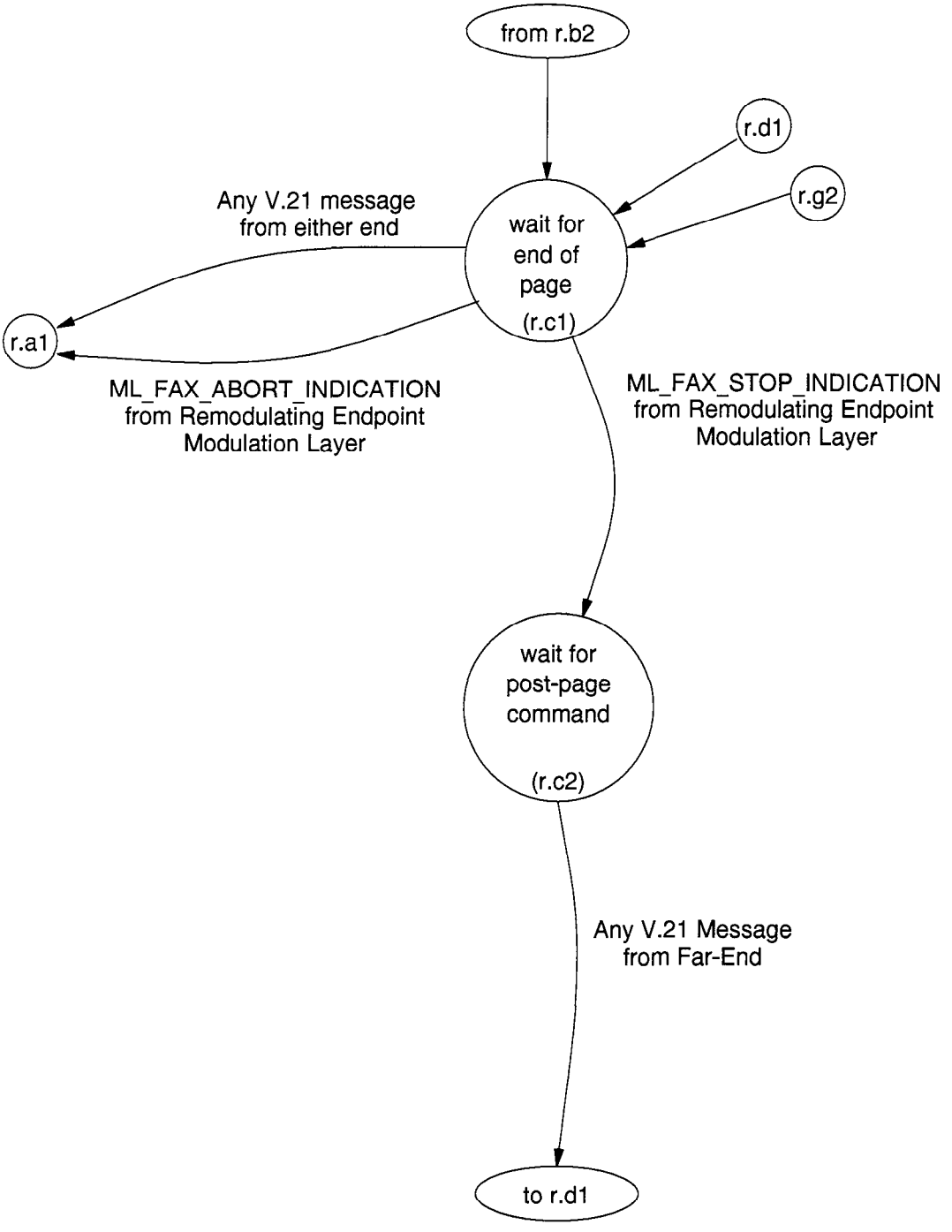


Figure 20 – Call-state layer state machine (State r.c)

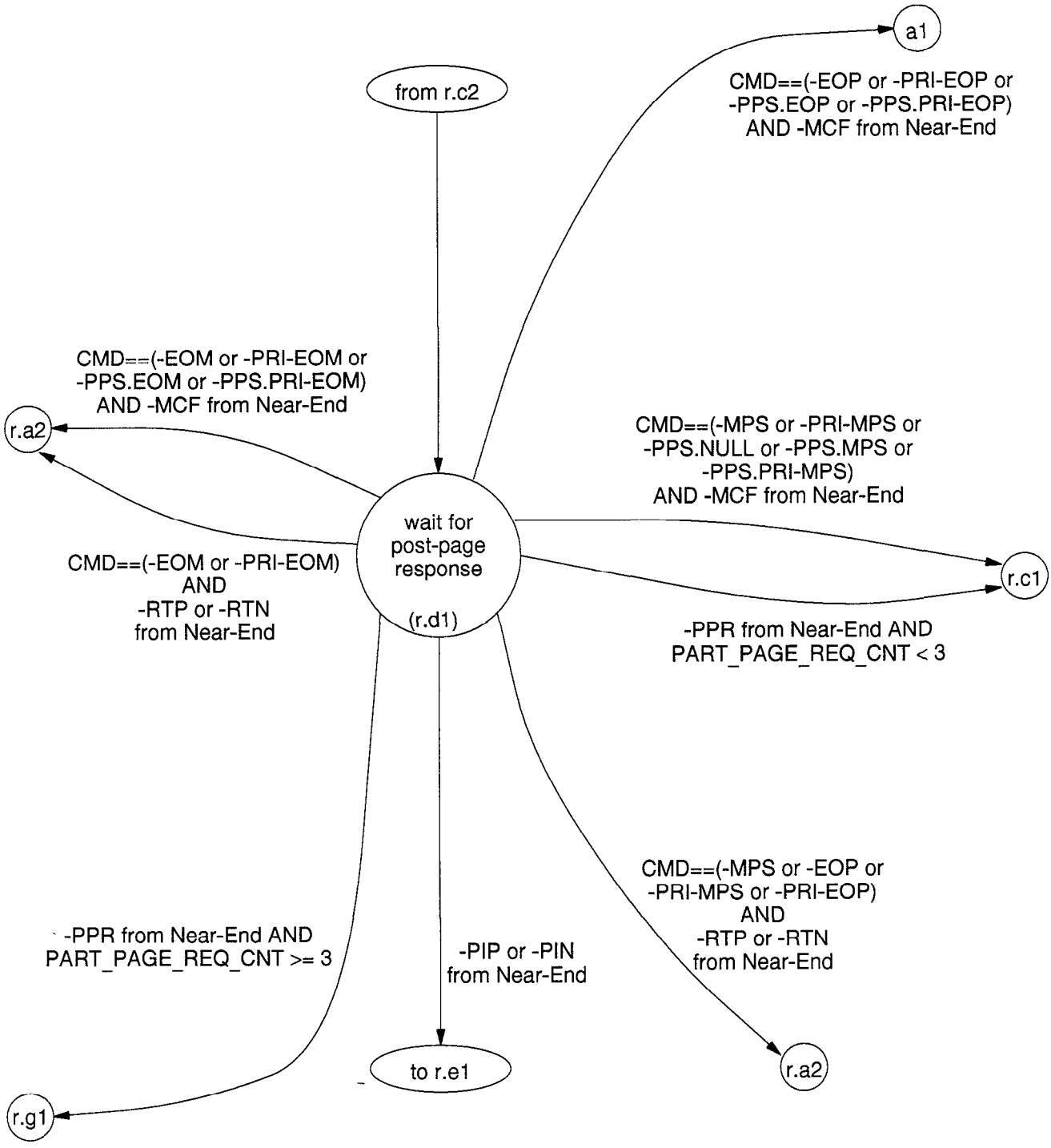


Figure 21 – Call-state layer state machine (State r.d)

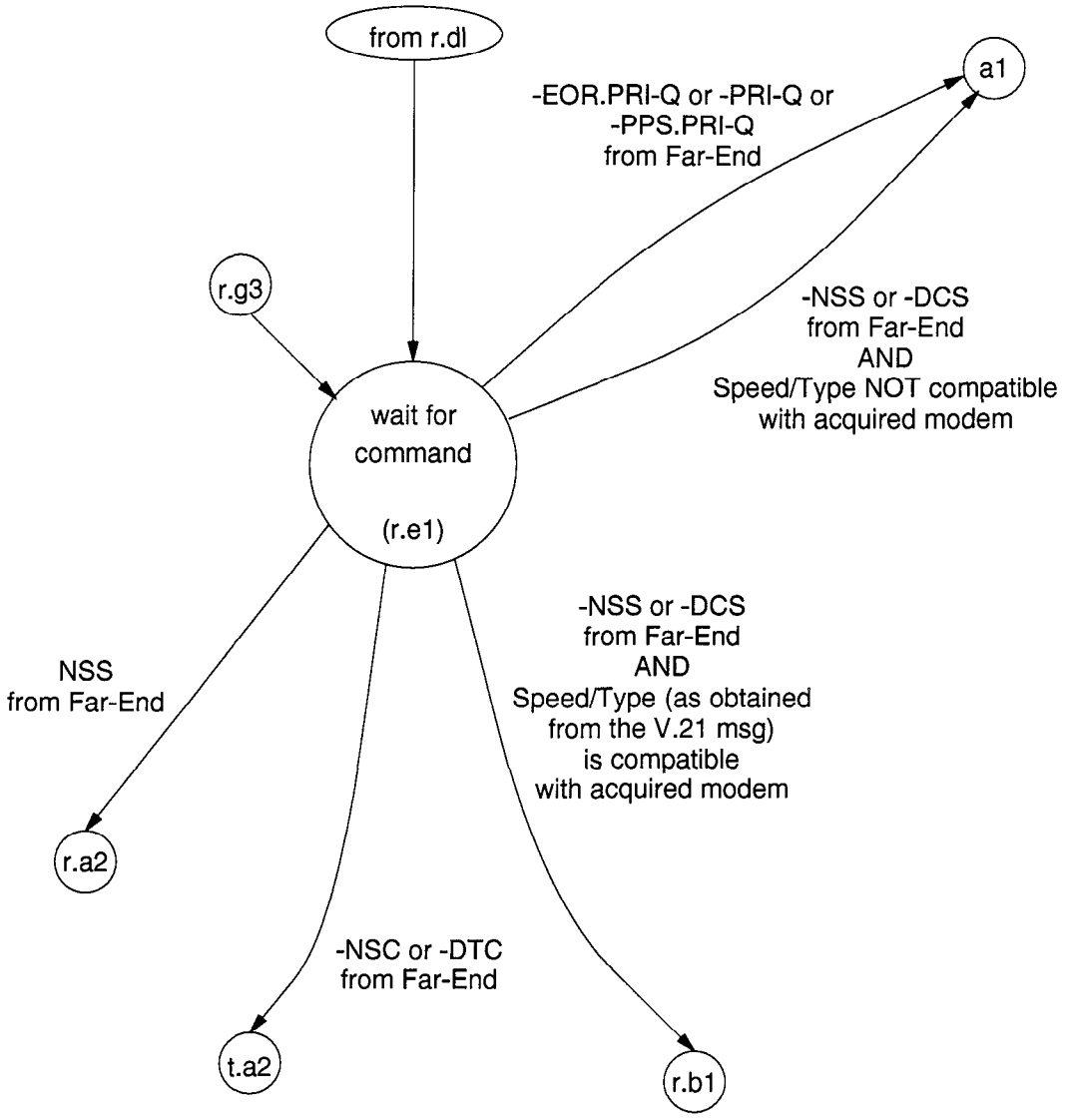


Figure 22 – Call-state layer state machine (State r.e)

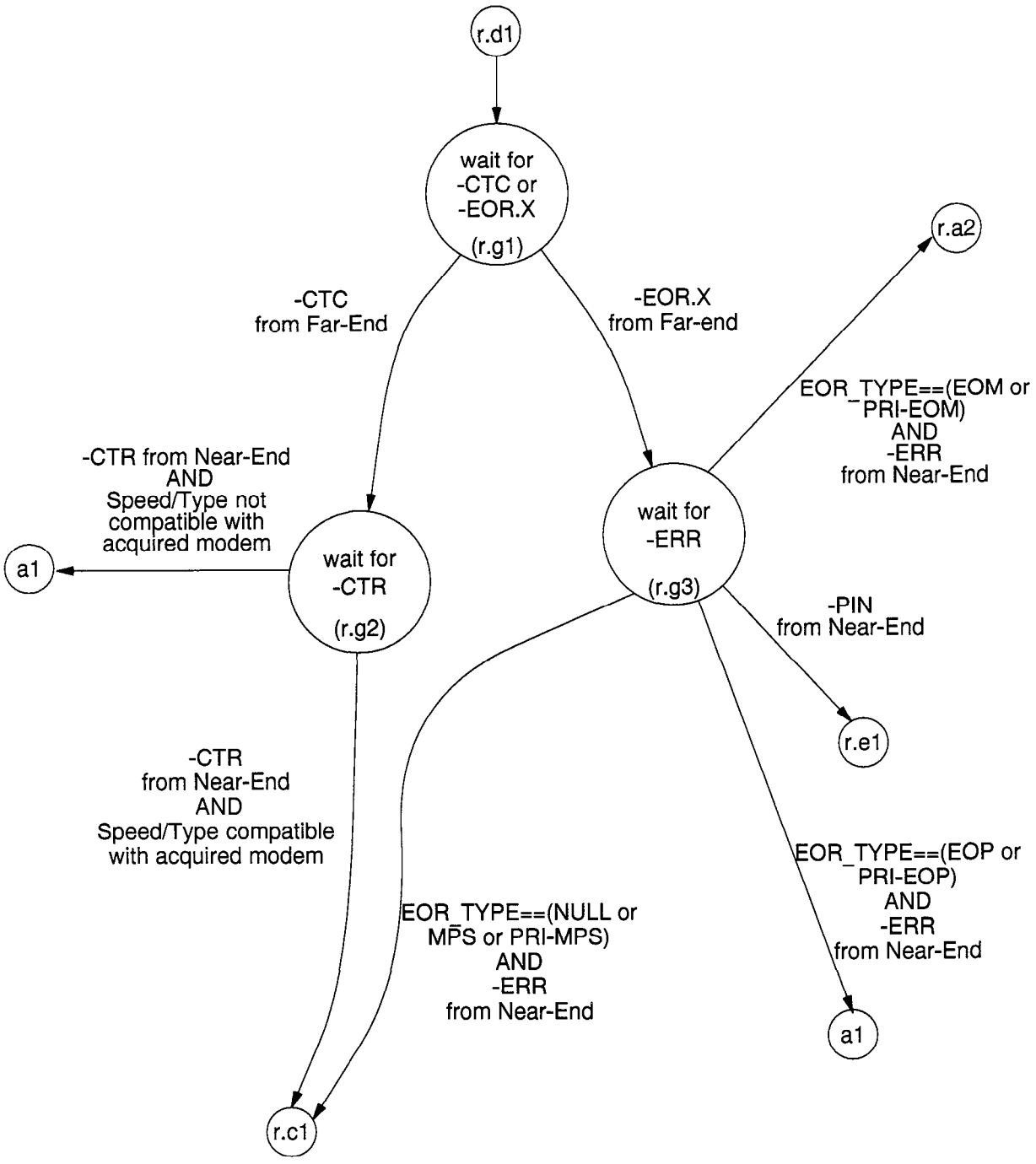


Figure 23 – Call-state layer state machine (State r.g)

15.4.2.3.5 WAIT_FOR_POST_PAGE_COMMAND state (t.c2)

In this state, the state machine is awaiting a V.21 message from the channel-oriented side. This message will be called subsequently the "post-page command." Upon receipt of the post-page command, the state machine moves to the WAIT_FOR_POST_PAGE_RESPONSE state (t.d1).

15.4.2.3.6 WAIT_FOR_POST_PAGE_RESPONSE state (t.d1)

In this state, the state machine is awaiting specific messages from the packetized side, which represent the response to the previously received post-page command. The combination of the command and response determine the actions.

15.4.2.3.6.1 End of procedure

The end of procedure is indicated by one of the following post-page commands: End of Procedure (EOP), Procedure Interrupt-EOP (PRI-EOP), Partial Page Signal-EOP (PPS-EOP), and PPS-PRI-EOP. If the post-page response is Message Confirmation (MCF), the state Machine releases all allocated resources and returns to the WAIT_FOR_FAX_CALL (a1) state.

15.4.2.3.6.2 End of message

The end of message is indicated by one of two combinations:

– one of the following commands: End of Message (EOM), Procedure Interrupt-EOM (PRI-EOM), Partial Page Signal-EOM (PPS-EOM), and PPS.PRI-EOM, along with the Message Confirmation (MCF) response;

– one of the following commands: End of Message (EOM) or Procedure Interrupt-EOM (PRI-EOM), along with one of the following responses: Retrain Positive (RTP) or Retrain Negative (RTN).

In either case, the state machine moves to the WAIT_FOR_DCS_FROM_NEAR_END state (t.a2).

15.4.2.3.6.3 Multiple page

The multiple page case is indicated by one of two combinations:

– one of the following commands: Multipage Signal (MPS), Procedure Interrupt-MPS (PRI-MPS), Partial Page Signal-MPS (PPS-MPS), PPS.PRI-MPS, PPS-Null, along with the Message Confirmation (MCF) response. In this case, the PART_PAGE_REQ_CNT variable is reset to 0;

– the response is a Partial Page Request (PPR) while the partial page received count PART_PAGE_REQ_CNT is less than 3. The post-page command is irrelevant. In this case, the PART_PAGE_REQ_CNT variable is incremented.

Additional actions are as follows:

a) Put the Demultiplexer in the FADCOMP position to route the information from the channel-oriented side through FADCOMP;

b) Send the ML_FAX_START_REQUEST-(ORIG_SPEED,ORIG_TYPE) primitive to the Modulation Layer of its corresponding originating endpoint;

c) Move to the WAIT_FOR_END_OF_PAGE state (t.c1).

15.4.2.3.6.4 Partial page request

This is the case when the response PPR is detected while the PART_PAGE_REQ_CNT has reached the limit of 3, regardless of the post-page command. The PART_PAGE_REQ_CNT variable is reset to 0 and the state changes to the WAIT_FOR_CTC_OR_EOR.X state (t.g1)

15.4.2.3.6.5 Procedure to retrain

This is the case when one of the following commands is detected: Multipage Signal (MPS), End of Procedure (EOP), Procedure Interrupt-MPS (PRI-MPS), Procedure Interrupt-EOP(PRI-EOP), along with either Retrain Negative (RTN) or Retrain Positive (RTP). In this case, the state changes to the WAIT_FOR_DCS_FROM_NEAR-END state (t.a2).

15.4.2.3.6.6 Procedure interrupt

This is the case when the response is either Procedure Interrupt Negative (PIN) or Procedure Interrupt Positive (PIP), regardless of the post-page command. The state changes to the WAIT_FOR_CMD state (t.e1).

15.4.2.3.7 WAIT_FOR_CMD state (t.e1)

If the V.21 demodulator indicates the arrival of a "non-final" Nonstandard Setup (NSS) from the channel-oriented side, the state machine shall return to the WAIT_FOR_DCS_FROM_NEAR-END state (t.a2).

If the V.21 demodulator indicates the arrival of either the Nonstandard facilities command (NSC) or the Digital Transmit Command (DTC) from the channel-oriented side, the state machine shall switch the role of the node to a remodulating node and shall move to the WAIT_FOR_DCS_FROM_FAR_END state (r a2).

If the V.21 demodulator indicates the arrival of either the Nonstandard Setup (NSS) or the Digital Command Signal (DCS) from the channel-oriented side, and the modulation type and speed extracted from the V.21 information are compatible with the allocated modem, the state machine shall:

a) store the modem speed and type in the system variables ORIG_SPEED and ORIG_TYPE, respectively;

b) put the Demultiplexer in the FADCOMP position to route the information from the channel-oriented side through FADCOMP;

c) send the ML_FAX_START-_REQUEST(ORIG_SPEED,ORIG_TYPE) primitive to the Modulation Layer of its corresponding originating endpoint;

d) set PART_PAGE_REQ_CNT to 0,

e) move to the WAIT_FOR_TCF_TO_END state (t.b1).

If the V.21 demodulator indicates the arrival of either the Nonstandard Setup (NSS) or the Digital Command Signal (DCS) from the channel-oriented side, and the modulation type and speed extracted from the V.21 information are not compatible with the allocated modem, all resources are released and the state machine moves back to the WAIT_FOR_FAX_CALL (a1) state.

If the V.21 demodulator indicates the arrival of EOR.PRI-Q, PRI-Q or PPS PRI-Q from the channel-oriented side, all resources are released and the state machine moves back to the WAIT_FOR_FAX_CALL (a1) state As defined in A7.1 of T.30, PRI-Q is a general term referring to either PRI-ROM, PRI-MPS or PRI-EOP post message command. These commands are used in the optional T.4 error correction mode. PPS.PRI-Q could be either PPS PRI-EOM, PPS.PRI-MPS or PPS PRI-EOP post message command.

15.4.2.3.8 WAIT_FOR_CTC_OR_EOR.X state (t.g1)

If the V.21 demodulator indicates the arrival of the Continue to Correct (CTC) from the channel-oriented side, the state machine shall set the speed to ORIG_SPEED and modulation type to ORIG_TYPE as extracted by the V.21 demodulator, and shall move to the WAIT_FOR_CTR state (t.g2).

If the V 21 demodulator indicates the arrival of EOR.X signal (End of Retransmission = EOR) from the channel-oriented side, where X is either Q or PRI-Q, it shall move to the WAIT_FOR_ERR state (t.g3). As defined in A7.1 of T.30, EOR Q represents either of the post-message commands EOR.EOM, EOR MPS, EOR.EOP, or EOR.Null. EOR.PRI-Q represents either EOR PRI-EOM, EOR PRI-MPS, or EOR PRI-EOP X is referred to as "EOR_TYPE" subsequently.

15.4.2.3.9 WAIT_FOR_CTR state (t.g2)

If the V.21 demodulator indicates the arrival of the Response to Continue to Correct (CTR) from the packetized side, and the speed and modulation type are compatible with the allocated modem, the state machine shall.

a) store the modem speed and type in the system variables ORIG_SPEED and ORIG_TYPE, respectively;

b) put the Demultiplexer in the FADCOMP position to route the information from the channel-oriented side through FADCOMP;

c) send the ML_FAX_START-_REQUEST(ORIG_SPEED,ORIG_TYPE) primitive to the Modulation Layer of its corresponding originating endpoint;

d) set PART_PAGE_REQ_CNT to 0,

e) move to the WAIT_FOR_END_OF_PAGE state (t c1)

If the V.21 demodulator indicates the arrival of the Response to Continue to Correct (CTR) from the packetized side, and the modulation type and speed extracted from the V.21 information are not compatible with the allocated modems, all

resources are released and the state machine moves back to the WAIT_FOR_FAX_CALL state (a1).

15.4.2.3.10 WAIT_FOR_ERR state (t.g3)

If the V.21 demodulator indicates the arrival of a Response to End of Retransmission (ERR) from the packetized side and the EOR type as set in state t.g1 is either EOM or PRI-EOM, the state machine shall move to WAIT_FOR_DCS_FROM_NEAR_END state (t.a2).

If the terminating endpoint V.21 demodulator indicates the arrival of a Response to End of Retransmission (ERR) from the packetized side, and the EOR type as set in state t.g1 is either Null, MPS or PRI-MPS, the state machine shall:

a) set the Demultiplexer to FADCOMP;

b) send the primitive ML_FAX_START-_REQUEST(ORIG_SPEED,ORIG_TYPE) to the Modulation Layer of the originating endpoint;

c) move to the WAIT_FOR_END_OF_PAGE state (t.c1).

If the V.21 demodulator indicates the arrival of a Procedure Interrupt Negative (PIN) from the packetized side, the state machine shall return to the WAIT_FOR_CMD state (t.e1).

If the V.21 demodulator indicates the arrival of Response for End of Retransmission (ERR) from the packetized side and the EOR type as set in state t.g1 is either EOP or PRI-EOP, the state machine shall set the Multiplexer and Demultiplexer to PVP, release all allocated resources, and then move to the WAIT_FOR_FAX_CALL state (a1).

15.4.2.4 Actions in the remodulating endpoint states

15.4.2.4.1 Actions in the WAIT_FOR-_DCS_FROM_FAR_END state (r.a2)

Each endpoint shall check whether traffic coded according to the modulation type and speed that have been provided by the corresponding V.21 demodulator can be remodulated with the available resource. This information is extracted either from the Digital Command Signal (DCS) of T.30 or from the information embedded in the Nonstandard Setup (NSS) from the V.21 demodulator of the packetized side. If compatibility has been verified, then the state machine shall:

a) Store the modem type and speed in the system variables TERM_SPEED and TERM_TYPE, respectively;

b) Put the Multiplexer in the FADCOMP position to route the information from the packetized side through FADCOMP. The Demultiplexer is put in the OFF position to prevent echo of the regenerated signal from returning to the channel-oriented side. The Call-State Layer shall send the ML_FAX_START-_RESPONSE(TERM_SPEED,TERM_TYPE) primitive to the Modulation Layer of its corresponding terminating endpoint.

The Call_State Layer shall set the system variable PART_PAGE_REQ_CNT to 0 and shall move to the WAIT_FOR_TCF_TO_END state (r.b1).

If the modulation type and speed as extracted from the T.30 message is not compatible with the PCME's resources, all resources are released and the state machine moves back to the WAIT_FOR_FAX_CALL state (a1).

If an NSC or DTC T.30 message is received from the packetized side, the machine changes its role to that of a demodulating endpoint and moves to the WAIT_FOR_DCS_FROM_NEAR_END state (t.a2).

15.4.2.4.2 Actions in the WAIT_FOR-_TCF_TO_END state (r.b1)

In this state, the Call-State Layer shall wait for the primitives from the corresponding Modulation Layer. If either V.21 demodulator indicates that V.21 is coming, the Call-State Layer shall put both the Multiplexer and the Demultiplexer in the PVP mode. It shall send the ML_FAX_STOP_RESPONSE to indicate to the Modulation Layer of the remodulating endpoint to return to the OFF state. It shall release all resources and return to the WAIT_FOR_FAX_CALL state (a1).

If the ML_FAX_STOP_INDICATION primitive arrives from the Modulation Layer of the terminating endpoint, the Call-State Layer shall put both the Multiplexer and the Demultiplexer in the PVP mode. It shall then move to the WAIT_FOR_CFR (r.b2) state.

If the ML_FAX_ABORT_INDICATION arrives from the Modulation Layer of the terminating endpoint, the Call-State Layer shall put both the Multiplexer and the Demultiplexer in the PVP mode. It shall release all resources and return to the WAIT_FOR_FAX_CALL state.

15.4.2.4.3 WAIT_FOR_CFR state (r.b2)

If either DCS or NSS signals arrive from the packetized side, the Call-State Layer shall put both the Multiplexer and the Demultiplexer in the PVP mode. It shall release all resources and return to the WAIT_FOR_FAX_CALL (a1) state.

If the Confirmation to Receive (CFR) arrives from the channel-oriented side, the state machine shall put the Multiplexer in the FADCOMP position to route the information from the packetized side through FADCOMP. The Demultiplexer shall be put in the OFF position to prevent echo of the regenerated signal from the channel-oriented side.

The Call-State Layer shall send the ML_FAX_START_RESPONSE(TERM_SPEED, TERM_TYPE) primitive to the Modulation Layer of its corresponding terminating endpoint.

The state machine shall then move to the WAIT_FOR_PAGE_END (r.c1) state.

If the V.21 demodulator indicates the arrival of any of the following messages from the channel-oriented side, the state machine shall return to the WAIT_FOR_DCS_FROM_FAR_END (r.a2) state: Command Repeat (CRP), Failure to Train (FTT), Nonstandard facilities command (NSC), or Digital Transmit Command (DTC).

15.4.2.4.4 WAIT_FOR_END_OF_PAGE state (r.c1)

In this state, the Call-State Layer shall wait for the primitives from the corresponding Modulation Layer. If either V.21 demodulator indicates that V.21 is coming, the Call-State Layer shall put both the Multiplexer and the Demultiplexer in the PVP mode. It shall send the ML_FAX_STOP_RESPONSE to indicate to the Modulation Layer of the terminating endpoint to return to the OFF state. It shall release all resources and return to the WAIT_FOR_FAX_CALL (a1) state.

If the ML_FAX_STOP_INDICATION primitive arrives from the Modulation Layer of the terminating endpoint, the Call-State Layer shall put both the Multiplexer and the Demultiplexer in the PVP mode. It shall then move to the WAIT_FOR_POST_PAGE_COMMAND (r.c2) state.

If the ML_FAX_ABORT_INDICATION arrives from the Modulation Layer of the terminating endpoint, the Call-State Layer shall put both the Multiplexer and the Demultiplexer in the PVP mode. It shall release all resources and return to the WAIT_FOR_FAX_CALL (a1) state.

15.4.2.4.5 WAIT_FOR_POST_PAGE_COMMAND state (r.c2)

In this state, the state machine is awaiting a V.21 message from the packetized side. This message will be called subsequently the "post-page command." Upon receipt of the post-page command, the state machine moves to the WAIT_FOR_POST_PAGE_RESPONSE state (r.d1).

15.4.2.4.6 WAIT_FOR_POST_PAGE_RESPONSE state (r.d1)

In this state, the state machine is awaiting specific messages from the channel-oriented side, which represent the response to the previously received post-page command. The combination of the command and response determines the actions.

15.4.2.4.6.1 End of procedure

The end of procedure is indicated by one of the following post-page commands: End of Procedure (EOP), Procedure Interrupt-EOP (PRI-EOP), Partial Page Signal-EOP (PPS-EOP), and PPS-PRI-EOP. If the post-page response is Message Confirmation (MCF), the state Machine releases all allocated resources and returns to the WAIT_FOR_FAX_CALL (a1) state.

15.4.2.4.6.2 End of message

The end of message is indicated by one of two combinations:

- one of the following commands: End of Message (EOM), Procedure Interrupt-EOM (PRI-EOM), Partial Page Signal-EOM (PPS-EOM), and PPS.PRI-EOM, along with the Message Confirmation (MCF) response;

- one of the following commands: End of Message (EOM) or Procedure Interrupt-EOM (PRI-EOM), along with one of the following

responses: Retrain Positive (RTP) or Retrain Negative (RTN).

In either case, the state machine moves to the WAIT_FOR_DCS_FROM_FAR_END state (r.a2).

15.4.2.4.6.3 Multiple page

The multiple page case is indicated by one of two combinations:

– one of the following commands: Multipage Signal (MPS), Procedure Interrupt-MPS (PRI-MPS), Partial Page Signal-MPS (PPS-MPS), PPS.PRI-MPS, PPS-Null, along with the Message Confirmation (MCF) response. In this case, the PART_PAGE_REQ_CNT variable is reset to 0;

– the response is a Partial Page Request (PPR) while the partial page received count PART_PAGE_REQ_CNT is less than 3. The post-page command is irrelevant. In this case, the PART_PAGE_REQ_CNT variable is incremented.

Additional actions are as follows:

a) put the Multiplexer in the FADCOMP position to route the information from the packetized side through FADCOMP;

b) put the Demultiplexer in the OFF position to prevent echo of the regenerated signal from the channel-oriented side;

c) send the ML_FAX_START_RESPONSE-(TERM_SPEED,TERM_TYPE) primitive to the Modulation Layer of its corresponding terminating endpoint;

d) move to WAIT_FOR_END_OF_PAGE state (r.c1).

15.4.2.4.6.4 Partial page request

This is the case when the response PPR is detected while the PART_PAGE_REQ_CNT has reached the limit of 3, regardless of the post-page command. The PART_PAGE_REQ_CNT variable is reset to 0 and the state changes to the WAIT_FOR_CTC_OR_EOR.X state (r.g1).

15.4.2.4.6.5 Procedure to retrain

This is the case when one of the following commands is detected: Multipage Signal (MPS), End of Procedure (EOP), Procedure Interrupt-MPS (PRI-MPS), Procedure Interrupt-EOP(PRI-EOP), along with either Retrain Negative (RTN) or Retrain Positive (RTP). In this case, the state changes to the WAIT_FOR_DCS_FROM_FAR_END state (r.a2).

15.4.2.4.6.6 Procedure interrupt

This is the case when the response is either Procedure Interrupt Negative (PIN) or Procedure Interrupt Positive (PIP), regardless of the post-page command. The state changes to WAIT_FOR_CMD state (r.e1).

15.4.2.4.6.7 WAIT_FOR_CMD state (r.e1)

If the V.21 demodulator indicates the arrival of the "non-final" Nonstandard Setup (NSS) from the packetized side, the state machine shall return to the WAIT_FOR_DCS_FROM_FAR_END state (r.a2).

If the V.21 demodulator indicates the arrival of either the Nonstandard facilities command (NSC) or the Digital Transmit Command (DTC) from the packetized side, the state machine shall switch the role of the node to a demodulating node and shall move to the WAIT_FOR_DCS_FROM_NEAR_END state (t.a2).

If the V.21 demodulator indicates the arrival of either the Nonstandard Setup (NSS) or the Digital Signal Command (DCS) from the packetized side, and the modulation type and speed as extracted from the V.21 information are compatible with the allocated resources, the state machine shall:

a) store the modem speed and type in the system variables TERM_SPEED and TERM_TYPE, respectively;

b) put the Multiplexer in the FADCOMP position to route the information from the packetized side through FADCOMP;

c) put the Demultiplexer in the OFF position to prevent echo of the regenerated signal from the channel-oriented side;

d) send the ML_FAX_START-_RESPONSE(TERM_SPEED,TERM_TYPE) primitive to the Modulation Layer of its corresponding terminating endpoint;

e) set PART_PAGE_REQ_CNT to 0;

f) move to the WAIT_FOR_TCF_TO_END state (r.b1).

If the V.21 demodulator indicates the arrival of either the Nonstandard Setup (NSS) or the

Digital Command Signal (DCS) from the packetized side, and the modulation type and speed extracted from the V.21 information are not compatible with the allocated modems, all resources are released and the state machine moves back to the WAIT_FOR_FAX_CALL state (a1).

If the V.21 demodulator indicates the arrival of EOR.PRI-Q or PRI-Q or PPS.PRI-Q from the packetized side, all resources are released and the state machine moves back to the WAIT_FOR_FAX_CALL state (a1). As defined in A7.1 of T.30, PRI-Q is a general term referring to either PRI.ROM, PRI.MPS, or PRI EOP post message command. These commands are used in the optional T 4 error correction mode PPS.PRI-Q could be either PPS PRI-EOM, PPS.PRI-MPS, or PPS.PRI-EOP post message command.

15.4.2.4.7 WAIT_FOR_CTC_OR_EOR.X state (r.g1)

If the V.21 demodulator indicates the arrival of the Continue to Correct (CTC) from the packetized side, the state machine shall set the speed to TERM_SPEED and the modulation type to TERM_TYPE as extracted by the V.21 demodulator, and shall move to the WAIT_FOR_CTR state (r.g2).

If the V.21 demodulator indicates the arrival of EOR.X signal (End of Retransmission = EOR) from the packetized side, where X is either Q or PRI-Q, it shall move to the WAIT_FOR_ERR (r.g3) state. As defined in A7.1 of T.30, EOR.Q indicates either EOR.EOM, EOR MPS, EOR.EOP, or EOR.Null post-message command. EOR.PRI-Q indicates either EOR PRI-EOM, EOR.PRI-MPS, or EOR.PRI-EOP X is referred to as "EOR type" subsequently.

15.4.2.4.8 WAIT_FOR_CTR state (r.g2)

If the V.21 demodulator indicates the arrival of the Response to Continue to Correct (CTR) from the channel-oriented side, and the modulation type and speed as extracted from the V.21 information are compatible with the allocated resources, the state machine shall:

a) store the modem speed and type in the system variables TERM_SPEED and TERM_TYPE, respectively;

b) put the Multiplexer in the FADCOMP position to route the information from the packetized side through FADCOMP;

c) put the Demultiplexer in the OFF position to prevent echo of the regenerated signal from the channel-oriented side;

d) send the ML_FAX_START-_RESPONSE(TERM_SPEED,TERM_TYPE) primitive to the Modulation Layer of its corresponding terminating endpoint;

e) set PART_PAGE_REQ_CNT to 0,

f) move to the WAIT_FOR_END_OF_PAGE state (r.c1)

If the V 21 demodulator indicates the arrival of the Response to Continue to Correct (CTR) from the channel-oriented side, and the modulation type and speed extracted from the V.21 information are not compatible with the allocated modems, all resources are released and the state machine moves back to the WAIT_FOR_FAX_CALL (a1) state.

15.4.2.4.9 WAIT_FOR_ERR state (r.g3)

If the V 21 demodulator indicates the arrival of a Response to End of Retransmission (ERR) from the channel-oriented side and the EOR type as set in state r g1 is either EOM or PRI-EOM, the state machine shall move to the WAIT_FOR_DCS_FROM_FAR_END state (r a2).

If the V.21 demodulator indicates the arrival of a Response to End of Retransmission (ERR) from the channel-oriented side, and the EOR_TYPE as set in state r g1 is either Null or MPS or PRI.MPS, the state machine shall:

a) set the Multiplexer to FADCOMP,

b) set the Demultiplexer to OFF;

c) send the primitive ML_FAX_START-_RESPONSE(TERM_SPEED,TERM_TYPE) to the Modulation Layer of the terminating endpoint,

d) move to the WAIT_FOR_END_OF_PAGE state (r.c1).

If the V 21 demodulator indicates the arrival of a Response to End of Retransmission (ERR) from the channel-oriented side, and the EOR type as set in state r.g1 is either EOP or PRI-EOP, all resources are released and the state machine

moves back to the WAIT_FOR_FAX_CALL state (a1)

If the V.21 demodulator indicates the arrival of a Procedure Interrupt Negative (PIN) from the channel-oriented side, the state machine shall return to the WAIT_FOR_CMD state (r.e1)

15.5 Facsimile originating (demodulating) endpoint procedures

15.5.1 Modulation layer procedures at the originating endpoint

This layer shall demodulate the 64-kbit/s facsimile page data and extract the baseband data It has the additional function of detecting echo protection tones and training sequences

The finite state machine, shown in figure 24, depicts the operation of the Modulation Layer.

State transitions occur after reception of REQUEST primitives from the Call-State timer expiration or upon arrival of specific signals from the channel-oriented side.

15.5.1.1 OFF state

This is the initial state of the Modulation Layer at the originating endpoint

While in this state, if a ML_FAX_CAPABILITY-_REQUEST(capabilities) primitive is received from the Call-State layer, the Modulation Layer shall send the PL_CAPABILITY-_REQUEST(capabilities) primitive to the Packet Layer

When the ML_FAX_START_REQUEST-(TYPE,SPEED) arrives from the Call-State Layer, the Modulation Layer shall place the TYPE and SPEED parameters into the ML_ORIG_TYPE and ML_ORIG_SPEED system variables. Then the state machine transitions to the IDLE state.

15.5.1.2 IDLE state

Upon entering the IDLE state, the Modulation Layer shall restart the WAIT_TIME timer. This timer measures the time that the originating end Modulation Layer shall wait for a signal (either an EPT tone or facsimile page modulation signal). The default value is 3 seconds. Other values are for further study.

The action of the Modulation Layer shall be as follows

a) If an EPT tone is detected, the Modulation Layer shall send the PL_FAX_SPURT-_HEADER_REQUEST(ACTION) primitive to the layer 3 entity, where ACTION is set to *1700 Hz EPT Start* or *1800 Hz EPT Start*, depending on the frequency of the tone. The Modulation Layer state machine shall stop the WAIT_TIME timer and then transition to the EPT state;

b) If a training sequence for modem type ML_ORIG_TYPE at speed ML_ORIG_SPEED is detected, the Modulation Layer sends the PL_FAX_SPURT_HEADER_REQUEST-(ACTION) primitive to the layer 3 entity where ACTION is set to "Generate a training sequence" The Modulation Layer state machine shall stop the WAIT_TIME timer and then transition to the TRAIN state,

c) If the timer WAIT_TIME expires, then the Modulation Layer shall send the ML_FAX-_ABORT_CONFIRMATION primitive to the Call-State Layer. It shall also send the PL_FAX_SPURT_HEADER_REQUEST (ACTION) primitive to the layer 3 entity, where ACTION is set to *Abort*. The Modulation Layer state machine shall then move to the OFF state;

d) If the ML_FAX_STOP_REQUEST primitive arrives from the Call_State Layer, the Modulation Layer shall move to the OFF state.

15.5.1.3 EPT state

Upon entering the EPT state, the Modulation Layer shall restart the EPT_TIME_ORIG timer. The default value for this timer is 500 ms

In the EPT state, the Modulation Layer shall await the end of the EPT tone in the voiceband signal When the end of EPT energy is declared, the Modulation Layer shall send the PL_FAX-_SPURT_HEADER_REQUEST(ACTION) to the layer 3 entity, where ACTION is set to *EPT Stop* The Modulation Layer shall then move back to the IDLE state.

If the timer EPT_TIME_ORIG expires, then the Modulation Layer shall send the PL_FAX-_SPURT_HEADER_REQUEST(ACTION) primitive to the layer 3 entity, where ACTION is set to *Abort* It shall send the ML_FAX_ABORT-_CONFIRMATION primitive to the Call-State layer

The Modulation Layer state machine shall then move to the OFF state. If the ML_FAX_STOP_-REQUEST primitive arrives from the Call_State Layer, the Modulation Layer shall move to the OFF state.

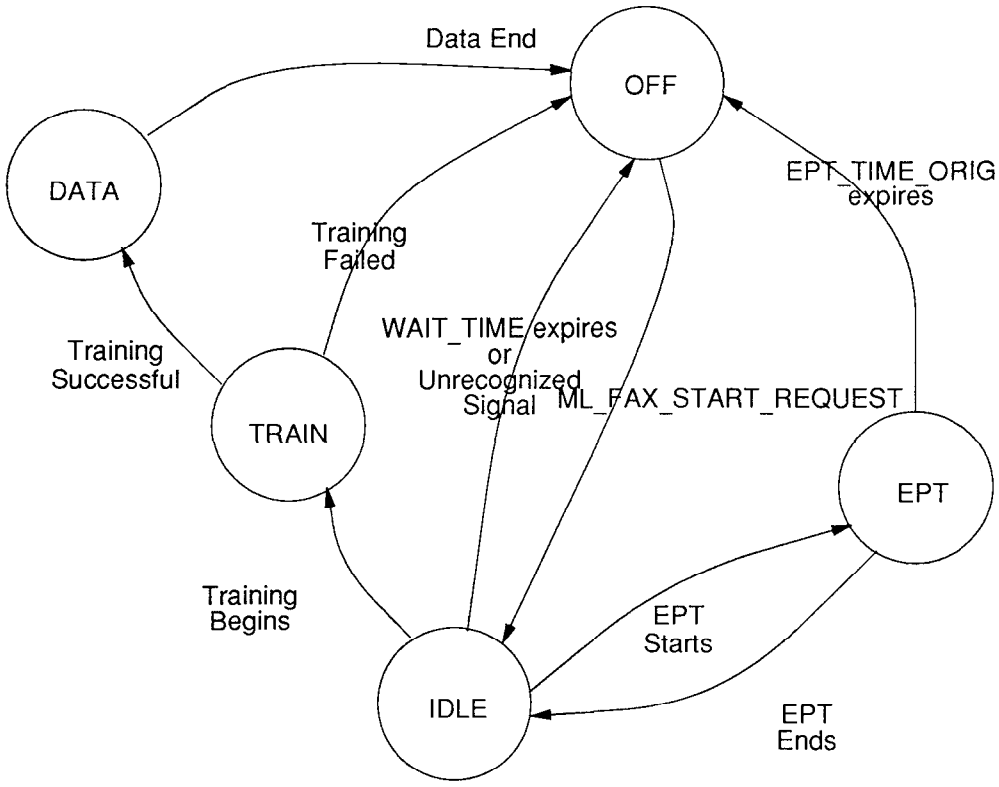


Figure 24 – Global States of the Originating End Modulation Layer Finite State Machine

NOTE – For simplicity, the transition from any state to the OFF state due to the ML_FAX_STOP_REQUEST primitive is not shown

15.5.1.4 TRAIN state

In the TRAIN state, the Modulation Layer shall synchronize to the incoming modulated signal and shall await the end of the training sequence.

If the training sequence completes and the Modulation Layer was able to train according to modem type ML_ORIG_TYPE and speed ML_ORIG_SPEED, the Modulation Layer shall send the PL_FAX_DATA_START_REQUEST primitive to the layer 3 entity, informing it that demodulated bits are about to arrive. The Modulation Layer state machine shall then move to the DATA state.

If the ML_FAX_STOP_REQUEST primitive arrives from the Call_State Layer, the Modulation Layer shall move to the OFF state.

If the Modulation Layer detects an error in the training sequence or it is unable to train, it shall send the PL_FAX_SPURT_HEADER-_REQUEST(ACTION) primitive to the layer 3 entity, where ACTION is set to *Abort*. It shall also send the ML_FAX_ABORT_CONFIRMATION primitive to the Call-State layer. The Modulation Layer state machine shall then move to the OFF state.

15.5.1.5 DATA state

In the DATA state, the Modulation Layer shall demodulate the voiceband signal according to ML_ORIG_TYPE and ML_ORIG_SPEED and unscramble the bits. The unscrambled bits are given to the Packet Layer grouped by symbols. For example, if ML_ORIG_TYPE is V.29, the demodulation shall be done according to CCITT Recommendation V.29, including the unscrambling operation of Appendix II/V.29. The output of this demodulator that is given to the Packet Layer shall consist of a series of unscrambled symbols containing unscrambled bits Q1 through Q4, for 9600 bit/s and Q2 through Q4 for 7200 bit/s (see 2.2.1 and 2.2.2 of CCITT Recommendation V.29), arranged as shown in figure 11, for the two speeds 9600 bit/s and 7200 bit/s. Similar actions are taken for V.27 ter.

When the carrier terminates, the Modulation Layer shall send the PL_FAX_DATA_STOP-_REQUEST primitive to the layer 3 entity and the ML_FAX_STOP_CONFIRMATION primitive to the Call-State Layer. The Modulation Layer state machine shall then move to the OFF state.

If the signal can no longer be demodulated, the Modulation Layer shall send the PL_FAX-_SPURT_HEADER_REQUEST(ACTION) primitive to the layer 3 entity, where ACTION is set to *Abort*. It shall send the ML_FAX_ABORT-_CONFIRMATION primitive to the Call_State Layer. It shall then move to the OFF state.

If the ML_FAX_STOP_REQUEST primitive arrives from the Call_State Layer, the Modulation Layer shall move to the OFF state.

15.5.2 Packet layer procedures at the originating endpoint

The function of this layer is to generate packets in response to REQUEST primitives.

15.5.2.1 Receipt of the PL_FAX_SPURT-_HEADER_REQUEST primitive

When the Modulation Layer informs the layer 3 entity to send a spurt header packet using the PL_FAX_SPURT_HEADER_REQUEST(ACTION) primitive, the layer 3 entity shall start a timer TVDELAY associated with that packet and shall format a facsimile Spurt Header frame (see figure 9) with the *ACTION* field set according to the parameters. Then, the remaining procedures, described in 15.5.2.5, are followed.

15.5.2.2 Receipt of the PL_FAX_CAPABILITY-_REQUEST primitive

When the Modulation Layer informs the layer 3 entity to send a capability packet using the PL-_FAX_CAPABILITY_REQUEST (capabilities) primitive, the layer 3 entity shall start a timer TVDELAY associated with that packet and shall format a facsimile Capability Indicator frame (see figure 8) with the V.27, V.29, V.33, and V.17 fields set according to the capabilities parameter. Then, the remaining procedures, described in 15.5.2.5, are followed.

15.5.2.3 Receipt of the PL_FAX_DATA-_START_REQUEST primitive

When the Modulation Layer informs the layer 3 entity to start packetizing data using the PL_FAX_DATA_START_REQUEST primitive, the layer 3 entity shall begin segmenting and buffering packets using the symbols supplied by the Modulation Layer. Each packet is of the format given in figure 10, which shows the Facsimile Page Information Frame format. The M-bit shall

be set to 1. For the first packet, the SEQ Field shall be set to 1; for subsequent packets, SEQ shall be incremented to 15 with a rollover to 1.

The symbols from the Modulation Layer shall be placed into the packet according to figure 11. The information field of the packet shall contain the number of symbols arriving in 20 ms, nominally 48, 32, and 24 for V.29 (9.6 and 7.2 kbit/s), 4.8 kbit/s V.27 ter, and 2.4 kbit/s V.27 ter, respectively. These correspond to 24, 16, and 12 octets of information most of the time, although, due to the lack of synchronization between the facsimile machine and the PCME, the information field can occasionally be one octet longer. The last packet of a page may contain fewer symbols since the number of symbols in a page does not need to be a factor of the number received in 20 ms. The last octet of a packet may contain less than two symbols, in which case the BILO field shall be set to indicate the number of bits that are not to be considered. For example, for V.29 and V.27 ter, the BILO field is set to either 0 or 4, depending on whether the number of symbols in the packet is even or odd, respectively. In the latter case, the value of the four most significant bits in the last octet of the information field shall not be considered.

After each packet is formed, the packet layer shall start a timer TVDELAY that is associated with that packet. Then, the remaining procedures, noted in 15.5.2.5, shall be followed.

The packetization process shall be continued until the PL_FAX_DATA_STOP_REQUEST primitive is received.

15.5.2.4 Receipt of the PL_FAX_DATA_STOP_REQUEST primitive

When the Modulation Layer informs the layer 3 entity to stop packetizing data using the PL_FAX_DATA_STOP_REQUEST primitive, the layer 3 entity shall finish the packet it is in the process of forming until all the bits are used up. The remaining procedures are identical to those for the previous facsimile page information packets, except that the M-bit is set to 0.

15.5.2.5 Time stamping procedures

Packets shall be buffered on a first-in/first-out (FIFO) basis. Upon receipt of the DL_L1_READY_INDICATION primitive from the Link Layer, the layer 3 entity shall stop the TVDELAY timer associated with the packet at the head of the FIFO queue and shall copy its value (in ms) to the TIME STAMP field. The value of the TIME STAMP field is clamped at 200. The packet shall be delivered to the Link Layer via either the DL_UNIT_H_DATA_REQUEST or DL_UNIT_DATA_REQUEST primitives.

15.5.3 Link layer procedures at the originating endpoint

The Link Layer procedures are the same as those defined in 5.2.1 and 5.2.2 of ANSI T1.312.

15.6 Intermediate node procedures

Same as for DICE (See 10.5).

15.7 Facsimile procedures at the terminating (remodulating) endpoint

15.7.1 Link layer procedures at the terminating endpoint

The Link Layer procedures are the same as specified in 5.2.3 and 5.2.4 of ANSI T1.312.

15.7.2 Packet layer procedures at the terminating endpoint

The function of the packet layer is to reassemble the baseband facsimile image data and call control information from the received packets.

15.7.2.1 Common procedures

Upon receipt of the DL_UNIT_H_DATA_INDICATION or DL_UNIT_DATA_INDICATION from the Link Layer, the layer 3 entity shall examine the values encoded in the PD, SC, and DMC fields.

If the PD value matches that for PVP, the layer 3 entity shall proceed according to the value of the SC field; otherwise the packet is dropped.

If the value of the SC field is 00, the value for voice and voiceband data, the layer 3 entity shall follow the procedures in clause 6 of ANSI T1.312.

If the value of the SC field is 11, the value of digital data, the layer 3 entity shall follow the procedures in either 10.6.2 for DICE or in 11.6.2 for VDLC.

If the value of the SC field is 01, the value for the digital modem class, and the DMC value is the value for facsimile, the layer 3 entity shall proceed as below. Actions for other values of

DMC are for further study

Procedures of the case of SC = 10 are left for further study.

15.7.2.2 Build-out delay

The build-out delay procedures are the same as for ANSI T1.312 (6.3.3.2), except that at playout time, the action depends on the packet type as indicated by the Type field. Specific actions are described in the following

The build-out delay system variable has the same definition as for PVP, but its value must be chosen so as to preserve the gap width between the facsimile call components. The build-out delay for facsimile depends on:

– the build-out delay for the voice path;

– the processing time (including ADPCM encoding and decoding) at both the originating and terminating endpoints for the V.21 call control signals;

– the processing time (including signal detection and demodulation times) at both the originating and terminating endpoints for the training sequence spurt header packet

Thus, there are separate build-out delay parameters, one for the voice path and the second for the facsimile demodulation path. Both these parameters indicate the end-to-end delay from packetization to depacketization, but do *not* include signal processing times (item 2 above for voice path, and item 3 for facsimile demodulation) or transmission propagation time (a function of the transmission facilities) The total end-to-end delay includes build-out, processing, and transmission propagation times. To preserve gap widths, the total end-to-end must be the same for both paths, i.e.,

voice_build-out + voice_processing_time =
fax_build-out + facsimile_processing_time

The propagation time is not included because it is the same on both sides of the equation The processing times are implementation dependent Either the voice build-out delay or the facsimile demodulation build-out delay shall be set by service administration and the other build-out delay may be derived as explained above

15.7.2.3 Actions as determined by type field

At playout time, as determined in the previous subclause, one of the following actions are taken

15.7.2.3.1 Spurt header packet (Type=00)

Type=00 indicates a spurt header packet. The layer 3 entity shall send the PL_FAX_SPURT-_HEADER_INDICATION(ACTION) to the Modulation Layer, where ACTION is set to the value encoded in the ACTION field of the arriving packet

15.7.2.3.2 Page information packet (Type=01)

Type=01 indicates a page information packet.

a) If the SEQ field > 0 and the M-bit is set to 1, then the layer 3 entity shall inform the Modulation Layer that a page information packet has arrived via the PL_FAX_DATA-_INDICATION primitive. It shall remove the symbols from the facsimile page information field, as per figure 11, in sequential order, and shall buffer them. The last octet of the information field shall be treated according to the BILO field The contents of the buffer shall be sent to the Modulation Layer a symbol at a time as needed by the Modulation Layer

b) If the SEQ field is > 0 and the value the M-bit is 0, the layer 3 entity shall remove the symbols in the facsimile page information field and shall add them to the buffer, the contents of which are sent to the Modulation Layer When the buffer is emptied, the layer 3 entity shall send the PL_FAX_DATA_STOP-_INDICATION primitive to the Modulation Layer

c) If SEQ = 0, the corresponding build-out procedures of ANSI T1 312 6 3 3 2 shall be followed

15.7.2.3.3 Capability indication packet (Type=10)

Type=10 indicates a capability indication packet The layer 3 entity shall send the PL_FAX-_CAPABILITY_INDICATION (capabilities) to the Modulation Layer, where capabilities are set according to the V.27, V 29, V.33 and V.17 fields of the arriving packet

15.7.3 Modulation layer procedures at the terminating endpoint

This layer shall reconstruct the original carrier and EPT signals for the facsimile page data component Figure 25 contains a state diagram of this layer

For all states, if the Modulation Layer receives a ML_FAX_STOP_RESPONSE primitive from the Call-State Layer, it shall go to the OFF state and stop any running timers.

15.7.3.1 OFF state

This is the initial state of the Modulation Layer.

While in this state, if the Modulation Layer receives a PL_FAX_CAPABILITIES-_INDICATION (capabilities) primitive from the Packet Layer, it sends a ML_FAX-_CAPABILITIES_INDICATION (capabilities) primitive to the Call-State Layer.

When the Modulation Layer receives the ML-_FAX_START_RESPONSE(TYPE,SPEED) primitive from the Call-State Layer, the Modulation Layer shall place the TYPE and SPEED parameters into the ML_TERM_TYPE and ML_TERM_SPEED system variables. It then moves to the IDLE state.

15.7.3.2 IDLE state

As soon as it enters this state, the Modulation Layer shall start the IDLE_TIME timer. The default value for this timer is 3 seconds. Other values are for further study.

In this state, the Modulation Layer shall wait for the layer 3 entity to indicate receipt of a facsimile spurt header packet, at which time the Modulation Layer shall stop the IDLE_TIME timer.

If the Modulation Layer receives the PL_FAX-_SPURT_HEADER_INDICATION(ACTION) primitive from the layer 3 entity then:

a) If ACTION is set to 1700 Hz EPT Start or 1800 Hz EPT Start, then the Modulation Layer shall take the following actions:

1) set system variable ML_TERM_EPT-_FREQ to indicate 1700 Hz or 1800 Hz, depending on the ACTION parameter;

2) move to the EPT state.

b) If ACTION is set to Training Sequence, then the Modulation Layer shall move to the TRAIN state;

c) If ACTION is set to *Abort*, the Modulation Layer shall:

1) send the ML_FAX_ABORT-_INDICATION primitive to the Call-State Layer;

2) move to the OFF state.

If the IDLE_TIME timer expires, the Modulation Layer shall send the ML_FAX_ABORT-_INDICATION primitive to the Call-State Layer, and then return to the OFF state.

15.7.3.3 EPT state

Upon entering the EPT state, the Modulation Layer shall:

a) start the timer EPT_TIME_TERM. This timer measures the time that the terminating end Modulation Layer remains in the EPT state generating an EPT tone. The default value of the timer is 500 ms;

b) begin generating the PCM samples that correspond to an echo protection tone of a frequency corresponding to the ML_TERM-_EPT_FREQ system variable. The tone signal power shall meet the specifications of CCITT Recommendation V.2.

If the Modulation Layer receives the PL_FAX-_SPURT_HEADER_INDICATION(ACTION) primitive from the layer 3 entity with ACTION set to *EPT Stop*, the Modulation Layer shall terminate the generation of the EPT tone, shall stop the EPT_TIME_TERM timer, and shall move to the IDLE state.

If the Modulation Layer receives the PL_FAX-_SPURT_HEADER_INDICATION(ACTION) primitive from the layer 3 entity with ACTION set to *Abort*, the Modulation Layer shall:

a) terminate the generation of the EPT tone and stop the EPT_TIME_TERM timer;

b) send the ML_FAX_ABORT_INDICATION primitive to the Call-State Layer;

c) move to the OFF state.

After expiration of the timer EPT_TIME_TERM, the Modulation Layer shall:

a) terminate the generation of the EPT tone;

b) send the ML_FAX_ABORT_INDICATION primitive to the Call-State Layer;

c) move to the OFF state.

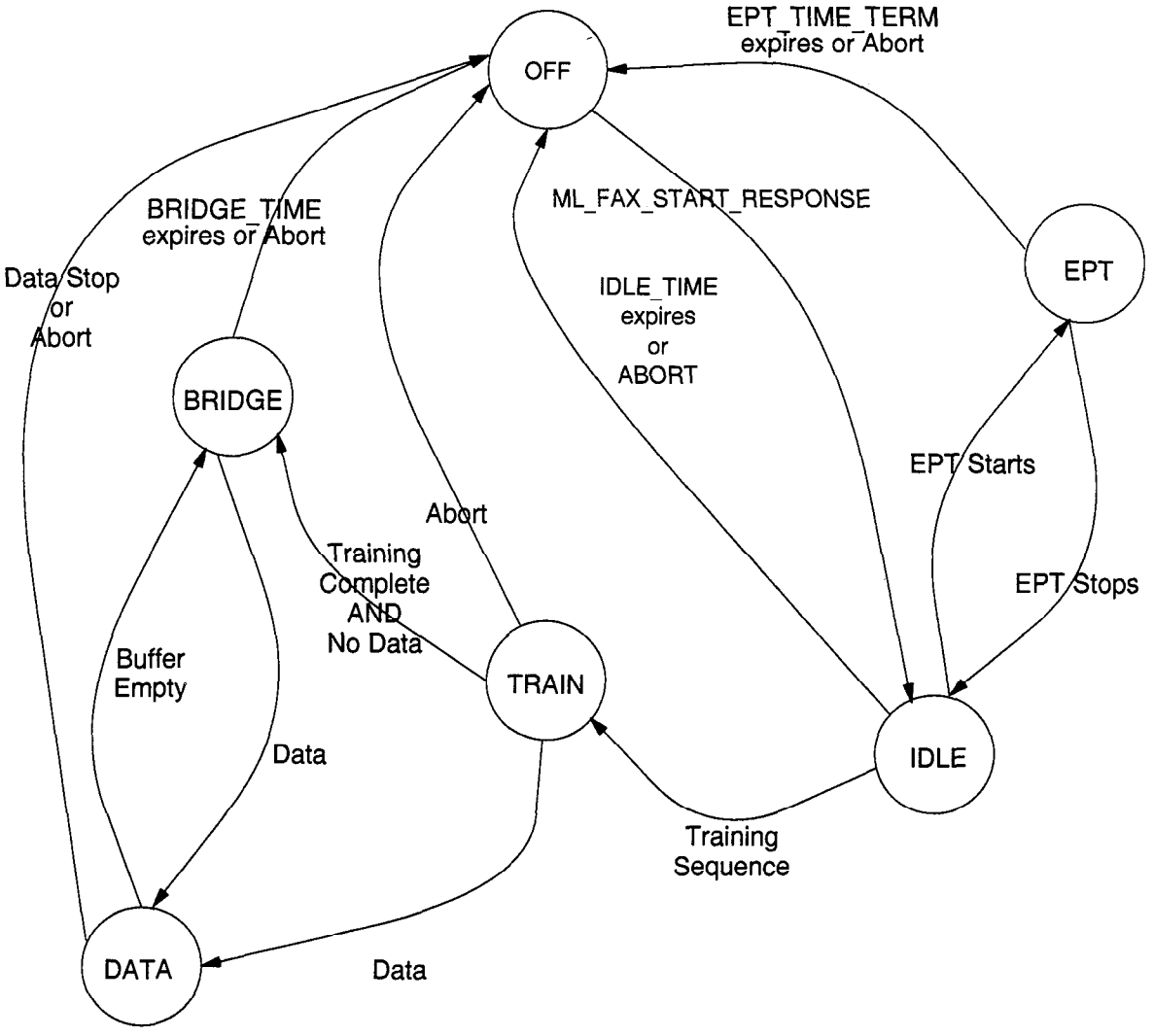


Figure 25 – Global states of the terminating end modulation layer terminating end finite state machine

NOTE – For simplicity, the transition from all states to the OFF state due to reception of the ML_FAX_STOP_RESPONSE primitive or of some spurt header packets is not shown.

15.7.3.4 TRAIN state

Upon entering the TRAIN state, the Modulation Layer shall begin generating the PCM samples that correspond to a training sequence specified for modem type ML_TERM_TYPE and speed ML_TERM_SPEED by the corresponding CCITT Recommendations.

While the training sequence is being generated, if the Modulation Layer receives the PL_FAX-_SPURT_HEADER_INDICATION(ACTION) primitive from the layer 3 entity, the Modulation Layer shall:

a) terminate the generation of the training sequence;

b) send the ML_FAX_ABORT_INDICATION primitive to the Call-State Layer;

c) move to the OFF state.

While the training sequence is being generated, if the Modulation Layer receives the PL_FAX-_DATA_INDICATION primitive, the Modulation Layer shall:

a) complete the generation of the full training sequence;

b) move to the DATA state

If the regenerated training sequence ends before the layer receives the PL_FAX_DATA-_INDICATION primitive, the Modulation Layer shall move to the BRIDGE state.

15.7.3.5 DATA state

In the DATA state, the Modulation Layer shall continue generating PCM samples representing a modulated signal of modem type ML_TERM_TYPE and modem speed ML_TERM_SPEED system variables

Upon entering the DATA state, the Modulation Layer shall obtain a symbol from the layer 3 entity and modulate it in the earliest available signal time to preserve the continuity with the training sequence.

In the case of V 29 at 9600 bits/second, the Modulation Layer shall scramble and modulate the unscrambled Q1 through Q4 bits of a symbol (see figure 11) as per CCITT Recommendation V.29 For V.29 at 7200 bits/second, the Modulation Layer shall scramble and modulate the unscrambled Q2 through Q4 bits of a symbol, as per CCITT Recommendation V.29 Note that at this speed, Q1 is 0 by definition. For V 27 ter at 4800 or 2400 bits/second, the Modulation Layer shall scramble and modulate the unscrambled T1 through T3 and D1 through D2 bits, respectively, from each symbol, according to CCITT Recommendation V.27 ter.

If the PL_FAX_DATA_STOP_INDICATION primitive arrives from the layer 3 entity, the following actions are taken:

a) modulate any remaining symbols;

b) terminate the generation of the carrier;

c) send the ML_FAX_STOP_INDICATION primitive to the Call-State Layer;

d) move to the OFF state.

If the Modulation Layer receives the PL_FAX-_SPURT_HEADER_INDICATION(ACTION) primitive from the layer 3 entity, the Modulation Layer shall:

a) terminate generation of the carrier;

b) send the ML_FAX_ABORT_INDICATION primitive to the Call-State Layer,

c) move to the OFF state

If all the available symbols have been demodulated (i.e , the buffer is empty) and the PL_FAX-_DATA_STOP_INDICATION has not arrived, the Modulation Layer shall move to the BRIDGE state

15.7.3.6 BRIDGE state

Upon entering the BRIDGE state, timer BRIDGE_TIME shall be started This timer measures the time that the terminating end Modulation Layer remains in the BRIDGE state, keeping the modem carrier alive while recovering from lost facsimile page information packets. The default value is 45 ms.

While in the BRIDGE state, the Modulation Layer shall continue to generate a modulation signal of modem type ML_TERM_TYPE and modem speed ML_TERM_SPEED system variables while awaiting data to arrive The pattern of symbols to be modulated shall correspond to the encoding of the all zeros symbol.

If the PL_FAX_DATA_INDICATION primitive arrives from the layer 3 entity, then the Modulation Layer state shall move to the DATA state

If the Modulation Layer receives the PL_FAX-_SPURT_HEADER_INDICATION(ACTION) primitive from the layer 3 entity with ACTION set to *Abort*, the Modulation Layer shall:

a) terminate generation of the carrier;

b) send the ML_FAX_ABORT_INDICATION primitive to the Call-State Layer;

c) move to the OFF state.

If timer BRIDGE_TIME expires, the following actions shall be taken:

a) terminate generation of the carrier;

b) send the ML_FAX_ABORT_INDICATION to the Call-State Layer;

c) move to the OFF state.

The value of the BRIDGE_TIME timer should be set as large as possible, taking into account the uncertainty due to time stamping. Its value should not violate the T.30 constraint that the minimum gap length between two components of a facsimile call is 55 ms. The default value is 45 ms. Other values are for further study.

15.8 System variables and protocol parameters

15.8.1 Variables

The variables listed below are used by the FADCOMP protocol. They are in addition to those defined in ANSI T1.312.

15.8.1.1 ML_ORIG_SPEED

This is the modem speed (e.g., 9600 bits/second) that was determined by the originating end Call-State Layer and is used by the originating end Modulation Layer.

15.8.1.2 ML_ORIG_TYPE

This is the modem type (e.g., V.29) that was determined by the originating end Call-State Layer and is used by the originating end Modulation Layer.

15.8.1.3 ML_TERM_EPT_FREQ

This is the frequency (1700 Hz, 1800 Hz) of the EPT that the Modulation Layer generates.

15.8.1.4 ML_TERM_SPEED

This is the modem speed (e.g., 9600 bits/second) as determined by the terminating end Call-State Layer and used by the terminating end Modulation Layer.

15.8.1.5 ML_TERM_TYPE

This is the Modem Type (e.g., V.29) as determined by the terminating end Call-State Layer and used by the terminating end Modulation Layer.

15.8.1.6 ORIG_SPEED

This is the speed of the modulation as extracted from T.30 messages by the Call-State Layer in its demodulator role.

15.8.1.7 ORIG_TYPE

This is the type of the modulation as extracted from T.30 messages by the Call-State Layer in its demodulator role.

15.8.1.8 TERM_SPEED

This is the speed of the modulation as extracted from the T.30 messages by the Call-State Layer in its remodulator role.

15.8.1.9 TERM_TYPE

This is the type of the modulation as extracted from the T.30 messages by the Call-State Layer in its remodulator role.

15.8.2 Timers

The following timers are used by the FADCOMP protocol in addition to those used in ANSI T1.312.

15.8.2.1 BRIDGE_TIME

This timer measures the time that the terminating end Modulation Layer remains in the BRIDGE state, keeping the modem carrier alive while recovering from lost facsimile page information packets. The default value is 45 ms.

15.8.2.2 EPT_TIME_ORIG

This timer measures the time that the originating end Modulation Layer remains in the EPT state. The default value is 500 ms.

15.8.2.3 EPT_TIME_TERM

This timer measures the time that the terminating end Modulation Layer remains in the EPT state generating an EPT tone. The default value is 500 ms.

15.8.2.4 IDLE_TIME

This timer measures the time that the terminating end Modulation Layer can remain in the IDLE state. The default value is 3 seconds.

15.8.2.5 TVDELAY

This timer is used to measure the variable queuing delay in a node that a packet encounters. It is used to update the TS field of a facsimile packet.

15.8.2.6 WAIT_TIME

This timer measures the time that the originating end Modulation Layer shall wait for a signal (either an EPT tone or facsimile page modulation signal) while its in the idle state. The default value is 3 seconds.

15.9 Summary of primitives

15.9.1 Primitives for the interfaces between link and packet layers

The primitives for the interfaces between the Link and Packet Layers have the same definition in clause 10 of ANSI T1.312 and in 9 for DICE.

15.9.1.1 DL_L1_READY_INDICATION

See 10.1.1 of ANSI T1.312.

15.9.1.2 DL_UNIT_DATA_REQUEST

See 10.1.2 of ANSI T1.312.

15.9.1.3 DL_UNIT_DATA_INDICATION

See 10.1.3 of ANSI T1.312.

15.9.1.4 DL_UNIT_H_DATA_REQUEST

See 10.1.4 of ANSI T1.312.

15.9.1.5 DL_UNIT_H_DATA_INDICATION

See 10.1.5 of ANSI T1.312.

15.9.2 Primitives for the interface between packet and modulation layers

15.9.2.1 PL_FAX_CAPABILITY_INDICATION-(CAPABILITIES)

This primitive is used by the terminating endpoint Packet Layer of the endpoint whose Call-State Layer role is that of a demodulator to indicate to the Modulation Layer that a capability indication packet has arrived for the Modulation Layer to transmit up to the Call-State Layer.

15.9.2.2 PL_FAX_CAPABILITY_REQUEST-(CAPABILITIES)

This primitive is used by the Modulation Layer of the originating endpoint whose Call-State Layer role is that of a remodulator to indicate to the Layer 3 entity that a capability indication packet is to be formed.

15.9.2.3 PL_FAX_DATA_INDICATION

This primitive is used by the Packet Layer of the terminating endpoint to indicate to the Modulation Layer that a page information packet that is not the last packet of a page spurt has arrived.

15.9.2.4 PL_FAX_DATA_START_REQUEST

This primitive is used by the Modulation Layer of the originating endpoint to indicate to the layer 3 entity that demodulated symbols are to be segmented into packets.

15.9.2.5 PL_FAX_DATA_STOP_INDICATION

This primitive is used by the layer 3 entity of the terminating endpoint to indicate to the Modulation Layer that the last page information packet of a page spurt has arrived and all its symbols have been read out of the buffer by the Modulation Layer.

15.9.2.6 PL_FAX_DATA_STOP_REQUEST

This primitive is used by the Modulation Layer of the originating endpoint to indicate to the layer 3 entity that there are no more demodulated symbols to segment into packets.

15.9.2.7 PL_FAX_SPURT_HEADER_INDICATION-(ACTION)

This primitive is used by the layer 3 entity of the terminating endpoint to indicate to the Modulation Layer an action to be executed regarding the

EPT tone or the training sequence.

15.9.2.8 PL_FAX_SPURT_HEADER_REQUEST-(ACTION)

This primitive is used by the Modulation Layer of the originating endpoint to indicate to the layer 3 entity that a packet indicating the given action is to be formed.

15.9.3 Primitives for the interface between the modulation and call-state layers

15.9.3.1 ML_FAX_ABORT_CONFIRMATION

This primitive is used by the Modulation Layer of the originating endpoint to confirm to the Call-State Layer that the facsimile page spurt has been aborted.

15.9.3.2 ML_FAX_ABORT_INDICATION

This primitive is used by the Modulation Layer of the terminating endpoint to inform the Call-State Layer that the facsimile remodulation has been aborted.

15.9.3.3 ML_FAX_CAPABILITY_INDICATION-(CAPABILITIES)

This primitive is used by the Modulation Layer of the terminating endpoint whose Call-State Layer is in the demodulator role to indicate to the Call-State Layer the remodulation capabilities of the far-end PCME.

15.9.3.4 ML_FAX_CAPABILITY_REQUEST-(CAPABILITIES)

This primitive is used by the Call-State Layer, when it is in the remodulator role, to indicate to the originating endpoint of the Modulation Layer the originating endpoint's remodulation capabilities.

15.9.3.5 ML_FAX_START_REQUEST-(TYPE,SPEED)

This primitive is used by the Call-State Layer of the originating endpoint to indicate to the Modulation Layer that a facsimile page spurt should be demodulated, using the given modem TYPE and SPEED.

15.9.3.6 ML_FAX_START_RESPONSE-(TYPE,SPEED)

This primitive is used by the Call-State Layer of the terminating endpoint to indicate to the Modulation Layer that a facsimile page spurt should be remodulated, using the given modem TYPE and SPEED.

15.9.3.7 ML_FAX_STOP_CONFIRMATION

This primitive is used by the Modulation Layer of the originating endpoint to confirm to the Call-State Layer that the facsimile page spurt has ended.

15.9.3.8 ML_FAX_STOP_INDICATION

This primitive is used by the Modulation Layer of the terminating endpoint to indicate to the Call-State Layer that the facsimile page spurt has ended.

15.9.3.9 ML_FAX_STOP_REQUEST

This primitive is used by the Call-State Layer of the originating endpoint to indicate to the Modulation Layer that the facsimile page spurt has been interrupted and that the Modulation Layer should return to the OFF state.

15.9.3.10 ML_FAX_STOP_RESPONSE

This primitive is used by the Call-State Layer of the terminating endpoint to indicate to the Modulation Layer that the facsimile page spurt has been interrupted and that the Modulation Layer should return to the OFF state.

16 Testing of the link

16.1 Overview

The test procedures consists of a compelled exchange of XID frames to allow the verification of the data link layer connection.

There are two types of test procedures:

- end-to-end tests where only edge nodes of the network shall process the frame;
- sectionalization tests, where each node of the path shall process the frame.

When the XID command frame with the test packet in its information field arrives at a node that must respond, the node shall respond with the indicated XID response frame. The sending node shall receive the XID response within a certain time limit T_{test} from sending the XID command. If the response is not received before T_{test} expires, the test is declared to be a failure.

Failure of the test indicates the occurrence of one or more of the following conditions·

– link not established (physically and/or logically);

– errors in processing;

– lost command or response due to network congestion.

The procedures allow for testing of all circuits irrespective of the type of traffic that they carry (voice, voiceband data, facsimile, video, digital data). This provides the capability of locating link failures in a wideband packet network and/or of ensuring the integrity of a permanent virtual circuit path before actual traffic is allowed to flow

The procedures allow for the testing for both two-way and one-way circuits In each case, the XID frame that contains the test packet is sent with the same address used for the user's traffic. For a two-way circuit, the response XID frame returns on the user's address. For a one-way circuit, the response XID frame flows on the management channel, the address of which is 8191 (the thirteen bits of the address are all set to 1).

16.2 Frame description

Figure 26 illustrates the format of the XID frame used for link testing. The following subsections describe the functional fields for the XID command for testing of two-way and one-way permanent virtual circuits (PVCs).

16.2.1 Address octets

Octets 1 and 2 represent the address field for a default two octet address. The first octet includes the address extension bit, the command/response (C/R) bit and the 6-bit upper subfield of the address The second octet includes the 7-bit lower subfield of the address (including the two address bits taken for congestion control, the discard eligibility flag bit) and the address extension bit. The C/R bit is set to 0 for a command and to 1 for a response. For all types of PVCs, the command is sent using the same data link connection identifier (DLCI) used for the user's traffic. The response is sent on the user's DLCI for two-way PVCs and on DLCI = 8191 for all other PVCs (a one-way PVC, broadcast PVCs, concentrated PVCs)

16.2.2 Control field

Octet 3 contains the control field for the XID frame.

16.2.3 Format identifier field

Octet 4 contains the Format Identifier field The Format Identifier field is a fixed length of one octet. Its value is 131 decimal.

16.2.4 Group identifier field

Octet 5 contains the Group Identifier field. This field identifies the function of the XID information field. The value of this field is selected to distinguish the various uses of the XID frame with the same value for Format Identifier, decimal 131. The Group Identifier field is selected to be 253 for end-to-end testing to indicate the frame is to be processed by the edge nodes only The value is set to 254 for a sectionalization test, i e., that all nodes in the path must process the frame.

16.2.5 Group length field

Octets 6 and 7 contain the group length field. This 16-bit field codes the "length" in octets of the remainder of this message, excluding FCS and closing flag. The length is coded in decimal format

16.2.6 DLCI value field

Octets 8 and 9 contain the values of the DLCI, including the bits for congestion control and discard eligibility. This field is useful when XID responses for non two-way permanent virtual circuits are sent on the management DLCI of 8191.

16.2.7 Responding node address

The responding node address is a field of up to 36 octets starting from octet 10 that contains an ASCII address for the node

16.2.8 FCS field

The last two octets of the message are for the FCS field.

16.3 Procedures

The test procedures are initiated by a command entered by the network personnel. Upon receipt of this command, the PCME shall send an XID command frame with a test packet as specified above It shall start a timer T_{test}, the value of which can be one of the following 3, 6, 10, 20, 30, 40, 60, 90, and 120 seconds.

Octet	Bits								Field Name
	8	7	6	5	4	3	2	1	
1							C/R	0	Address Octet 1
2								1	Address Octet 2
3	1	0	1	0	1	1	1	1	XID Control Field
4	1	0	0	0	0	0	1	1	Format Identifier (131)
5	1	1	1	1	1	1	0	1	Group Identifier (253)
	1	1	1	1	1	1	1	0	Group Identifier (254)
6	X	X	X	X	X	X	X	X	Group Length Octet 1
7	X	X	X	X	X	X	X	X	Group Length Octet 2
8									DLCI Value Octet 1
9									DLCI Value Octet 2
10	..	..	..	..	..	..	..		Responding Node Address (36 Octets)
	..	..	..	..	..	..	..	..	
	..	..	..	..	..	..	..	..	
46									FCS Octet 1
47									FCS Octet 2

Figure 26 – XID Command/response frame for testing permanent virtual circuits X = 0 or 1

Depending on the value of the GI field, the XID command frame with the test packet shall be processed either by each node in the network or by the edge nodes only. For end-to-end testing (GI=253), the terminating endpoint (the edge node) shall loop the XID command frame with the test packet and send it back to the originating endpoint. For sectionalization testing (GI=254), each intermediate node, as well as the terminating endpoint, shall loop back the XID frame as well.

The responses are displayed to the user.

NOTE – In the case of a voice connection with channel associated signaling, the XID frame tests the logical connection for voice only. Testing of the associated signaling logical connection is obtained through the procedures of ANSI T1.312, in particular through the timer TSIG_KA (7.4 of ANSI T1.312).

The value of the timer TSIG_KA ranges from 1.5 seconds to 90 seconds, depending on the value of the timer TSIG_REF, as given in table 12 (see 9.2 and 9.3 of ANSI T1.312):

Table 12 – Values of TSIG_KA

TSIG_REF	TSIG_KA
1	1.5 2.5 3.5 4.5
5	7.5 12.5 17.5 22.5
10	15 25 35 45
20	30 50 70 90

Therefore, T_{test} shall be at least equal to the value of TSIG_KA. For the default value of TSIG_KA = 25 seconds, T_{test} shall be at least 30 seconds.

In an end-to-end test, if the timer T_{test} expires before an XID response frame is received, the test is declared to be a failure.

In a sectionalization test, if no response arrives before timer T_{test} expires, the test is declared to be a failure. The responses from the responding nodes are displayed to the user. If all nodes in the path respond, and the responses arrive before timer T_{test} expires, the test is declared to be a success.

17 Permanent virtual circuit restoration

17.1 Overview

The purpose is to provide the PCME with a mechanism for PVC restoration after detection of a fault in transmission paths associated with a network node. The PVC restoration procedures involve a backup after the failure and a switchback after the fault is repaired. The XID response frame is used with the format specified below and is sent in the backward path for each direction of the connection. Figure 27 shows how this is done for a two-way connection. In a one-way connection, only one XID frame will be launched.

Two approaches are considered: end-to-end notification (method 1) and consolidated PVC restoration notification (method 2).

17.1.1 End-to-end notification (Method 1)

This method is restricted to two-way connections. In this approach, a PVC restoration/switchback request is generated for each failed PVC that has a backup PVC after a facility alarm has occurred or has been cleared. This individual fault indication message shall be generated by the node that terminates the failed facility. The messages are transmitted towards the endpoints, and each message includes the identity of each virtual circuit on the associated network facilities affected by the failure. At each node, the frames are relayed through the network and their DLCI is translated to the appropriate address toward the next node. At each endpoint node, the virtual circuits are switched to alternate virtual circuits on one or more different physical facilities.

17.1.2 Consolidated PVC restoration notification (Method 2)

A consolidated PVC restoration notification is a PVC restoration/switchback message that carries information related to multiple DLCIs up to the limit of the LAPD information field length of 260 octets. In the case where many DLCIs are affected, more than one message may be generated, although the total number of generated frames is much less than in the first case. Furthermore, this approach is more general since it can be used for all types of PVCs while the first method is restricted to two-way PVCs. In this approach, every intermediate node is required to terminate the message and do further processing.

17.2 Frame description

Figures 28 and 29 illustrate the format of the XID frame for the PVC backup/switchback procedure for methods 1 and 2, respectively. The following subsections describe the functional fields of the XID frames used.

17.2.1 Address octets

Octets 1 and 2 represent the address field for a default two octet address. The first octet includes the C/R bit and the 6-bit upper subfield of the address. The second octet includes the 7-bit lower subfield of the address (including the two address bits taken for congestion control, the discard eligibility flag bit) and the address extension bit. The C/R bit is set to 1 to indicate that the XID used is a response. In method 1, the DLCI used is that of the primary PVC path, while in method 2, the DLCI is set to 8191.

17.2.2 Control field

Octet 3 contains the control field for the XID frame.

17.2.3 Format identifier field

Octet 4 contains the Format Identifier field. The value of 131 decimal is used for wideband packet applications.

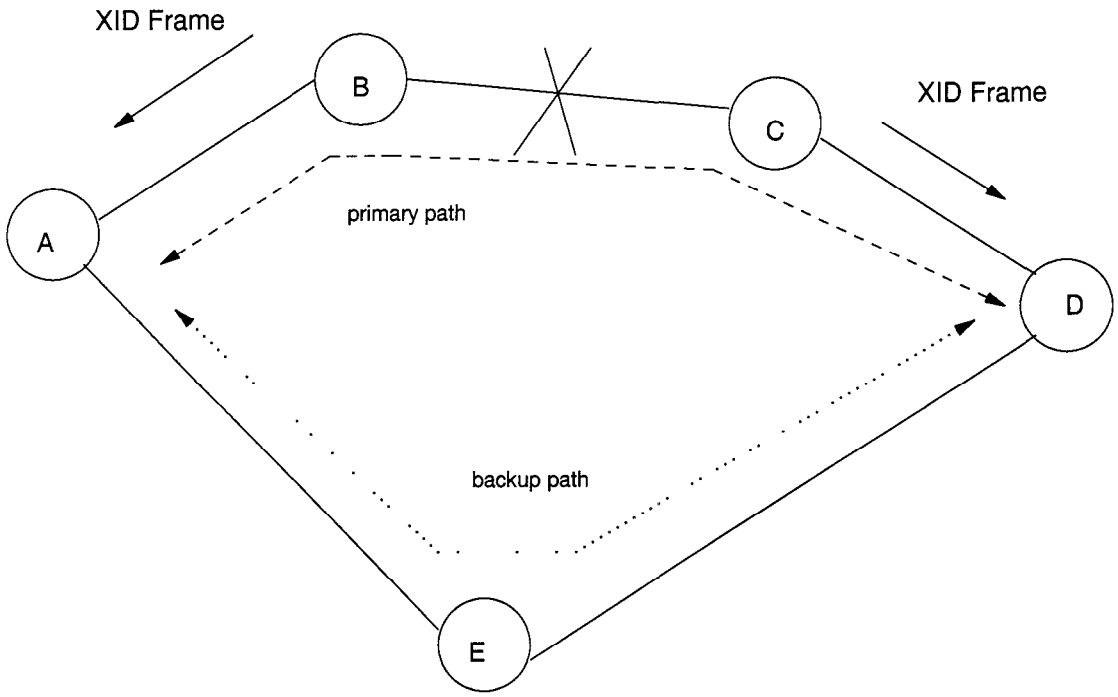


Figure 27 – Diagram for PVC restoration and switchback in a two-way connection

17.2.4 Group identifier field

Octet 5 contains the Group Identifier field. This field identifies the function of the XID information field. The value of this field is selected to distinguish the various uses of the XID frame with the same value for Format Identifier, decimal 131. The Group Identifier field is selected as follows:

a) For method 1, it is set to 243 for restoration through backup, and to 241 for switchback. This means that the frame shall be processed by the edge nodes only;

b) For method 2, the value shall be set to 242 for restoration through backup and to 240 for switchback. This means that all nodes in the path shall process the frame by updating the list of DLCIs in the information field of the XID frame and updating the group length field accordingly. It is also possible in this case that the arriving XID frame may be split into several XID frames, if the new DLCIs correspond to virtual circuits that have different physical links.

NOTE 1 – Method 2 may facilitate future internetworking with frame relay networks defined in CCITT Recommendation Q.922.

NOTE 2 – In method 2, the use of the same DLCI for all links of a PVC (i.e., on an end-to-end basis) may alleviate the computational load.

17.2.5 Group length field

In Figure 29, octets 6 and 7 contain the group length field. This 16-bit field describes the "length" of the octets in the remainder of this message, excluding FCS and closing flag.

17.2.6 DLCI value field

In Figure 29, octet(s) 8 and onward present the DLCI values that identify logical links that shall be backed up or switched back. For this field, the first octet shall represent the first octet of DLCI. The next octet shall represent the second octet of the DLCI.

17.2.7 FCS field

The last two octets of the message are for the FCS field. They are octets 6 and 7 in figure 28, and octets (7+2n+1) and [7+2(n+1)] in figure 29.

17.3 Procedures

17.3.1 Backup

After detecting a fault in the transmission path, a node shall identify the affected PVCs with backups that terminate at that node and shall generate an XID response frame to indicate that a fault has affected these PVCs. In method 1, the notification message is sent in the backward direction, using the primary path DLCI with the GI=243 (decimal). In method 2, the message is sent on DLCI=8191 and GI=242 (decimal) in the backward direction.

The format of the XID frames used is given above. The XID frame is an XID response frame, i.e., with the C/R bit set to 1.

When it receives such a message, any network node shall determine whether it terminates the affected virtual circuits. If it does, then:

a) In method 1, it shall initiate the switching to the backup links;

b) In method 2, it shall remove its DLCI from the DLCI list and translate the remaining DLCIs to the new values that correspond to the same PVCs on the outgoing physical link(s). The arriving XID frame may be split, if necessary, into one or more XID frames, each having the format of figure 31, depending on the association of the virtual circuits with the physical links.

If the network node does not terminate the affected virtual circuit, then:

a) In method 1, the node shall only translate the layer 2 address to that which is associated with the virtual circuit on the outgoing physical link;

b) In method 2, the translation shall be performed on all DLCIs in the information field of the XID frame. It is also possible in this case that the arriving XID frame may be split into several XID frames, if the new DLCIs correspond to virtual circuits that have different physical links.

17.3.2 PVC integrity check

In method 2, the end-to-end integrity of the PVC may be checked before the switchback. This is done after the alarm in the transmission path associated with a given PVC has been cleared. This integrity check is done by launching an XID command on the management link (DLCI=8191) and with the Group Identifier=239. The format of the command is given in figure 30. The node awaits for XID responses for each PVC and records the corresponding DLCIs. The format of response, given in figure 31, shows how the responding DLCI is recorded in the XID Response Frame. The responding DLCIs, if any, are then used in the switchback procedures.

Octet	Bits								Field Name
	8	7	6	5	4	3	2	1	
1							C/R	0	Address Octet 1
2								1	Address Octet 2
3	1	0	1	0	1	1	1	1	XID Control Field
4	1	0	0	0	0	0	1	1	Format Identifier (131)
5	1	1	1	1	0	0	1	1	Group Identifier Backup (243)
	1	1	1	1	0	0	0	1	Group Identifier Switchback (241)
6									FCS Octet 1
7									FCS Octet 2

Figure 28 – XID response frame for PVC backup/switchback (method 1)

Octet	Bits 8	7	6	5	4	3	2	1	Field Name
1	1	1	1	1	1	1	C/R	0	Address Octet 1
2	1	1	1	1	1	1	1	1	Address Octet 2
3	1	0	1	0	1	1	1	1	XID Control Field
4	1	0	0	0	0	0	1	1	Format Identifier (131)
5	1	1	1	1	0	0	1	0	Group Identifier Backup (242)
	1	1	1	1	0	0	0	0	Group Identifier Switchback (240)
6									Group Length Octet 1
7									Group Length Octet 2
8	–	–	–	–	–	–	–	–	DLCI Value Octet 1 (1st DLCI)
9									DLCI Value Octet 2 (1st DLCI)
	..	..	..	..	..	..	..	..	
	..	..	..	..	..	..	..	..	
	..	..	..	..	..	..	..	..	
7 + (2n –1)	–	–	–	–	–	–	–	–	DLCI Value Octet 1 (nth DLCI)
7+2n*									DLCI Value Octet 2 (nth DLCI)
7 + 2n +1									FCS Octet 1
7 +2(n + 1)									FCS Octet 2

Figure 29 – XID Response frame for PVC backup/switchback (method 2)

* Total length shall not exceed 256 octets.

17.3.3 Switchback

17.3.3.1 Without integrity check

After the alarm in the transmission path associated with a given PVC has been cleared, that node shall generate an XID response as a fault clearance indication message and transmit it on all the physical links associated with the other network nodes that have at least one virtual circuit affected by the failure. The GI is set to decimal 241 in method 1 and 240 in method 2.

The XID response can take one of the two formats displayed above.

When it receives such a message, any network node shall determine whether it terminates the affected virtual circuits. If it does, then:

a) In method 1, it shall initiate the switching back to the original links;

b) In method 2, it shall remove its DLCI from the DLCI list and translate the remaining DLCIs to the new values that correspond to the same PVCs on the outgoing physical link(s). The arriving XID frame may be split, if necessary, into one or more XID frames, depending on the association of the virtual circuits with the physical links.

If the network node does not terminate the affected virtual circuit, it shall relay the XID. In method 1, the node shall translate the layer 2 address to that associated with the virtual circuit on the outgoing physical link. In method 2, the translation shall be performed on all the DLCIs in the information field of the XID frame with the possible splitting of the arriving XID frame into one or more XID frames, depending on the association of the virtual circuits with the physical links.

17.3.3.2 With integrity check

This applies to method 2 only. If an integrity check is used, the switchback is initiated as above but only for the DLCIs that have

responded to the XID command frame for PVC integrity checking.

18 Packet stream backup

18.1 Overview

The packet stream restoration feature allows the PCME to backup the traffic from a failed packet stream on a preprovisioned backup packet stream. A packet stream is a collection of logical links multiplexed together onto one physical channel between two endpoints of the wideband packet network. The two streams provisioned as backup-pair act as primary and secondary transmission paths for traffic on either stream. In normal conditions, i.e., no facility failures, the traffic is routed on each primary stream so that the two streams are carrying their normal share of traffic. When the facility corresponding to either packet stream fails, the traffic is switched to the backup stream.

Both nodes terminating a packet stream need to activate backup/switchback action on that stream. For this purpose, node-to-node communication is required between the two end PCMEs. This is achieved via an XID message with GI field set to 245 for initiating the backup and 247 for initiating the switchback message. The XID messages shall be transmitted on the primary stream, and, after receiving the message, the PCME node shall activate or deactivate the backup action for the failed stream.

Octet	Bits								Field Name
	8	7	6	5	4	3	2	1	
1	1	1	1	1	1	1	0	0	Address Octet 1
2	1	1	1	1	1	1	1	1	Address Octet 2
3	1	0	1	0	1	1	1	1	XID Control Field
4	1	0	0	0	0	0	1	1	Format Identifier (131)
5	1	1	1	0	1	1	1	1	Group Identifier PVC Integrity Check (239)
6									Group Length Octet 1
7									Group Length Octet 2
8	–	–	–	–	–	–	–	–	DLCI Value Octet 1 (1st DLCI)
9									DLCI Value Octet 2 (1st DLCI)
	..	..	..	..	..	..	..	..	
	..	..	..	..	..	..	..	..	
	..	..	..	..	..	..	..	..	
7 + (2n –1)	–	–	–	–	–	–	–	–	DLCI Value Octet 1 (nth DLCI)
7 + 2n*									DLCI Value Octet 2 (nth DLCI)
7 + 2n +1									FCS Octet 1
7 +2(n + 1)									FCS Octet 2

Figure 30 – XID command frame for PVC integrity check

* Total length shall not exceed 256 octets.

Octet	Bits								Field Name
	8	7	6	5	4	3	2	1	
1	1	1	1	1	1	1	1	0	Address Octet 1
2	1	1	1	1	1	1	1	1	Address Octet 2
3	1	0	1	0	1	1	1	1	XID Control Field
4	1	0	0	0	0	0	1	1	Format Identifier (131)
5	1	1	1	0	1	1	1	1	Group Identifier PVC Integrity Check (239)
6	–	–	–	–	–	–	–	–	DLCI Value Octet 1 (1st DLCI)
7									DLCI Value Octet 2 (1st DLCI)
8									FCS Octet 1
9									FCS Octet 2

Figure 31 – XID response frame for PVC integrity check

18.2 Frame description

Figure 32 illustrates the format of the XID frame for the packet stream backup/switchback procedures. The following subsections describe the functional fields for the XID response.

18.2.1 Address octets

Octets 2 and 3 represent the address field for a default two octet address. The first octet includes the C/R bit and the 6-bit upper subfield of the address. The second octet includes the 7-bit lower subfield of the address (including the two address bits taken for congestion control, the discard eligibility indicator bit) and the address extension bit. The C/R bit is set to 1 to indicate that the XID used is a response. The DLCI used is any DLCI on the backup packet stream.

18.2.2 Control field

Octet 3 contains the control field for the XID frame.

18.2.3 Format identifier field

Octet 4 contains the Format Identifier field. The value of 131 decimal is used for wideband packet applications.

18.2.4 Group identifier field

Octet 5 contains the Group Identifier field. This field identifies the function of the XID information field. The value of this field is selected to distinguish the various uses of the XID frame with the same value for Format Identifier, decimal 131. The Group Identifier field is 245 for backup, and 247 for switchback. This means that the frame shall be processed by the edge nodes only.

18.2.5 FCS field

The last two octets of the message are for the FCS field.

18.3 Procedures

18.3.1 Backup

The node that detects a fault in the transmission path associated with a given packet stream shall generate an XID response frame to indicate that a fault has affected this packet stream. The notification message is sent on any DLCI with GI set to decimal 245.

The XID response takes the format displayed above.

When it receives such a message, any network node shall determine whether it terminates the affected packet stream. If it does, then it shall initiate the switching to the backup packet stream.

If the network node does not terminate the affected virtual circuit, the node shall only translate the layer 2 address to that which is associated with the packet stream on the outgoing physical link.

18.3.2 Switchback

After the alarm in the transmission path associated with a given packet stream has been cleared, that node shall generate an XID response frame as a fault clearance indication message and transmit it on the physical link

Octet	Bits								Field Name
	8	7	6	5	4	3	2	1	
1							C/R	0	Address Octet 1
2								1	Address Octet 2
3	1	0	1	0	1	1	1	1	XID Control Field
4	1	0	0	0	0	0	1	1	Format Identifier (131)
5	1	1	1	1	0	1	0	1	Group Identifier for Backup (245)
	1	1	1	1	0	1	1	1	Group Identifier for Switchback (247)
6									FCS Octet 1
7									FCS Octet 2

Figure 32 – XID response frame for packet stream backup/switchback

associated with the backup packet stream. The DLCI is any DLCI of the packet stream and the GI is set to decimal 247. The format of the XID response is given in figure 32.

When it receives such a message, any network node shall determine whether it terminates the affected virtual circuits. If it does, then it shall initiate the switching back to the original packet stream.

If the network node does not terminate the affected packet stream, it shall translate the layer 2 address to that associated with the packet stream on the outgoing physical link.

19 Generation of the alarm indication signal

The AIS is transmitted downstream of a transmission facility to indicate an upstream failure and prevent the propagation of the alarm to other downstream network equipment. Therefore, a node shall declare an AIS on a facility, if and only if, the following conditions are true:

– all channels (24 or 30 depending on the interface) are packetized on one packet stream;

– the primary path for this packet stream has failed and either there is no backup path or the backup path has also failed;

– there are no PVC backup connections for any of the channels (24 or 30).

20 Data collection

20.1 Transmission facilities monitoring

Each PCME should monitor each of the incoming and outgoing digital links for the following conditions and parameters, and store separate cumulative counts of each type of events as required by users: bit error ratio (BER), errored seconds (ES), and severely errored seconds (SES). Default and user provisionable thresholds for these performance measures are available for major and minor alarms. The user can display the current values upon request.

When a threshold is exceeded, the corresponding major or minor alarm occurs. Trunk processing operations minimize effects of errors following major or minor alarms. This results in the removal of the affected trunks from service and making them appear "busy" to other network elements. An alarm indication signal (AIS) should be sent.

Trunk processing consists of sending a data word and two different signaling words on each affected incoming full-rate 64-kbit/s channel. The first signaling word is sent during the first 2500 ms after detection of a failure condition. Transmission of the second signaling word continues for the remainder of the failure condition.

Other counters are for slips, out-of-frame errors and change of frame alignment, CRC-6 counter, bipolar violations.

The PCME shall collect in an hourly summary, for each circuit or packet connection:

– the number of layer 2 frames sent and received on each circuit or packet connection;

– the number of user information segments sent and received on each circuit or packet connection, where the length of a user information segment is a system administerable parameter;

– the number of speech bit blocks dropped;

– the number of speech, voiceband data, video, or V.120 data frames dropped due to congestion.

The hourly summaries shall be available by request over the Operations, Administration, and Maintenance (OA&M) interface, and they shall be kept for at least three days and until erased by command.

20.2 Traffic statistics

The PCME shall monitor and store records of the various parameters needed to evaluate the traffic handling performance.

20.2.1 Voice-band statistics

To be defined.

20.2.2 Digital data statistics

To be defined.

20.2.3 Facsimile statistics

To be defined.

20.2.4 Video statistics

To be defined.

20.3 Maintenance report

To be defined according to the guidelines of Monitoring of Error Performance (G.821).

21 Congestion control

Congestion control techniques can be functionally divided into several aspects:

– load measurement;

– network reaction after detection of congestion;

– notification messages sent to the users,

– user reactions.

The notification procedure must be flexible enough to fit the different traffic types that could be carried in a wideband packet network Subclause 20.1 describes the anticipated different traffic categories in future frame relay networks Subclause 20.2 presents the topological model on which this mechanism is based. Subclause 20.3 presents the consolidated link layer management congestion message for congestion control. Subclause 20.4 contains procedures that build upon this framework, taking into account different traffic categories. The network and user action are described in further detail.

Throughout the discussion, it is assumed that network congestion is an exceptional situation that requires prompt action. It is also assumed that receipt of the congestion notification by end user equipment does not create an error condition, although user's response is optional. It is assumed that the consolidated link layer management message will be sent between two nodes that have a common understanding of the LAPD address size. For the purpose of illustration, the default address shown is the default size of two octets The communication of the message between two subnetworks that employ different address sizes is an item for further study

21.1 Traffic categories in wideband packet networks

Potential traffic categories in a wideband packet network can be classified on the basis of their sensitivities at the user level to loss and to delay variation, as follows.

– *Type A:* Administrative Traffic, such as signaling and management traffic;

– *Type B:* Loss Sensitive/Delay Variation Tolerant Traffic, such as asynchronous and synchronous data transfer,

– *Type C:* Loss Sensitive/Delay Variation Sensitive Traffic, such as facsimile and continuous bit rate or isochronous data and video;

– *Type D:* Loss Tolerant/Delay Variation Sensitive Traffic, which includes packetized voice and embedded coded video,

– *Type E.* Loss Tolerant/Delay Variation Tolerant Traffic, such as telemetry traffic

There are several ways to distinguish among the various classes and sub-classes of traffic. One way is to use a Protocol Identifier to separate delay sensitive for delay tolerant traffic (time stamping is needed to process delay sensitive traffic and to avoid delay variations between successive packets of the same traffic burst)

Another way is to use different DLCI assignments as the "marking" agent. The use of DLCI may be on an individual or group basis. Furthermore, the DLCI space may be segmented with each range corresponding to a type of traffic. At a third level, the use of the Sub-Class field can be used to distinguish various types of packets for the same DLCI (e.g., voiceband, facsimile, or digital data).

NOTE – For the purpose of internetworking with frame relay networks, as defined in CCITT Recommendation Q.922, the use of DLCI as the distinguishing factor is appropriate. However, the exact method for internetworking is left for further study.

The following paragraphs highlight the main characteristics of each traffic type.

21.1.1 Type A: administrative traffic

Type A traffic includes the "control" and "management" traffic that would flow across a frame relay network, such as signaling and layer management traffic. This type of traffic flow can be characterized as having the following properties:

- very essential to the operation of a communications network;
- packet loss should be avoided as much as possible;
- packets of information must be delivered in a timely fashion (i.e., with minimum delay).

Examples of this type of information stream are the Q.931 signaling messages used for the purpose of negotiation of the services desired from the serving subnetwork, together with identification of facilities requested. Other examples include management messages for layer management (e.g., LAPD broadcast messages used in TEI assignment) as well as for the transport of OSI network management protocols.

Because administrative traffic is always present in a network, all nodes in a communication subnetwork will transport at least two forms of traffic.

21.1.2 Type B: loss sensitive/delay variation tolerant traffic

Type B traffic describes the transport of data information that has a non-continuous or bursty presentation of information at the access point to the network.

Examples of such traffic include (a) asynchronous data traffic from an asynchronous terminal to a host computer, and (b) bulk transfer data traffic. This type of traffic can be characterized as having the following properties:

- packet loss should be kept as low as possible. This is because the consequences of packet loss may be severe, e.g., session re-establishment;
- information is tolerant of delay. Information presented at the ingress node of a network may be queued and presented to the egress node in an asynchronous fashion;
- delay must be kept to a defined limit. However, this delay may exceed that of types A, C, and D.

This type of traffic could use the Discard Eligibility Indicator defined in Recommendation Q.922 to differentiate the presentation of traffic that exceeds the negotiated range.

21.1.3 Type C: loss sensitive/delay variation sensitive traffic

Type C traffic describes the transport of data information that has a continuous presentation of information at the access point to the network. This includes, for example, the support of facsimile and continuous bit rate or isochronous data and video.

This type of traffic can be characterized as having the following properties:

- packet loss should be kept as low as possible. The consequences of packet loss could be severe (e.g., session re-establishment);
- delay variation must be minimal to preserve the quality of the played back information;
- information is presented to the ingress network node/terminal adaptor as an uninterrupted bit stream and must be played out continuously at the egress node/terminal adaptor.

This type of traffic requires that all nodes of a subnetwork, together with end points must conform to the negotiated traffic rate determined at subscription time.

21.1.4 Type D: loss tolerant/delay variation sensitive traffic

Type D traffic includes information that may be arranged such that the same packet contains blocks of different levels of significance. Under congestion, the packet size may be reduced by shedding less significant blocks. Here, the main objective is to deliver information (even with some information loss) and variations in packet arrival at the terminating end must be limited. This is clear in the case of embedded coding of voice and video (continuous-bit-rate video is discussed in 20.1.3). Another example may correspond to data with advanced forward error correction methods. All blocks will be delivered as long as the delivery delay is acceptable. Otherwise, "less significant" blocks are dropped. The notion of "droppable" blocks infers that delivery is only necessary if the transit delay is under some threshold. Note that procedures for handling this form of traffic are described in ANSI T1.312.

This type of traffic can be characterized as having the following properties:

- partial information has more value than no information (i.e., some blocks of a packet may be dropped or some packets may be lost or dropped);
- packets of information must be delivered in sequence and at a regular intervals (i.e., with minimum delay variation).

21.1.5 Type E: loss tolerant/delay variation tolerant traffic

Type E traffic includes information that arrives infrequently to the access point of a network. It also includes frames that the network node has marked as "discardable." It is not required that all frames be delivered and delay is tolerated. Examples of this type of traffic include telemetry traffic.

This type of traffic can be characterized as having the following properties:

- partial information has more value than no information (i.e., some packets may be lost or dropped);
- packets of information must be delivered in sequence but the delivery interval may be irregular (i.e., tolerant of delivery delay variation).

21.1.6 Explicit congestion notification

A frame relay node will support two or more traffic types concurrently. Therefore, properly designed network nodes will have the ability to distinguish traffic by service categories and to take appropriate action in the event of congestion. If needed, the distinction of traffic types may be through several means such as the use of the DLCI assignment.

21.2 Topological model

The topological model used in this proposal is shown in figure 33. Node Y is the node under congestion; it has generated an explicit congestion notification message. Node X is in the backward direction from the congested node, and node Z is in the forward direction. UNI is the User to Network Interface.

There are two independent algorithms used when congestion is declared. The network algorithm is used to protect network resources and/or to maintain with high probability the negotiated quality of service and/or bandwidth. This algorithm affects both the end user traffic and the network administration traffic.

The second type of algorithm is the end user algorithm, which controls the offered load from the end user.

21.3 Action of the congested node

When a wideband packet node becomes severely congested, it determines the DLCIs of all permanent logical links connected to the congested resource and then sends a notification using one or more consolidated link layer management messages. The purpose of the consolidated link layer management message is (a) to inform the edge nodes of the network of the current state of the congestion (i.e., congestion is getting worse, is staying the same, or is decreasing), and/or (b) to notify the source that negotiated traffic ranges have been exceeded. The congested node may also drop frames that have been identified as "discardable," according to the procedures of CCITT Recommendation Q.922.

The edge node may take action to reduce network congestion such as blocking additional call establishment. Estimation of the network state is network-dependent.

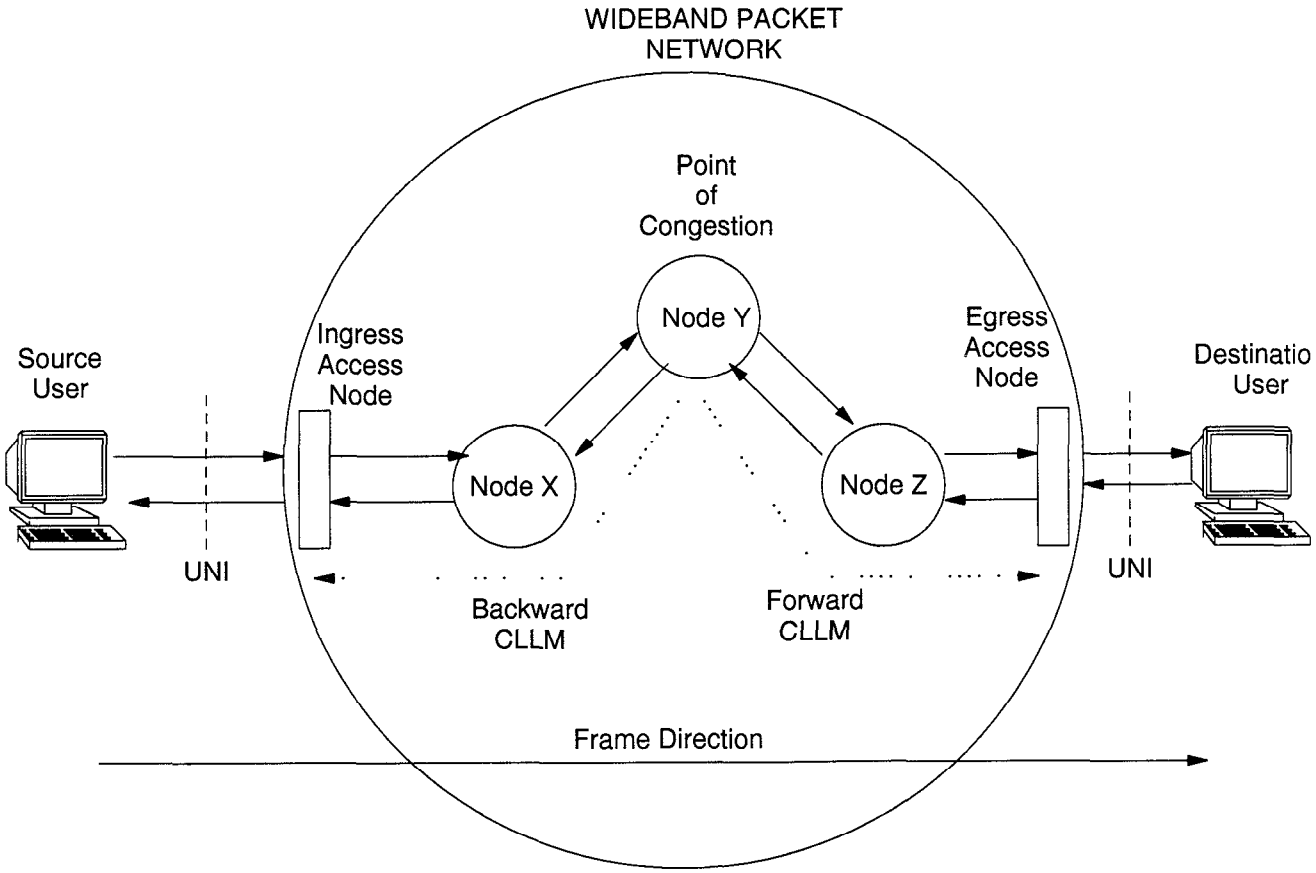


Figure 33 – Wideband packet network topology

There are three congestion states in a node: Normal, Moderate Congestion, and Maximum Congestion.

– *Normal State:* There is no congestion;

– *Moderate Congestion State:* the node has reached a level of congestion in which the end user and network administration traffic has increased to a point greater than the available network resources. The node and/or network is not in danger of collapse, but it cannot maintain the agreed upon quality of service and/or bandwidth;

– *Maximum Congestion State:* The node and/or network is in danger of collapse and immediate traffic reduction is required.

The network node determines the current congestion states and its action depends on the past congestion state:

a) If the Congestion State is Normal, and the previous Congestion State was Normal, then no message is generated;

b) If the previous Congestion State was Normal, and a change of state has occurred, then the appropriate state transition message is generated. The node generates messages every T_{cong} seconds to update the congestion level information. The minimum value of T_{cong} is 10 seconds; other values are for further study. Once the node has returned to the Normal State, the message generation is terminated.

21.3.1 Transition messages

The following six state transition messages are used:

– *Congestion increasing*: The level of congestion has increased to a Moderate Congestion level from the Normal State. The congestion is not, however, increasing fast enough to bring down the network;

– *Congestion clearing*: The level of congestion has reduced since the last measurement and/or message. The congestion has disappeared, and the State has changed from Moderate to Normal;

– *Sustained congestion*: The level of congestion has not changed since the last measurement and/or message. This message can be generated when the node is either in the moderate congestion state or in the maximum congestion state;

– *Critical congestion*: The level of congestion has increased since the last measurement and/or message. The level of congestion indicates that the network is in danger of collapse and requires immediate action;

– *Congestion abating*: The level of congestion has decreased since the last measurement and/or message. The congestion has not cleared, and the Congestion State has changed from Maximum Congestion to Moderate Congestion;

– *Critical congestion clearing*: The level of congestion has decreased considerably since the last measurement and/or message and the State has changed from Maximum Congestion to Normal.

The state transition diagram is shown in figure 34.

The consolidated link layer management message lists all the DLCIs that correspond to the congested packet stream(s). These DLCIs will correspond to sources that are currently active and those that are not. The purpose is to prevent those sources that are not active from becoming active and hence increasing congestion. This is particularly needed for bursty traffic such as traffic types B and D.

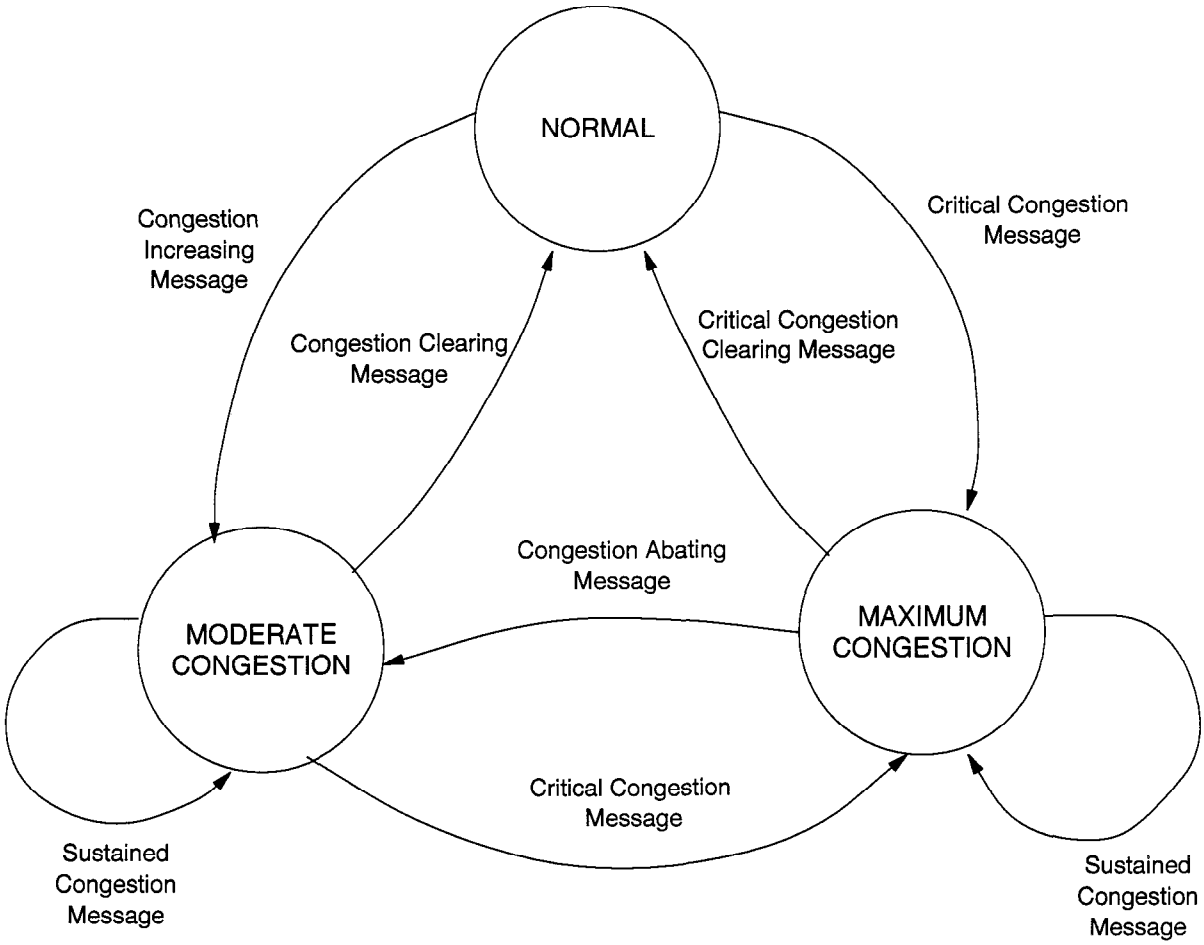


Figure 34 – State transitions of a congested wideband packet node

It may be necessary to send more than one consolidated link layer management message to notify all the logical links whose DLCIs are associated with the congested packet stream(s).

Further actions, if any, depend on the type of traffic, as explained below.

– *Type A Traffic*: No further action is needed.

– *Type B Traffic*: If the node experiences congestion, then a notification is sent. If the congestion persists, frames marked as discardable are dropped.

– *Type C Traffic*: If the node experiences congestion, the network node shall inform the edge node to block additional calls.

– *Type D Traffic*: If the node experiences congestion, the network node may reduce the coding rate for voice or data. If the congestion persists, the network node identifies the channels that may be inactivated and then generates a message in the backward direction to block additional call establishment.

Furthermore, other messages may be used as well. For example, an ANSI T1.312 signaling packet with the Normal/Alarm bit set to 1 may be used as explicit congestion notification in the forward direction for channels that have channel-associated signaling and that have trunk conditioning as an administered option.

– *Type E Traffic*: If the node experiences congestion, a notification is sent. If the congestion persists, then frames marked as discardable are dropped.

21.4 Network node response to CLLM

Network nodes can react in different ways to the Consolidated Link Layer Management Congestion messages.

21.4.1 Network ingress node reaction

After receiving a consolidated link layer management message, a network node interprets the list of congested DLCIs and then translates it to outgoing DLCIs. Because the outgoing DLCIs may not always be on the same packet stream or access facility, multiple outgoing messages may be generated in the backward direction in response to a single incoming message. However, a single message will contain several DLCIs from the same packet stream. At the edge of the network, the access channels are made to appear "busy" ("trunk conditioning") for some or all of the calls so that no further traffic is routed to these congested access channels. This will reduce adding to the congestion level. A second action would be to prevent the establishment of all new calls, which would eventually reduce the congestion level for that access channel.

If the node interfaces with an International Switching Center (ISC), the ISC procedures of Recommendation Q.50 shall be followed.

21.4.2 Operation of wideband packet network node in the presence of congestion

The actions refer to two aspects: (a) treatment of the traffic, and (b) generation of CLLMs.

21.4.2.1 Traffic treatment

Once an explicit notification of congestion has been made, the network node will examine the incoming traffic and may serve different categories of traffic with different priorities of processing attention. For example, if the speech is coded according to the embedded ADPCM algorithms of ANSI T1.310, one or several blocks containing the least significant bits may be dropped. The node may also drop frames that have been marked as discardable. Once the congestion has been eliminated, the network node may once again resume serving all traffic.

21.4.2.2 Message generation

The network node determines the current congestion states and its action depends on the past congestion state:

a) If the Congestion State is Normal, and the previous Congestion State was Normal, then no message is generated;

b) If the previous state was Normal, and a change of state has occurred, then the appropriate state transition message is generated. The node generates messages every T_{cong} seconds to update the congestion level information. Once the node has returned to the Normal State, the message generation is terminated.

In general, CLLM messages are sent from the point of congestion and terminated at the edge nodes. However, in the case of connections to frame relay networks and terminals, the edge

node may pass the CLLM onwards across the Network to Network Interface or the User to Network Interface.

NOTE – The use of the same DLCI for all links of a permanent virtual circuit (i.e., on an end-to-end basis) may alleviate the computational load.

21.5 End-user's response to the CLLM

The end-user's equipment response is optional; however, receipt of the congestion notification by end user equipment shall not create an error condition.

The following proposal presents different ways that end-user equipment can react to congestion notification by a wideband packet network.

No action should be taken for types A, D and E traffic.

21.5.1 Type B traffic

21.5.1.1 End-user's action

– *Congestion increasing*: The end user should reduce the offered load (to the next step if possible). Since the node is in the Moderate Congestion State, the service is only degraded. The end user should, therefore, be allowed to implement an algorithm that will reduce the offered load, if this can increase the service quality to some of the end user traffic, while reducing or degrading further other end user traffic. These ideas set the stage for the bandwidth enforcement or dynamic change of coding algorithms feature.

– *Congestion clearing*: The end user should allow the offered load to increase to return service quality to the negotiated value for all traffic. The end user may, however, implement the return following an algorithm that protects the network against "flip-flop" between the Normal and Moderate Congestion States.

– *Sustained congestion*: The end user may either keep the offered load at the current reduced rate, or further reduce the offered load to force the node into the Normal State.

– *Critical congestion*: Since the level of congestion has increased considerably so that the network is in danger, the end user must reduce the offered load so that the node is at least forced back to the moderately Congested State (this action will protect the network against the situation in which an end user algorithm has not been altered to take into account network growth, and so a "faulty" end user algorithm cannot fail to bring runaway congestion under control).

– *Congestion abating*: The return to the Moderate Congestion State will also allow the end user to adjust the offered load. The end user may increase the offered load, or re-implement the algorithms used when the Congestion Increasing message is received.

– *Critical congestion clearing*: The end user should allow the offered load to increase, to return service quality to the negotiated value for all traffic. The end user may, however, implement the return following an algorithm that protects the network against "flip-flop" between the Normal and Critical Congestion States.

21.5.1.2 Load reduction

Reduction of the end user's information transfer rate should be guided by the following guidelines:

– the end user adjustment of offered load should be rate based (to distinguish from window based mechanisms);

– some lower bound on rate reduction is to be established to achieve fairness among end users;

– rate reduction steps should not be too drastic to eliminate big fluctuations in throughput;

– reduction in response to implicit notification should depend on whether or not an explicit notification has been received.

For example, upon the reception of the first CLLM, the user should reduce its offered load to the negotiated Committed Information Rate (CIR), which is the rate at which the network agrees to transfer information under normal conditions (see ANSI T1.606 and ANSI T1.606a). If the operating point is at or below this level, no reduction is required. This will achieve some fairness among users operating at different load levels and contributing differently to congestion. When a second CLLM is received the user shall reduce its offered load to 75% of negotiated CIR. If a third CLLM is received, the user shall reduce its offered load to 50% of the CIR. No further reduction is required in response to explicit notification messages. This avoids penalizing

complying users, thus achieving some degree of fairness.

If the user can operate at the new reduced rate without receiving any type of congestion notification (explicit or implicit), the user is allowed to raise the offered load to 75% of the CIR. If operation at the new increased load is possible without receiving a congestion notification, the user can increase the rate again. The first increase is to 75% of CIR. The second increase brings the offered load to the negotiated CIR. The rate can then be increased to the original offered load.

An implicit notification received following an explicit one should be interpreted as an indication of severe congestion, and the rate should be reduced to 50% of CIR level, irrespective of the current offered load. An implicit notification not preceded by an explicit one is likely to have been caused by bit errors and requires more moderate reduction in offered load.

21.5.2 Type C traffic

Upon the reception of a congestion notification, the access point shall check that the presentation of data to the ingress node of the network is at the subscription rate. A continued violation of this subscription rate will lead to overflow conditions in the serving network. The resolution of this error event remains for further study. A second action would be to prevent the establishment of new calls.

21.6 Frame format for the consolidated layer management congestion message

Figure 35 illustrates the format of this CLLM frame.

21.6.1 Address octets

Octets 1 and 2 represent the Address field for a default two octet address. The first octet includes the C/R bit and the 6-bit upper subfield of the address. The second octet includes the 5-bit lower subfield of the address, the two address bits taken for congestion control and the address extension bit. The Consolidated Layer Management Congestion message is sent over DLCI=8191 to ensure compatibility with the D-channel.

21.6.2 Control field

Octet 3 contains the control field code point for this type of message. This represents the control field for XID.

21.6.3 Format identifier field

Octet 4 contains the Format Identifier field (decimal 131).

21.6.4 Group identifier field

Octet 5 contains the Group Identifier field (decimal 250).

21.6.5 Group length field

Octets 6 and 7 contain the Group Length field. This 16-bit field describes the "length" of the octets in the remainder of this message, excluding FCS and closing flag. For compatibility with D-channel applications, the maximum length is set to 260.

21.6.6 Cause identifier value

Octet 8 contains the Cause Identifier, which identifies the following causes:

– *Congestion increasing*: The level of congestion has increased to a Moderate Congestion level from the Normal State. The congestion is not, however, increasing fast enough to bring down the network;

– *Congestion clearing*: The level of congestion has reduced since the last measurement and/or message. The congestion has disappeared, and the State has changed from Moderate to Normal;

– *Sustained congestion*: The level of congestion has not changed since the last measurement and/or message. This message can be generated when the node is in either the Moderate Congestion State or in the Maximum Congestion State;

– *Critical congestion*: The level of congestion has increased considerably since the last measurement and/or message. The level of congestion indicates that the network is in danger of collapse and requires immediate action;

Octet	Bits								Field Name
	8	7	6	5	4	3	2	1	
1	1	1	1	1	1	1	C/R	0	Address Octet 1
2	1	1	1	1	1	1	1	1	Address Octet 2
3	1	0	1	0	1	1	1	1	XID Control Field
4	1	0	0	0	0	0	1	1	Format Identifier (131)
5	1	1	1	1	1	0	1	0	Group Identifier (250)
6									Group Length Octet 1
7									Group Length Octet 2
8									Cause Identifier Value
9	0	0	0	0	0	0	0	1	Parameter Identifier = 1 (Network Identifier)
10	0	0	0	0	0	0	1	0	Parameter Length = 2
11	–	–	–	–	–	–	–	–	Network Identifier Value
12									
13	0	0	0	0	0	0	1	0	Parameter Identifier = 2 (DLCI Identifier)
14									Parameter Length
15	–	–	–	–	–	–	–	–	DLCI Value Octet 1 (1st DLCI)
16									DLCI Value Octet 2 (1st DLCI)
	..	..	..	..	..	..	..	..	
	..	..	..	..	..	..	..	..	
	..	..	..	..	..	..	..	..	
14+(2n –1)	–	–	–	–	–	–	–	–	DLCI Value Octet 1 (nth DLCI)
14+2n*									DLCI Value Octet 2 (nth DLCI)
14 + 2n +1									FCS Octet 1
14 +2(n + 1)									FCS Octet 2

Figure 35 – Consolidated layer management congestion message

* Maximum number of octets is 260.

– *Congestion abating*: The level of congestion has decreased since the last measurement and/or message. The congestion has not ceased, and the Congestion State has changed from Maximum Congestion to Moderate Congestion;

– *Critical congestion clearing*: The level of congestion has decreased considerably since the last measurement and/or message and the State has changed from Maximum Congestion to Normal.

The codes for the cause field are shown in table 13.

Table 13 – Cause field codes

Bits	Field Name
8 7 6 5 4 3 2 1	
0 0 0 0 0 0 1 0	Congestion Increasing
0 0 0 0 0 0 1 1	Congestion Clearing
0 0 0 0 0 1 0 0	Sustained Congestion
0 0 0 0 0 1 1 0	Critical Congestion
0 0 0 0 0 1 1 1	Critical Congestion Clearing
0 0 0 0 1 0 0 0	Congestion Abating

21.6.7 Parameter field for network identifier

Octet 9 contains the Parameter Identifier field for the Network Identifier. When the Parameter Identifier field is set to 1, then the following octets of this parameter contain the Network Identifier.

21.6.8 Parameter length field

Octet 10 contains the length of the Network Identifier Length field. In figure 35, the value shown is decimal 2; however, other values may be chosen.

21.6.9 Network identifier value field

In figure 35, octets 11 through 12 contain the value of the Network Identifier that identifies the network that originated the congestion message. The Network Identifier shall follow the rules in clause 5 of CCITT Recommendation E.164.

21.6.10 Parameter field for DLCI identifier

Octet 13 contains the Parameter Identifier field for the DLCI Identifier. If the DLCI Identifier field is missing, then the frame shall be ignored. When the Parameter Identifier field is set to 2, then the following octets of this parameter contain the DLCI(s) of the frame relay links that are congested.

21.6.11 Parameter length field for DLCI identifier

Octet 14 contains the total length in octets of the DLCI(s) being reported. For example, if (n) DLCIs are being reported and they have lengths of two octets each, then the octet size is (n) times (2).

21.6.12 Parameter value field for DLCI(s)

The following octets present the DLCI values that identify the logical links that have encountered a congested state. In figure 35, these values are in octets 15 and up. For this field, the first octet contains the first octet of the address field, while the second octet contains the second octet of the address field.

21.6.13 FCS field

The last two octets of the message contain the frame check sequence field.

21.7 User notification message

The same format is used at the Network-Network Interface and the User-Network Interface. The notification of the user terminal equipment of the presence of congestion (by the edge network node) will require a user notification message. This message could be of the same format as the Consolidated Link Layer Management message. However, the list of DLCIs would not necessarily be the same.

21.8 Actions in special cases

Typical PVCs are usually two-way connections. Therefore, the DLCI for the transmit and receive directions usually will be the same value.

In some cases, the DLCI in one direction may be different than the DLCI for the reverse direction, such as in broadcast circuits as applied to the multidestinational configuration of satellite connections.

a) In this case, the node receives a congestion message, may determine the origin of the message, and then reduce the part of the traffic that is going to the congested node only;

b) In packet/circuit broadcast or concentration applications, there will be mapping of two or more DLCIs to a single 64-kbit/s channel. If an ingress node receives a congestion message, the end user on the access side of the ingress node shall be informed. In these situations, the ingress node may either:

– disable packetization for all logical channels and/or circuits transmitting to the network, or;

– disable packetization for a subset of the affected connections. The subset shall include all traffic destined for the congested node.

c) In a concentration application, there will be mapping of two or more logical channels to a single logical channel, or two or more circuits on a logical channel, or two or more logical channels on a circuit. When an intermediate node is congested or has received a congestion message, it shall generate congestion messages for all PVC paths through the node.

Note that a packet stream backup may cause the bandwidth to be shared between two packet streams. The situation, however, may cause congestion.

It may not be fair to the users of one packet stream to experience some degradation of service due to the congestion caused by backing up other users. To avoid this, the node may have the ability to distinguish between the DLCIs of both packet streams so that the source of the DLCIs can be distinguished.

In the event of a congestion state being declared, the node may be able to reduce traffic of a backed-up packet stream first, instead of reducing the traffic of both packet streams simultaneously. Note that the node may be able reduce the non-backed up traffic first, if there is a hierarchy of traffic importance.

21.9 Variable bandwidth links

In a variant of the topological model described in 20.2, one or more of the communication channels between the nodes may provide bandwidth on demand by physical expansion of the channel capacity, e.g., satellite links using Load-Adaptive TDMA. In this case, an additional congestion relief action is possible by increasing the channel capacity of one or more channels serving the node, subject to availability. This topic is for further study.

22 Dynamic load control

The interface of a packet circuit multiplication equipment with a switch is defined in CCITT Recommendation Q.50. Actions taken between PCME are for further study.

23 Interface with Q.922 frame relay networks

Internetworking with frame relay networks defined in CCITT Recommendation Q.922 is for further study.

24 Synchronization

The PCME shall be integrated into the synchronization hierarchy as described and specified in ANSI T1.101. The clocks used to synchronize the PCME shall be stratum 3 or better quality. In the case of SONET interfaces, ANSI T1.105 shall apply.

Timing and synchronization application guidelines are contained in Annex A to T1.105. Note that while the channelized-side interfaces and the packetization/depacketization functions must operate synchronously, the packetized side interfaces can each operate asynchronously, since layer 2 interframe fill can be adjusted to compensate for individual rate differences.

Annex A
(normative)

Performance Considerations

This annex primarily addresses speech performance. Performance considerations of other types of traffic (digital data, voiceband data, facsimile, video, signaling) are left for further study.

A.1 Signal classification errors

The signal classifier shall not classify voiceband data at a lower rate than the actual rate or as speech more than five times per thousand calls.

The signal classifier shall not classify voiceband data at a higher rate than the actual rate or as speech more than 50 times per thousand calls.

The combination of the signal classifier and speech coder shall operate such that the initial training of voiceband modems operates at the same error rate as for the entire call.

A.2 Speech detection

The Packet Circuit Multiplication Equipment (PCME) shall incorporate a speech detector to eliminate silent intervals and stop packetization.

The speech detector shall operate such that the speech activity measured by packet flow does not exceed the actual speech activity by more than 5%. For example, if the actual speech activity is 38%, then the speech activity measured at the packet interface shall be less than 43%.

The mean opinion score (MOS) measured when silence elimination is activated, but without bit dropping or packet loss, shall not be more than 0.3 less than the MOS without silence elimination. The MOS is a value obtained by using the Absolute Category Rating (ACR) method that the CCITT has defined in Annex A of Supplement No. 14 to the P Series Recommendations on "Telephone Transmission Quality."

A.3 Packet loss

Speech packets may be lost due to errors in transmission or to queue underflow or overflow at the terminating end. The PCME shall include means to minimize these effects so that the MOS for 1% packet loss is not worse than the normal MOS by more than 0.3.

A number of fill-in strategies can close the gaps in the talk spurt:

- replace the discarded packets with samples of zero-amplitude values or with background noise;
- repeat the most recent packet;
- reconstruct the lost packet by sample interpolation using the samples that are in the arrived packets;
- use speech segments from the last received packets to replace the missing speech segments.

Although silence substitution or noise interpolation is easy to implement, it can degrade the subjective quality of voice. Packet repetition has less stringent requirements on memory space and processing power. The repetition algorithm may vary if speech is classified in several classes. For example, the last pitch waveform may be repeated for voice segments, while the last packet may be used otherwise.

A.4 Noise fill

Noise shall be filled in to replace the silent intervals eliminated from the talk spurt at the originating end.

The noise shall be white and within 1 dB of the value in the noise field of the last received G.764 packet. It shall be free of buzzes, tones or other audible clues to regularity.

Annex B
(informative)

Augmented Protocol Specification Language

B.1 The Augmented Protocol Specification Language (APSL)

The Augmented Protocol Specification Language (APSL) is an extension of the Protocol Specification Language (PSL) developed by Sabnani and Lapone to specify protocols formally using communicating Finite State Machines (FSMs). Formal specification deals with the behavior of these FSMs with respect to the external world and does not consider the internal events of the protocol. Therefore, PSL does not include a succinct syntax to specify internal operations (e.g., the update of internal counters and setting of flags). A concise way of specifying these low-level details is necessary in the design and implementation of hardware protocol controllers. Therefore, PSL was extended to the APSL, whose syntax and semantics are described in the next clause.

B.2 The syntax of APSL

In APSL, a protocol is described as a collection of communicating FSMs, each specified as an APSL process with its own input and output message sets. External inputs and outputs are designated as belonging to *external* processes. Each communicating FSM is specified as an APSL process in the following form:

EXTERNAL *ext_proc1, . , ext_procn*;

PROCESS *process_name*;

CONTEXT *variable_name*;

STATES *n0-n1, n2-n3, n4, ... ,nk-nl*;

REND *src1?msg1,src1?msg2, ... src1!msga,src1!msgb*;

INITIAL STATE *N0*;

TRANSITIONS
transition_1,
transition_2,
...
transition_k.

The order of the statements is to be followed strictly. The following subclauses explain the meanings of the various declaration statements.

B.2.1 EXTERNAL statement

The EXTERNAL statement is optional. Its purpose is to allow the independent checking of the specifications of individual processes and to model external inputs and outputs in a consistent manner.

B.2.2 PROCESS statement

The PROCESS statement names the process. Acceptable names are alphanumeric strings that start with an alphabetic character.

B.2.3 CONTEXT statement

The CONTEXT statement lists the context variables of the process. Acceptable variable names are alphanumeric strings that start with an alphabetic character. Context variable may be entered in any particular order.

B.2.4 STATES statement

The STATES statement lists the states of the process. Acceptable state names are alphanumeric string that start with an alphabetic character, but are currently restricted to numbers. The compiler ignores unused states and renumbers all states sequentially in the internal representation of the processes. A range *n0-n1* indicates all the numbers from *n0* to *n1*, both inclusive.

B.2.5 REND statement

The REND (from *rendez-vous*) statement defines all the I/O messages associated with the process. These are the messages through which the FSMs of the protocol communicate among themselves and with the external world. Messages are of the form *a?mi* or *b!mj*, where *a?mi* indicates the reception of message *mi* from the machine *a* and *b!mj* indicates the transmission of message *mj* to the machine *b* The semantics of this notation are similar to that of the CSP notation developed by Hoare, the key characteristic being that it requires both a sender and a receiver and is a synchronizing primitive.

B.2.6 INITIAL STATE statement

The initial state of the machine is specified by the INITIAL STATE statement. The initial state must be among the states specified in the STATES statement.

B.2.7 TRANSITIONS statement

The general form of a transition in APSL is

```
i:j WHEN {condition} a?m1 * b!m2 [update-operation]
```

where the syntactic elements *condition* and *update-operation* are extensions to PSL The *condition* is a test involving a conditional expression containing internal (context) variables or the keyword DEFAULT (*see below).* Updates of internal variables are done in the *update-operation* specification.

While no major restrictions are placed on the *condition* and *update-operation* parts of the transition specification, hardware architectures may restrict the complexity of the condition that can be tested in one cycle. A *simple* condition is an expression of the type

```
a > b
```

or

```
a + b <= c
```

where *a*, *b*, and *c* are variables or constants. Furthermore, most hardware implementations allow only one assignment per cycle. Therefore, we define the *canonical* form of the specification to include *simple conditions* only and to have at most one *update-operations*. The user writes an initial protocol description and the compiler translates it into the canonical form, generating any

needed intermediate states. For example, the user may input the following description:

```
i:j WHEN { C1 | (C2 & C3 ) } a?m1 * b!m2 [ A1; A2; ]
```

where Ci are conditional statements and Ai are update operations, becomes after translation:

```
i:t1  WHEN {C1} a?m1 * b!m2 [A1;];
t1:j  WHEN [A2;];
i:t2  WHEN {C2};
t2:i  WHEN {~C3};
t2:t1 WHEN {C3} a?m1 * b!m2 [A1;].
```

Here t1 and t2 are intermediate states that the APSL compiler have introduced; they remain invisible to the external world. To simplify the writing of specifications, the keyword DEFAULT can be used in place of a conditional expression. The transition marked with a DEFAULT condition *cannot* have any input messages. It can have output messages and/or update operations. There can be only one DEFAULT transition per state. The DEFAULT transition for a state is activated if *none* of the other transitions from that state can be taken Thus, even though the DEFAULT keyword is used in place of a condition, it actually applies to the conjunction of condition and input messages. The update operations are restricted to increment, decrement, add, subtract, and assignment operations. For example, if the specification has an input message *a?m1* in machine *b*, the specification of machine *a* must contain the output message *b!m1*. The compiler flags as errors messages those that do not occur in matched pairs. This check is by-passed, however, for machines that are specified as EXTERNAL.

B.3 Some semantic restrictions

The machines specified in APSL are assumed to be deterministic machines. This requires that, in any given state, only one outgoing edge be enabled under the given conditions

Consider the statements

```
i:j WHEN {C1} a?m1 * b!m2 [update-operation]
i:k WHEN {C2} c?m4 * b!m3 [update-operation]
```

When the machine is in state *i*, the conditions are checked in some order and the first condition to succeed enables the corresponding edge. Thus if $C1$ is true, the machine goes from *i* to *j* on the reception of message $m1$ from machine *a*. The conditions, however, are checked only on the occurrence of an input event. Thus if message $m1$ appears as input, then $C1$ is evaluated. if it evaluates true, the transition *i* to *j* occurs with the attendant updates; otherwise, the transition does not occur and the input message is consumed. Furthermore, the set of the conjunction of conditions and input events on all the outgoing edges from a state must be mutually exclusive. In other words, if the conditions are $C_1, C_2, \cdots, C_n$, and the input events are $I_1, I_2, \cdots, I_n$, then

$$(C_i \cap I_i) \cap (C_j \cap I_j) = \varnothing, \quad i \neq j.$$

Annex C
(informative)

Formal Description of the Digital Circuit Emulation Protocol (DICE)

C.1 Processes in APSL Description

Table C.1 lists all the processes used in the Augmented Protocol Specification Language (APSL) specification of the DICE Protocol. It includes a brief description of each:

Table C.1

Process name	Description
data_collector	Data collector at the originating end
hi_layer_orig	High layer entity at the originating end
hi_layer_term	High layer entity at the terminating end
idle_code_detector	Idle code detector at the originating end
interm_pt_dice_relay	DICE intermediate node
parallel_to_serial_conv	Converter of an octet to a series of bits at the originating end
playout_dice	DICE packet layer playout
pre_processor	Pre-processor at the originating end
rcv_dice	DICE packet layer receiver at the terminating end
rcv_queue	DICE packet layer queue at the terminating end
serial_to_parallel_conv	Converter of a series of bits into an octet at the terminating end
xmtr_dice	DICE packet layer transmitter at the originating end
t_idle	Idle update timer
tvdelay_interm	DICE variable delay timer at the intermediate node
tvdelay_orig	DICE variable delay timer at the originating end
clock	Clock
layer2_interm	Layer 2 at the intermediate node
layer2_orig	Layer 2 at the originating node
layer2_term	Layer 2 at the terminating node
mgmt_term	Management entity at the terminating end
user_orig	User process at the originating end
user_term	User process at the terminating end

The following are treated as EXTERNAL processes: clock, layer2_orig, layer2_interm, layer2_term, mgmt_orig, user_orig, and user_term.

Table C.2 lists messages that apply for any timer.

Table C.2

Message name	Description
timer!expired	The timer has exceeded its time-out value.
timer?start	Start the timer.
timer?stop	Stop the timer and reset all variables to zero.

C.2 Functions in the APSL Description of DICE

Table C.3 alphabetically lists the various functions used in the APSL specification of DICE:

Table C.3

Function	Description
chk_idle_tbl()	At the originating endpoint, checks if 'octet' corresponds to an idle code in Table 1 at the bit rate defined in 'speed', then sets the variables 'idle_orig' and 'normal_idle' to TRUE or FALSE accordingly.
count_bits()	Counts the number of bits in the DICE information field.x
drop_pkt()	Drops the current packet.
get_idle_octet()	Checks if the IBT value is a valid entry in Table 3. If IBT is valid, it sets 'idle_term' to TRUE and then maps the IBT value to 'idle_code' using Table 3; otherwise, it sets 'idle_term' to FALSE
get_last_ibt_octet()	Maps the value of 'ibt_last' into an idle code using Table 3 and puts the idle code in out_octet
get_octet()	Reads octet of data and puts it in 'playout_octet ' The data are arranged as shown in Figure 5, i.e., bit 1 of playout_octet is the least significant bit of the data while bit 8 is the most significant bit.
ibt_update()	Updates the value of the ibt variable based on the value retrieved from the idle code table that corresponds to the idle octet
output_delay()	Outputs the delay between reception and transmission of a packet at a node. This delay is measured by the tvdelay_dice process for DICE packets.
pop_pkt()	Takes a packet off the playout queue for playout and sets the terminating end variables 'eq_term', 'ibt_term' and 'seq_term' accordingly.
push_pkt()	Queues a received packet and sets the corresponding playout_time variable.
rseq_increment()	Increments the receive sequence number (RSEQ) (1 to 15 with rollover to 1).
seq_increment()	Increments the sequence number seq_orig (1 to 15 with rollover to 1) at the originating end and seq_term at the terminating end.
set_playout_time()	Sets the variable 'playout_time' to BUILDOUT - time stamp + current_time
set_pkt_length()	Calculates the packet length in octets and stores it in the variable 'pkt_length'.
ts_update()	Updates the time stamp by the value of the delay measured by the 'tvdelay_orig' or 'tvdelay_interm' timers.

C.3 Primitives

Table C.4 describes the various primitives used in the APSL description:

Table C.4

Primitive	APSL representation
DL_L1_READY_INDICATION	dl_l1_ready_indication
DL_PVP_DATA_REQUEST	dl_pvp_data_request
DL_PVP_H_DATA_REQUEST	dl_pvp_h_data_request
DL_UNIT_DATA_INDICATION	dl_unit_data_indication
DL_UNIT_H_DATA_INDICATION	dl_unit_h_data_indication
DL_UNIT_DATA_REQUEST	dl_unit_data_request
DL_UNIT_H_DATA_REQUEST	dl_unit_h_data_request
PL_DICE_DATA_INDICATION	pl_dice_data_indication
PL_DICE_DATA_REQUEST	pl_dice_data_request_term
PL_DICE_DATA_REQUEST(SSEQ,BILO,C,IBT)	pl_dice_data_request_orig
PL_DICE_IDLE_INDICATION(BILO,C,IBT)	pl_dice_idle_indication
PL_DICE_IDLE_REQUEST(C,IBT)	pl_dice_idle_request

C.4 Counters in APSL Description

Table C.5 lists all the counters used in the APSL specification of the DICE Protocol. It includes a brief description of each:

Table C.5

Counter name	What it counts
bit_cnt_term	Bits already right shifted in 'out_octet' by the serial_to_parallel converter at the terminating end
c_count_orig	Octets copied in subrate data at the originating end
c_count_term	Octets copied in subrate data at the terminating end
data_bit_count	Bits in the data collector,
fail_idle_cnt	Consecutive identical failure codes arriving for the channelized side
idle_lat_cnt	Consecutive identical idle codes arriving for the channelized side
nbits_orig	Bits received in the series to parallel converter of the originating end
nbits_term	Bits processed in the high layer entity of the terminating end
ndata_bits	Data bits in the DICE information field as returned by count_bits
norm_idle_cnt	Consecutive identical normal idle codes received from the channelized side
octet_cnt_orig	Octets in the DICE information field at the originating end

C.5 Internal Variables

Table C.6 lists internal variables used in the APSL description:

Table C.6

Variable	Description
arriving_bit	Bit peeled from playout_octet before bit stuffing
bilo_orig	Bit in the last octet at the originating endpoint
bilo_term	Bit in the last octet at the terminating endpoint
bit	Bit sent from the parallel_to_serial_converter to the data collector
c_bit_orig	= 0 when control bits have been removed at originating endpoint; = 1 otherwise
c_bit_term	= 0 when control bits have been removed at terminating endpoint; = 1 otherwise
current_time	Current time as determined by a universal clock
data_octet	Octet of data bits formed in the data collector (bit 1 is the least significant bit arithmetically, while bit 8 is the most significant bit)
delay	Time span measured by the t_idle timer
delay_interm	Delay in the intermediate node
delay_orig	Delay in the originating end
eq_orig	Value of the delay equalization bit in the packet to be transmitted
eq_term	Value of the delay equalization bit in the packet received
first_bit	First bit to be retained from the input octet
have_room	= TRUE when the packet can be stored; = FALSE otherwise
ibt_last	Value of the last ibt that corresponds to the entries of Table 3
ibt_idle	The most recently updated value of the ibt code to
ibt_orig	Value of the ibt field at the originating end, be put in a DICE packet
ibt_term	Value of the ibt field at the terminating end
idle_code	Idle code to be put in the output channelized stream
idle_orig	= TRUE when the input octet corresponds to an idle code in Table 1; = FALSE otherwise
idle_term	= TRUE when the IBT of the arriving packet corresponds to a value in Table 3; = FALSE otherwise
input_octet	Octet arriving from the channelized side with the first bit arriving from the channelized side (the most significant bit) being in bit 1, i.e., the subrate framing bit is in bit 1 of input_octet and the control bit is in bit 8 (bit 8 is the least significant bit)
last_bit	Last bit to be retained from the input octet
multipoint	= 1 when the multipoint option is used, = 0 otherwise
normal_idle	= TRUE when the input octet corresponds to a normal idle code in Table 1, = FALSE otherwise

(Continued)

Table C.6 (concluded)

Variable	Description
octet	Copy of input_octet for the idle code detector with bit 1 being the most significant bit and bit 8 the least significant bit
old_idle	Last received idle code from the channelized side
out_bit	Bit peeled from playout_octet
out_octet	Octet output by the serial_to_parallel converter at the terminating end with the bits arranged as in Tables 1 and 3 so that bit 1 of the channelized pattern is the most significant bit of out_octet while bit 8 is the least significant bit
pd	Protocol Discriminator
pkt_length	Length in octets of the DICE information field in the received frame
playout_octet	Octet to be converted from parallel to serial representation by the high layer entity of the terminating end
playout_time	Time scheduled for playout of a received packet
q_empty	=TRUE when playout queue is empty; = FALSE otherwise
rseq_term	RSEQ variable at the terminating end
send_idle	= TRUE when the circuit is provisioned to send idle update packets; = FALSE otherwise
remove_control	= TRUE when the pre_processor is to remove the control bit; = FALSE otherwise
seq_orig	SSEQ variable at the originating end
seq_term	SSEQ variable in the received packet
sig_class	Signal class
speedbit	Rate
ts	Time Stamp (updated by the ts_update function)
uih	= TRUE when the frames used are UIH; = FALSE when the frames used are UI

C.6 Coordination Messages

Table C.7 lists the messages used in the APSL description to coordinate between various originating end processes:

Table C.7

Origin	Message	Destination
data_collector	data_in last_data_in ready	hi_layer_orig " parallel_to_serial_conv
hi_layer_orig	ready pl_dice_data_request_orig pl_dice_idle_request start stop	data_collector xmtr_dice " t_idle "
hi_layer_term	bit_in dice_data idle pl_dice_data_request_term	serial_to_parallel_conv " " playout_dice
idle_code_detector	error ready start "	mgmt_orig pre_processor parallel_to_serial_conv "
interm_pt_dice_relay	dl_pvp_data_request dl_pvp_h_data_request start stop	layer2_interm " tv_delay_interm "
parallel_to_serial_conv	bit_in ready start stop	data_collector pre_processor data_collector "
playout_dice	pl_dice_data_indication pl_dice_idle_indication read_pkt scan	hi_layer_term " rcv_queue "
pre_processor	data_in " start " " stop " "	idle_code_detector parallel_to_serial_conv hi_layer_orig idle_code_detector t_idle hi_layer_orig idle_code_detector t_idle

(Continued)

Table C.7 (Concluded)

Origin	Message	Destination
rcv_queue	queue_empty pkt_in	playout_dice "
rcv_dice	write_pkt	rcv_queue
serial_to_parallel_conv	octet_in ready	user_term hi_layer_term
t_idle	expired	hi_layer_orig
user_orig user_orig user_orig	input_in data_in start " stop "	pre-processor idle_code_detector pre-processor idle_code_detector pre-processor idle_code_detector
user_term	ready start stop	serial_to_parallel_conv " "
xmtr_dice	dl_unit_data_request dl_unit_h_data_request start stop	layer2_orig " tv_delay_orig "

```
/*
        various definitions
                        */

#define BUILDOUT 100
#define CMAX 5
#define DIGITAL_DATA 3
#define FAIL_IDLE_MAX 500
#define FALSE 0
#define IDLE_LAT_MAX 2
#define IDLE_LENGTH 5
#define IDLE_VALUE 60
#define NMAX 133
#define NORM_IDLE_MAX 3
#define PVP 0x44
#define TRUE  1
#define VMAX 133
/*
                arriving_bit    = bit peeled from playout_octet before bit stuffing,
                bilo_orig       = bit in the last octet at the originating endpoint,
                bilo_term       = bit in the last octet at the terminating endpoint,
                bit             = bit sent from the parallel_to_serial_converter to the data collector
                bit_cnt_term    = counter of bits already right shifted in the
                                  serial_to_parallel converter at  the terminating end
                c_bit_orig      = 0 when control bits have been removed at originating endpoint,
                                = 1 otherwise;
                c_bit_term      = 0 when control bits have been removed at terminating endpoint,
                                = 1 otherwise;
                c_count_orig    = counter for the number of octets copied in subrate data at the originating end
                c_count_term    = counter for the number of octets copied in subrate data at the terminating enc
                current_time    current time as determined by a universal clock
                data_bit_count  = counter for bits in the data collector,
                data_octet      = octet of data bits formed in  the data collector,
                delay           = time span measured by the t_idle timer,
                delay_interm    = delay in the intermediate node,
                delay_orig      = delay in the originating end,
                eq_orig         = value of the delay equalization bit in the packet to be transmitted,
                eq_term                 = value of the delay equalization bit in the packet  received,
                fail_idle_cnt   = counter for the number of consecutive identical failure
                                  idle codes received from the channelized side,
                first_bit = first bit to be retained from the input octet,
                have_room       = TRUE when the packet can be stored,
                                = FALSE otherwise;
                ibt_last = value of the last ibt that corresponds to the
                                  entries of Table 3,
                ibt_idle = the most recently updated value of the ibt code to
                ibt_orig = value of the ibt field at the originating end,
                                  be put in a DICE packet,
                ibt_term        = value of the ibt field at the terminating end
                idle_code       idle code to be put in the output channelized stream
                idle_lat_cnt    = counter for the number of consecutive identical idle codes
                                  received from the channelized side,
                idle_orig       = TRUE when the input octet corresponds to an idle code in Table 1,
                                = FALSE otherwise,
                idle_term       = TRUE when the IBT of the arriving packet  corresponds to a value in Table 3,
```

```
                                = FALSE otherwise,
        input_octet     = octet arriving from the channelized side with the first
                          bit arriving from the channelized side (the most significant bit) being in bit 1,
                          i.e., the subrate framing bit is in bit 1 of input_octet and
                          the control bit is in bit 8,
        last_bit = last bit to be retained from the input octet,
        multipoint      = 1 when the multipoint option is used,
                          = 0 otherwise;
        nbits_orig      = number of bits received in the series to parallel converter of the originating end,
        nbits_term      = number of bits received in the series to parallel converter of the originating end,
        ndata_bits      = number of data bits in the DICE information field as returned by count_bits,
        norm_idle_cnt   = counter for the number of consecutive identical normal
                          idle codes received from the channelized side,
        normal_idle     = TRUE when the input octet corresponds to a normal idle code in Table 1,
                          = FALSE otherwise,
    octet           = a copy of input_octet for the idle code detector,
        octet_cnt_orig  = counter for the number of octets in the DICE information field
                          at the originating end,
        old_idle = last received idle code from the channelized side,
        out_bit         = bit peeled from playout_octet,
        out_octet       = octet output by the serial_to_parallel converter at the terminating end with
                          the bits arranged as in Tables 1 and 3 so that bit 1 of the channelized
                          pattern is the most significant bit of out_octet while bit 8 is
                          the least significant bit,
        pd              Protocol Discriminator
        pkt_length      = length in octets of the DICE information field in the received frame,
        playout_octet   = octet to be converted from parallel to serial representation
                          by the high layer entity of the terminating end,
        playout_time    = time scheduled for playout of a received packet,
        q_empty         = TRUE when playout queue is empty,
                          = FALSE otherwise;
        remove_control = TRUE when the pre_processor is to remove the control bit,
                          = FALSE otherwise,
        rseq_term       RSEQ variable at the terminating end
        send_idle       = TRUE when the circuit is provisioned to send idle update packets,
                          = FALSE otherwise,
        seq_orig        SSEQ variable at the originating end
        seq_term        SSEQ variable in the received packet
        sig_class       signal class
        speed           bit rate
        ts              Time Stamp (updated by the ts_update function)
        uih             = TRUE when the frames used are UIH
                          = FALSE when the frames used are UI
                                                                */

/*
                External Processes
*/
EXTERNAL clock, layer2_orig, layer2_interm, layer2_term, mgmt_orig, user_orig,  user_term;
/*
                Pre-Processor at the Originating End
*/
PROCESS     pre_processor;
CONTEXT     c_bit_orig, c_count_orig, first_bit, input_octet, last_bit, multipoint, octet, remove_control, speed;
```

```
STATES          0-12;
REND  hi_layer_orig!start, hi_layer_orig!stop,
        idle_code_detector!start, idle_code_detector!stop,
        idle_code_detector!data_in, idle_code_detector?ready,
        parallel_to_serial_conv!data_in, parallel_to_serial_conv?ready,
        user_orig?start, user_orig?stop, user_orig?input_in,
        t_idle!start, t_idle!stop;
INITIAL STATE 0;
TRANSITIONS

        /* Figure C.1 (a) */

0:1     WHEN user_orig?start * hi_layer_orig!start * idle_code_detector!start * t_idle!start
           [c_count_orig=0; octet=0; input_octet=0;],
1:0     WHEN user_orig?stop * hi_layer_orig!stop *idle_code_detector!stop * t_idle!stop,
1:2     WHEN user_orig?input_in,
        /*
           The first bit to arrive is the subrate framing bit and the last bit is the control bit;
           input_octet has the subrate framing bit in bit 1 and the control bit in bit 8 as shown in Table 1
           In a LAPD frame, the subrate framing bit must be in octet 1, since according to LAPD, bit 1 is
           transmitted first: i.e., the bit order must be reversed

           octet is a copy of input_octet for the idle code detector
         */
2:3     WHEN {speed >= 56},
3:6     WHEN {speed != 56} [octet=input_octet;first_bit=1; last_bit=8;c_bit_orig=1;],
3:4     WHEN {speed ==56},
4:6     WHEN {multipoint == TRUE} [octet=input_octet;first_bit=1; last_bit=7;c_bit_orig=0;],
4:5     WHEN {multipoint != TRUE},
5:6     WHEN {remove_control != TRUE} [octet=input_octet;first_bit=1;last_bit=8; c_bit_orig=1;],
5:6     WHEN {remove_control == TRUE} [octet=input_octet;first_bit=1;last_bit=7;c_bit_orig=0;],
6:7     WHEN idle_code_detector?ready,
7:8     WHEN parallel_to_serial_conv?ready,
8:1     WHEN idle_code_detector!data_in * parallel_to_serial_conv!data_in,
2:9     WHEN {speed < 56} [c_count_orig ++;],
9:10    WHEN {c_count_orig!=1},
10:1    WHEN {c_count_orig < CMAX},
10:1    WHEN {c_count_orig == CMAX} [c_count_orig=0;],
9:11    WHEN {c_count_orig == 1},


                /* Figure C.1 (b) */

11:12   WHEN {multipoint== FALSE},
12:6    WHEN {remove_control== FALSE} [first_bit=2; last_bit=8;c_bit_orig=1;octet=input_octet;],
12:6    WHEN {remove_control== TRUE} [first_bit=2; last_bit=7;c_bit_orig=0;octet=input_octet;],
11:6    WHEN {multipoint== TRUE} [first_bit=2; last_bit=7;c_bit_orig=0;octet=input_octet;].
/*
                Idle Code Detector
*/

PROCESS         idle_code_detector;
CONTEXT         fail_idle_cnt, idle_lat_cnt,  idle_orig, normal_idle, norm_idle_cnt, octet, old_idle;
STATES          0-11;
REND  pre_processor ?start, pre_processor ?stop,
```

```
        pre_processor?data_in, pre_processor!ready,
        parallel_to_serial_conv!start, parallel_to_serial_conv!stop;

INITIAL STATE 0;
TRANSITIONS

                /* Figure C.2 (a) */

0:1     WHEN pre_processor?start [idle_lat_cnt=0;old_idle=0xFF;norm_idle_cnt=0;fail_idle_cnt=0;],
1:0     WHEN pre_processor?stop,
1:2     WHEN pre_processor?data_in [chk_idle_tbl();],
2:1     WHEN {idle_orig == FALSE} parallel_to_serial_conv!start [idle_lat_cnt=0; norm_idle_cnt=0; fail_idle_cnt=0;],
2:3     WHEN {idle_orig == TRUE},
3:11    WHEN {octet != old_idle} parallel_to_serial_conv!start
        [old_idle = octet; idle_lat_cnt = 1;],
11:1    WHEN {normal_idle == FALSE} [fail_idle_cnt=1; norm_idle_cnt=0;],
11:1    WHEN {normal_idle == TRUE} [fail_idle_cnt=0; norm_idle_cnt=1;],
3:4     WHEN {octet == old_idle} [idle_lat_cnt++;],

                /* Figure C.2 (b) */

4:5     WHEN {idle_lat_cnt < IDLE_LAT_MAX},
4:6     WHEN {idle_lat_cnt >= IDLE_LAT_MAX},
6:5     WHEN {idle_lat_cnt > IDLE_LAT_MAX},
6:5     WHEN {idle_lat_cnt == IDLE_LAT_MAX} [ibt_update();],
5:7     WHEN {normal_idle == FALSE} [fail_idle_cnt++;],
5:8     WHEN {normal_idle == TRUE} [norm_idle_cnt++;],
7:1     WHEN {fail_idle_cnt < FAIL_IDLE_MAX} pre_processor!ready,
7:9     WHEN {fail_idle_cnt >= FAIL_IDLE_MAX},
9:1     WHEN {fail_idle_cnt > FAIL_IDLE_MAX} pre_processor!ready,
9:1     WHEN {fail_idle_cnt == FAIL_IDLE_MAX} pre_processor!ready * parallel_to_serial_conv!stop
        [fail_idle_cnt=0;idle_lat_cnt=0;],
8:1     WHEN {norm_idle_cnt < NORM_IDLE_MAX} pre_processor!ready,
8:10    WHEN {norm_idle_cnt >= NORM_IDLE_MAX},
10:1    WHEN {norm_idle_cnt > NORM_IDLE_MAX} pre_processor!ready,
10:1    WHEN {norm_idle_cnt == NORM_IDLE_MAX} pre_processor!ready * parallel_to_serial_conv!stop
        [norm_idle_cnt=0;idle_lat_cnt=0;].
/*
                Parallel_To_Serial_Conv

*/

PROCESS     parallel_to_serial_conv;
CONTEXT     last_bit, nbits_orig, first_bit, bit, octet ;
STATES      0-4;
REND    data_collector!bit_in, data_collector!start, data_collector!stop,
        data_collector?ready,
        idle_code_detector?start, idle_code_detector?stop,
        pre_processor?data_in, pre_processor!ready;
INITIAL STATE 0;
TRANSITIONS

                /* Figure C.3 */

0:1     WHEN idle_code_detector?start * data_collector!start,
```

```
1:0     WHEN idle_code_detector?stop * data_collector!stop,
1:2     WHEN pre_processor?data_in [nbits_orig=1;],
2:1     WHEN {nbits_orig > last_bit } pre_processor!ready,
2:3     WHEN {nbits_orig <= last_bit },
3:2     WHEN {nbits_orig < first_bit} [nbits_orig++;  octet= octet << 1  ; ],
3:4     WHEN {nbits_orig >= first_bit} [bit = octet&0x80;  octet= octet << 1   ; ],
4:2     WHEN data_collector?ready * data_collector!bit_in [nbits_orig++;].



/*
                Data Collector
*/

PROCESS     data_collector;
CONTEXT bilo_orig,bit, data_bit_count, data_octet;
STATES      0-5;
REND  parallel_to_serial_conv?start, parallel_to_serial_conv?stop, parallel_to_serial_conv?bit_in,
      parallel_to_serial_conv!ready, hi_layer_orig?ready,
      hi_layer_orig!data_in, hi_layer_orig!last_data_in;

INITIAL STATE 0;
TRANSITIONS

                /* Figure C.4 */


0:1     WHEN parallel_to_serial_conv?start [data_bit_count=0; data_octet=0;],
1:3     WHEN parallel_to_serial_conv?stop [bilo_orig= 8-data_bit_count;],
3:4     WHEN {data_bit_count < 8},
4:3     WHEN [data_octet = ( (bit|data_octet) >> 1 ); data_bit_count++;],

3:5     WHEN {data_bit_count == 8} hi_layer_orig?ready * hi_layer_orig!last_data_in,
5:0     WHEN parallel_to_serial_conv!ready,
1:2     WHEN parallel_to_serial_conv?bit_in [data_bit_count++;],
2:1     WHEN {data_bit_count < 8}
        [data_octet = ((bit |data_octet) >> 1  );],
2:3     WHEN parallel_to_serial_conv?stop [bilo_orig= 8-data_bit_count;],
2:1     WHEN {data_bit_count == 8 } hi_layer_orig?ready * hi_layer_orig!data_in * parallel_to_serial_conv!ready
          [data_bit_count = 0; data_octet=0;].

/*
                high_layer_orig
*/

PROCESS     hi_layer_orig;
CONTEXT     bilo_orig, ibt_orig,  octet_cnt_orig, ndata_bits, send_idle, seq_orig;
STATES      0-6;
REND  data_collector?data_in,
      data_collector?last_data_in,
      data_collector!ready,
      pre_processor?start, pre_processor?stop,
      t_idle!start, t_idle?expired, t_idle!stop,
      xmtr_dice!pl_dice_data_request_orig, xmtr_dice!pl_dice_idle_request;
```

```
INITIAL STATE 0;
TRANSITIONS

/*
        PACKETIZE STATE

            /* Figure C.5 */
*/

0:1     WHEN pre_processor?start [seq_orig=0; octet_cnt_orig=0;],
1:0     WHEN pre_processor?stop,
1:4     WHEN data_collector?last_data_in * xmtr_dice!pl_dice_data_request_orig
        * t_idle!start [octet_cnt_orig=0;],
1:2     WHEN data_collector?data_in [octet_cnt_orig++;],
2:1     WHEN {octet_cnt_orig < NMAX} data_collector!ready,
2:3     WHEN {octet_cnt_orig >= NMAX} data_collector!ready [octet_cnt_orig=0; bilo_orig=0;],
3:1     WHEN xmtr_dice!pl_dice_data_request_orig [seq_increment();],


            /* IDLE STATE */
/*
            Figure C.6

*/

4:5     WHEN t_idle?expired,
5:4     WHEN {send_idle == TRUE} xmtr_dice!pl_dice_idle_request * t_idle!start,
5:4     WHEN {send_idle != TRUE} t_idle!start,
4:2     WHEN data_collector?data_in * t_idle!stop [seq_orig=0;octet_cnt_orig=1;].


/*
        Originating End  XMTR -- Figure C.7
                        */

PROCESS     xmtr_dice;
CONTEXT bilo_orig,eq_orig,seq_orig,uih;
STATES      100-104;
REND  hi_layer_orig?pl_dice_idle_request, hi_layer_orig?pl_dice_data_request_orig,
      tvdelay_orig!start, tvdelay_orig!stop,
      layer2_orig?dl_l1_ready_indication,
      layer2_orig!dl_unit_data_request,
      layer2_orig!dl_unit_h_data_request;

INITIAL STATE 100;
TRANSITIONS

100:101     WHEN  hi_layer_orig?pl_dice_data_request_orig,
100:101     WHEN  hi_layer_orig?pl_dice_idle_request [seq_orig =0;eq_orig=0;bilo_orig=0000;],
101:102     WHEN  tvdelay_orig!start,
102:103     WHEN  layer2_orig?dl_l1_ready_indication * tvdelay_orig!stop,
103:104     WHEN  [ts_update();] ,
104:100     WHEN  {uih == FALSE}layer2_orig!dl_unit_data_request,
104:100     WHEN  {uih == TRUE}layer2_orig!dl_unit_h_data_request
```

```
/*
        Intermediate Point -- Figure C.8
                                                */

PROCESS     interm_pt_dice_relay;
CONTEXT pd, uih;
STATES      0-3;
REND  layer2_interm?dl_pvp_data_indication,
        layer2_interm?dl_pvp_h_data_indication,
        layer2_interm!dl_pvp_data_request,
        layer2_interm!dl_pvp_h_data_request,
        layer2_interm?dl_l1_ready_indication,
        tvdelay_interm!start, tvdelay_interm!stop;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  layer2_interm?dl_pvp_h_data_indication [uih = TRUE;],
0:1     WHEN  layer2_interm?dl_pvp_data_indication [uih = FALSE;],
1:0     WHEN  {pd != PVP},
1:2     WHEN  {pd == PVP} tvdelay_interm!start,
2:3     WHEN  layer2_interm?dl_l1_ready_indication * tvdelay_interm!stop [ts_update();],
3:0     WHEN  {uih == FALSE} layer2_interm!dl_pvp_data_request,
3:0     WHEN  {uih == TRUE} layer2_interm!dl_pvp_h_data_request.


/*
        Receiver Terminating End        -- Figure C.9
                                    */
PROCESS     rcv_dice;
CONTEXT pd,sig_class,ts;
STATES      0-4;
REND  layer2_term?dl_unit_h_data_indication,
        layer2_term?dl_unit_data_indication,  rcv_queue!write_pkt;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  layer2_term?dl_unit_h_data_indication,
0:1     WHEN  layer2_term?dl_unit_data_indication,
1:0     WHEN  {pd != PVP},
1:2     WHEN  {pd == PVP},
2:0     WHEN  {sig_class !=  DIGITAL_DATA},
2:3     WHEN  {sig_class ==  DIGITAL_DATA},
3:0     WHEN  {ts > BUILDOUT},
3:4     WHEN  {ts <= BUILDOUT} [set_play_out_time();],
4:0     WHEN  rcv_queue!write_pkt.


/*
        Receiver Queue          -- Figure C.10
                                */
PROCESS     rcv_queue;
CONTEXT have_room,q_empty;
STATES      0-3;
```

```
REND  rcv_dice?write_pkt, playout_dice?scan,
      playout_dice!pkt_in, playout_dice!queue_empty,
      playout_dice?read_pkt;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  rcv_dice?write_pkt,
1:0     WHEN  {have_room == TRUE} [push_pkt();],
1:0     WHEN  {have_room != TRUE} [drop_pkt();],
0:2     WHEN  playout_dice?scan,
2:0     WHEN  {q_empty == TRUE} playout_dice!queue_empty,
2:0     WHEN  {q_empty != TRUE} playout_dice!pkt_in,
0:3     WHEN  playout_dice?read_pkt,
3:0     WHEN  [set_pkt_length();pop_pkt();].



/*
      Playout DICE --  Figure C.11 (a)
                                    */
PROCESS     playout_dice;
CONTEXT current_time, eq_term, ibt_last, ibt_term, idle_term, pkt_length, playout_time, rseq_term, seq_term;
STATES      0-4,10-14,20-25;
REND  hi_layer_term?pl_dice_data_request_term,
      hi_layer_term!pl_dice_idle_indication,
      hi_layer_term!pl_dice_data_indication,
      rcv_queue!scan, rcv_queue?pkt_in,rcv_queue?queue_empty,
      rcv_queue!read_pkt,
      user_term?start, user_term?stop, clock?time;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  user_term?start [rseq_term=0;ibt_last=0xf;],
1:0     WHEN  user_term?stop,
1:2     WHEN  hi_layer_term?pl_dice_data_request_term,
2:3     WHEN  rcv_queue!scan,
3:1     WHEN  rcv_queue?queue_empty * hi_layer_term!pl_dice_idle_indication,
3:4     WHEN  rcv_queue?pkt_in * rcv_queue!read_pkt,
4:10    WHEN  {seq_term == 0},
4:20    WHEN  {seq_term>0},

/*
      -- Figure C.11 (b)

*/
10:11   WHEN  clock?time,
11:1    WHEN  {current_time >playout_time},
11:12   WHEN  {current_time <=playout_time},
12:11   WHEN  {current_time <playout_time} clock?time,
12:13   WHEN  {current_time == playout_time },
13:14   WHEN  {pkt_length != 0} hi_layer_term!pl_dice_data_indication [rseq_term=1; get_idle_octet();],
13:14   WHEN  {pkt_length == 0} hi_layer_term!pl_dice_idle_indication [get_idle_octet();],
14:1    WHEN  {idle_term == FALSE},
```

```
14:1    WHEN {idle_term == TRUE} [ibt_last=ibt_term;],

/*
        -- Figure C.11 (c)

        Non-zero sequence number
                        */
20:25   WHEN {seq_term == rseq_term} hi_layer_term!pl_dice_data_indication [rseq_increment();get_idle_octet(;
20:21   WHEN {seq_term != rseq_term} hi_layer_term!pl_dice_idle_indication [rseq_term = seq_term;],
21:22   WHEN clock?time,
22:1    WHEN {current_time > playout_time },
22:23   WHEN {current_time <= playout_time },
23:21   WHEN {current_time < playout_time } clock?time,
23:24   WHEN {current_time == playout_time },
24:25   WHEN {pkt_length != 0} hi_layer_term!pl_dice_data_indication
                [rseq_increment(); get_idle_octet();],
24:25   WHEN {pkt_length == 0} hi_layer_term!pl_dice_idle_indication [rseq_term=0;get_idle_octet();],
25:1    WHEN {idle_term == FALSE},
25:1    WHEN {idle_term == TRUE} [ibt_last=ibt_term;].

/*
        High layer entity at the terminating end

        -- Figure C.12 (a)
                        */

PROCESS     hi_layer_term;
CONTEXT     bilo_term, pkt_length, ibt_term, nbits_term,
        playout_octet, out_bit, arriving_bit ;
STATES      0-3,10-14;
REND    playout_dice?pl_dice_idle_indication, playout_dice?pl_dice_data_indication,
        playout_dice!pl_dice_data_request_term,
        user_term?start, user_term?stop,
        serial_to_parallel_conv?ready,
        serial_to_parallel_conv!idle,
        serial_to_parallel_conv!dice_data,
        serial_to_parallel_conv!bit_in;
INITIAL STATE 0;
TRANSITIONS

0:1     WHEN user_term?start,
1:0     WHEN user_term?stop,
1:2     WHEN playout_dice!pl_dice_data_request_term,
2:10    WHEN playout_dice?pl_dice_data_indication * serial_to_parallel_conv!dice_data
            [nbits_term =0;],
2:3     WHEN playout_dice?pl_dice_idle_indication,
3:1     WHEN serial_to_parallel_conv?ready * serial_to_parallel_conv!idle,
/*
        -- Figure C.12 (b)

      PLAY_DICE_DATA STATE

      Send data octet bit by bit to the serial to parallel converter
*/
10:13   WHEN {pkt_length == 1 } [nbits_term=8-bilo_term;],
```

```
13:14   WHEN {nbits_term > 0} [out_bit = playout_octet %2;
                playout_octet = playout_octet/2;],
13:1    WHEN {nbits_term <= 0},
14:13   WHEN  serial_to_parallel_conv?ready *
            serial_to_parallel_conv!bit_in [nbits_term--;],
10:11   WHEN {pkt_length != 1 } [get_octet(); pkt_length--;nbits_term=0;],
11:12   WHEN {nbits_term < 8} [out_bit = playout_octet %2;
                playout_octet = playout_octet/2;],
12:11   WHEN  serial_to_parallel_conv?ready *
            serial_to_parallel_conv!bit_in [nbits_term++;],
11:10 WHEN {nbits_term >= 8}.

/*
        -- Figure C.13 (a)
                                        */

PROCESS       serial_to_parallel_conv;
CONTEXT bit_cnt_term, c_bit_term, c_count_term , ibt_term,
        out_octet, speed;
STATES        0-5,9-15,20-22,29-33,40-42,50-53,60-62,100-104,300-303;
REND  hi_layer_term?bit_in,
        hi_layer_term?dice_data,
        hi_layer_term?idle,
        hi_layer_term!ready,
        user_term!octet_in,
        user_term?ready,
        user_term?start,
        user_term?stop;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN user_term?start * hi_layer_term!ready [bit_cnt_term=0;c_count_term=0;],
1:2     WHEN {speed < 56},
2:0     WHEN user_term?stop,
2:9     WHEN hi_layer_term?dice_data [out_octet=0;],
9:10    WHEN {c_bit_term ==0},
9:100   WHEN {c_bit_term !=0},
2:20    WHEN hi_layer_term?idle,
1:3     WHEN {speed >= 56},
3:0     WHEN user_term?stop,
3:4     WHEN {speed == 56},
4:29    WHEN hi_layer_term?dice_data [out_octet=0;],
29:30   WHEN {c_bit_term ==0},
29:300  WHEN {c_bit_term !=0},
4:40    WHEN hi_layer_term?idle,
3:5     WHEN {speed != 56},
5:0     WHEN user_term?stop,
5:50    WHEN hi_layer_term?dice_data [out_octet=0;],
5:60    WHEN hi_layer_term?idle,

/*
        -- Figure C.13 (b)
```

DICE operation at Speed less than 56 kbit/s

```
    with the control bit to be added
                                    */

10:11   WHEN hi_layer_term?bit_in [bit_cnt_term++;],
10:20   WHEN hi_layer_term?idle,
10:0    WHEN user_term?stop,
11:12   WHEN {bit_cnt_term ==1 } [out_octet=1; bit_cnt_term++;],
11:12   WHEN {bit_cnt_term != 1 },
12:10   WHEN {bit_cnt_term < 7} hi_layer_term!ready
            [out_octet = 2 * out_octet +out_bit;],
12:13   WHEN {bit_cnt_term >= 7 } [out_octet = 2 * out_octet +out_bit;],
13:14   WHEN [out_octet = 2 * out_octet +1;],
14:15   WHEN user_term?ready * user_term!octet_in [c_count_term++;],
15:14   WHEN {c_count_term <= CMAX },
15:9    WHEN {c_count_term > CMAX } hi_layer_term!ready [bit_cnt_term=0; out_octet=0; c_count_term =0;],

/*
        -- Figure C.13 (c)

    Idle Operation at less than 56 kbit/s
                                    */

20:0    WHEN user_term?stop,
20:21   WHEN [get_last_ibt_octet();],
21:22   WHEN user_term?ready * user_term!octet_in [c_count_term++;],
22:21   WHEN {c_count_term <= CMAX },
22:2    WHEN {c_count_term > CMAX } hi_layer_term!ready [c_count_term =0;],

/*
        -- Figure C.13 (d)

    DICE operation at Speed  of 56 kbit/s
    with the control bit to be added
*/

30:0    WHEN user_term?stop,
30:31   WHEN hi_layer_term?bit_in [bit_cnt_term++;],
30:40   WHEN hi_layer_term?idle,
31:30   WHEN {bit_cnt_term <= 6} hi_layer_term!ready
          [out_octet=2*out_octet+ out_bit ;],
31:32   WHEN {bit_cnt_term > 6 } [out_octet = 4 * out_octet + 2 * out_bit+1;],
32:33   WHEN user_term?ready * user_term!octet_in,
33:29   WHEN hi_layer_term!ready [bit_cnt_term=0; out_octet=0;],


/*
        -- Figure C.13 (e)

    Idle Operation at 56 kbit/s

                                    */

40:0    WHEN user_term?stop,
40:41   WHEN [get_last_ibt_octet();],
41:42   WHEN user_term?ready * user_term!octet_in,
```

```
42:4    WHEN hi_layer_term!ready [out_octet=0;],

/*
        -- Figure C.13 (f)

    DICE Operation  for 64 kbit/s
                                    */

50:0    WHEN user_term?stop,
50:60   WHEN hi_layer_term?idle,
50:51   WHEN hi_layer_term?bit_in [bit_cnt_term++;],
51:50   WHEN {bit_cnt_term <= 7} hi_layer_term!ready
          [out_octet=2*out_octet+ out_bit ;],
51:52   WHEN {bit_cnt_term > 7 } [out_octet=2*out_octet+out_bit;],
52:53   WHEN user_term?ready * user_term!octet_in,
53:50   WHEN hi_layer_term!ready [bit_cnt_term=0; out_octet=0;],

/*
        -- Figure C.13 (g)

    Idle Operation at 64 kbit/s
                                    */

60:0    WHEN user_term?stop,
60:61   WHEN [get_last_ibt_octet();],
61:62   WHEN user_term?ready * user_term!octet_in,
62:5    WHEN hi_layer_term!ready [out_octet=0;],
/*
        -- Figure C.13 (h)

    DICE operation at Speed less than 56 kbit/s
    with the control bit not to be added
*/
100:0   WHEN user_term?stop,
100:101         WHEN hi_layer_term?bit_in [bit_cnt_term++;],
100:20  WHEN hi_layer_term?idle,
101:102         WHEN {bit_cnt_term ==1 } [out_octet=1; bit_cnt_term++;],
101:102         WHEN {bit_cnt_term != 1 },
102:100         WHEN {bit_cnt_term <= 7} hi_layer_term!ready
          [out_octet = 2 * out_octet +out_bit;],
102:103         WHEN {bit_cnt_term > 7 } [out_octet = 2 * out_octet +out_bit;],
103:104         WHEN user_term?ready * user_term!octet_in [c_count_term++;],
104:103         WHEN {c_count_term <= CMAX },
104:9   WHEN {c_count_term > CMAX } hi_layer_term!ready [bit_cnt_term=0; out_octet=0; c_count_term =0;],

/*
        -- Figure C.13 (i)

    DICE operation at Speed less than 56 kbit/s
    with the control bit not to be added
*/
300:0   WHEN user_term?stop,
300:301         WHEN hi_layer_term?bit_in [bit_cnt_term++;],
300:40  WHEN hi_layer_term?idle,
301:300         WHEN {bit_cnt_term <= 7} hi_layer_term!ready
```

```
            [out_octet = 2 * out_octet +out_bit;],
301:302         WHEN {bit_cnt_term > 7 } [out_octet = 2 * out_octet +out_bit;],
302:303         WHEN user_term?ready * user_term!octet_in,
303:29  WHEN hi_layer_term!ready [bit_cnt_term=0; out_octet=0;].

/*
        T_IDLE -- Figure C.14
                                        */
PROCESS     t_idle;
CONTEXT delay;
STATES      0-2;
REND  hi_layer_orig?start, hi_layer_orig!expired, hi_layer_orig?stop,
        pre_processor?start, pre_processor?stop, clock?tick;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  hi_layer_orig?start [delay=0;],
0:1     WHEN  pre_processor?start [delay=0;],
1:2     WHEN  clock?tick  [delay++;],
2:1     WHEN  {delay < IDLE_VALUE} ,
2:0     WHEN  {delay == IDLE_VALUE} hi_layer_orig!expired [delay=0;],
1:0     WHEN  pre_processor?stop [delay=0;],
1:0     WHEN  hi_layer_orig?stop [delay=0;].

/*
        TVDELAY_ORIG -- Figure C.15
                                        */
PROCESS     tvdelay_orig;
CONTEXT delay_orig;
STATES      0-1;
REND  xmtr_dice?start, xmtr_dice?stop, clock?tick;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  xmtr_dice?start [delay_orig=0;],
1:1     WHEN  clock?tick [delay_orig++;],
1:0     WHEN  xmtr_dice?stop[output_delay();delay_orig=0;].

/*
        TVDELAY_INTERM -- Figure C.16
                                        */
PROCESS     tvdelay_interm;
CONTEXT delay_interm;
STATES      0-1;
REND  interm_pt_dice_relay?start, interm_pt_dice_relay?stop,
        clock?tick;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  interm_pt_dice_relay?start [delay_interm=0;],
1:1     WHEN  clock?tick [delay_interm++;],
1:0     WHEN  interm_pt_dice_relay?stop[output_delay();delay_interm=0;].
```

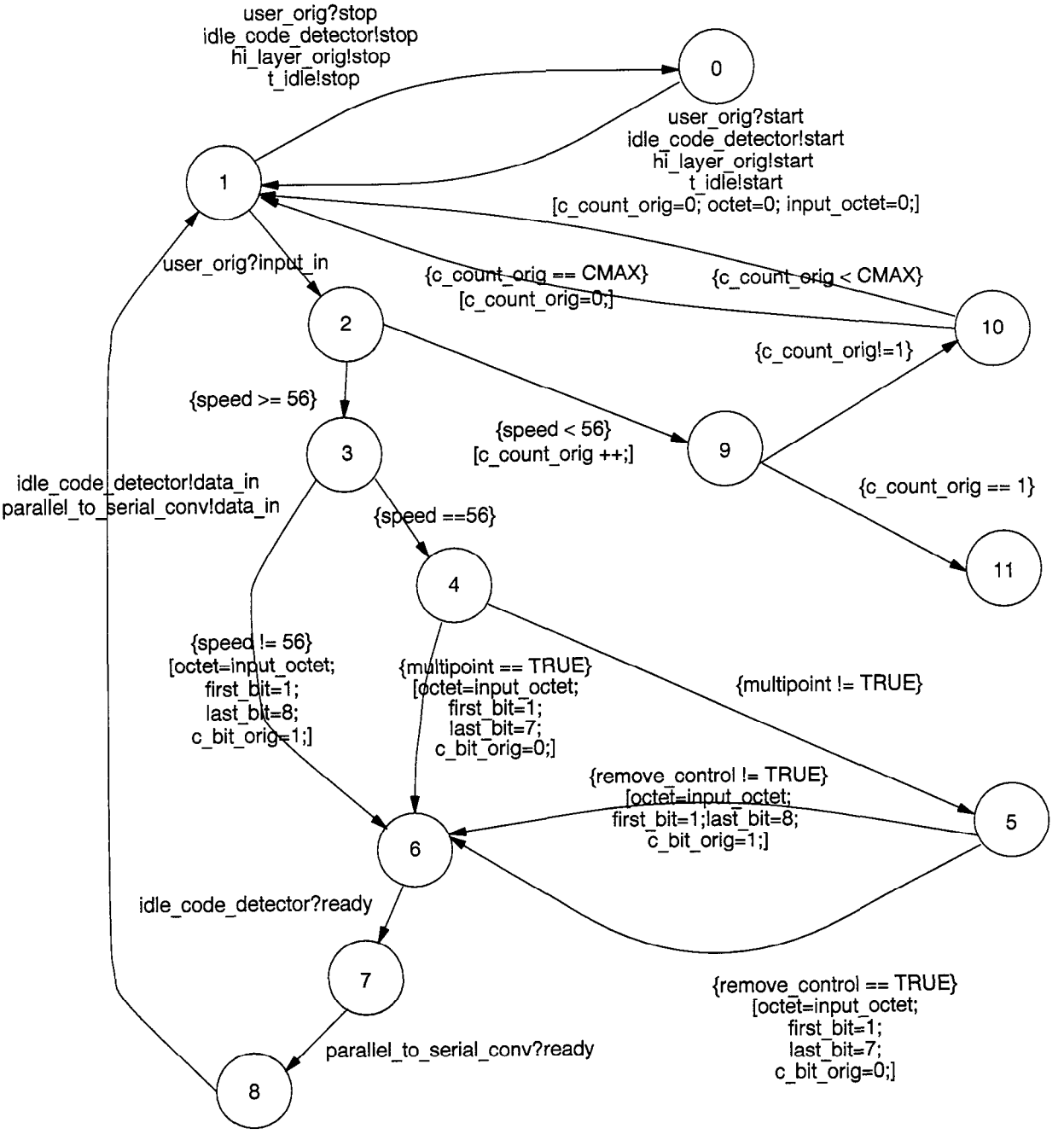


Figure C.1 – pre_processor
Part a

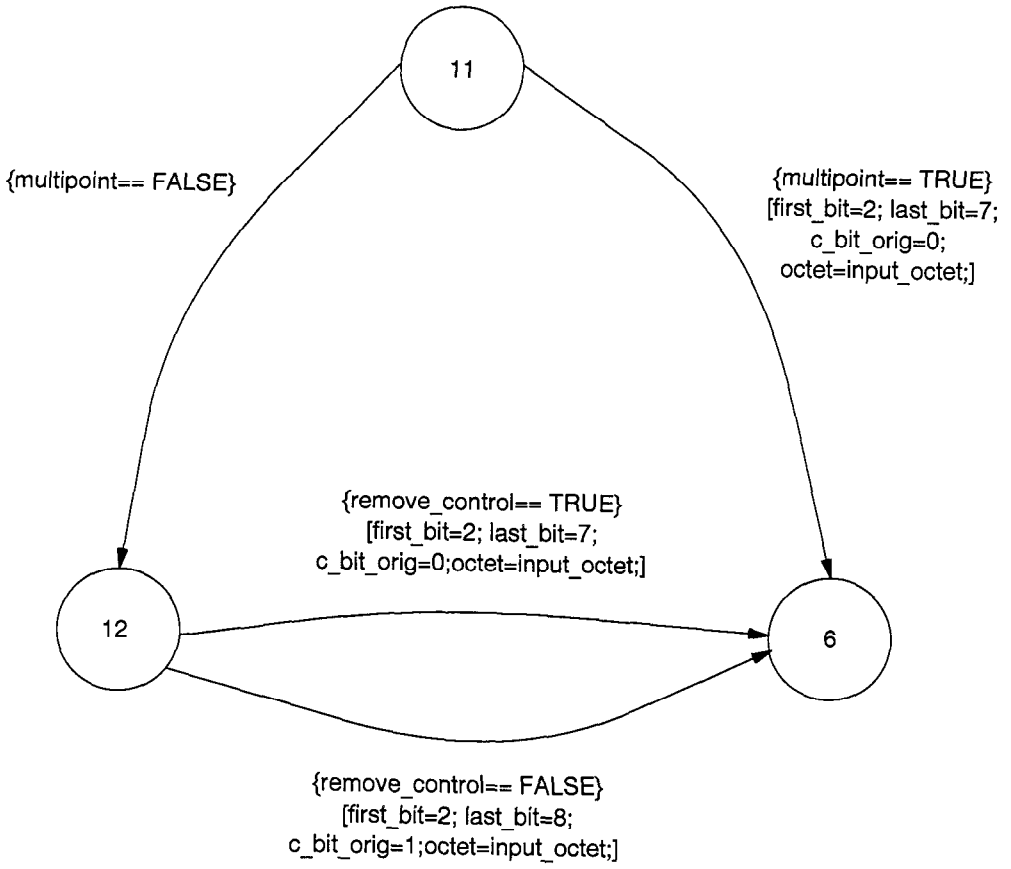


Figure C.1 (concluded) – pre_processor
Part b

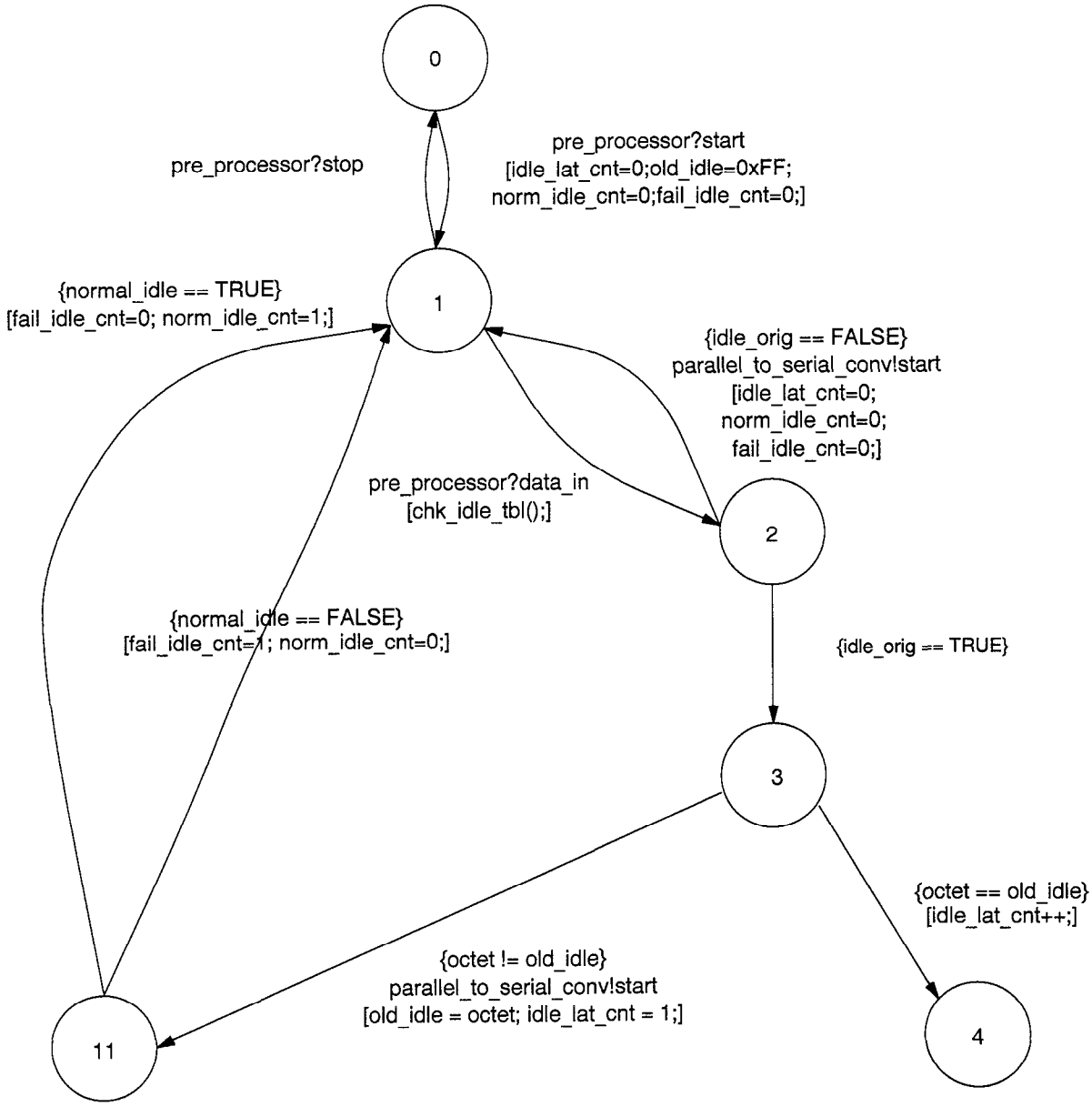


Figure C.2 – idle_code_detector
Part a

Figure C.2 (concluded) – idle_code_detector
Part b

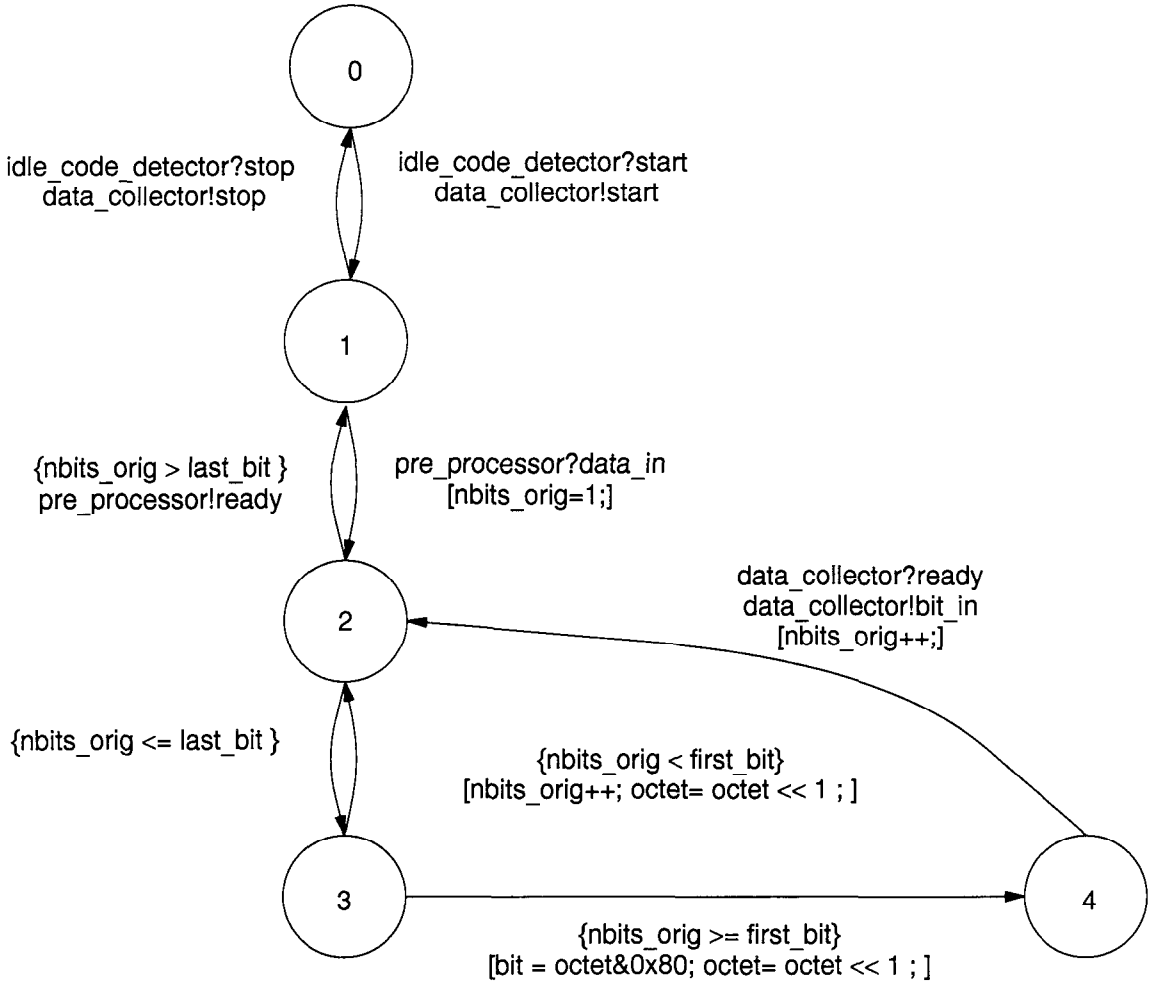


Figure C.3 – parallel_to_serial_converter

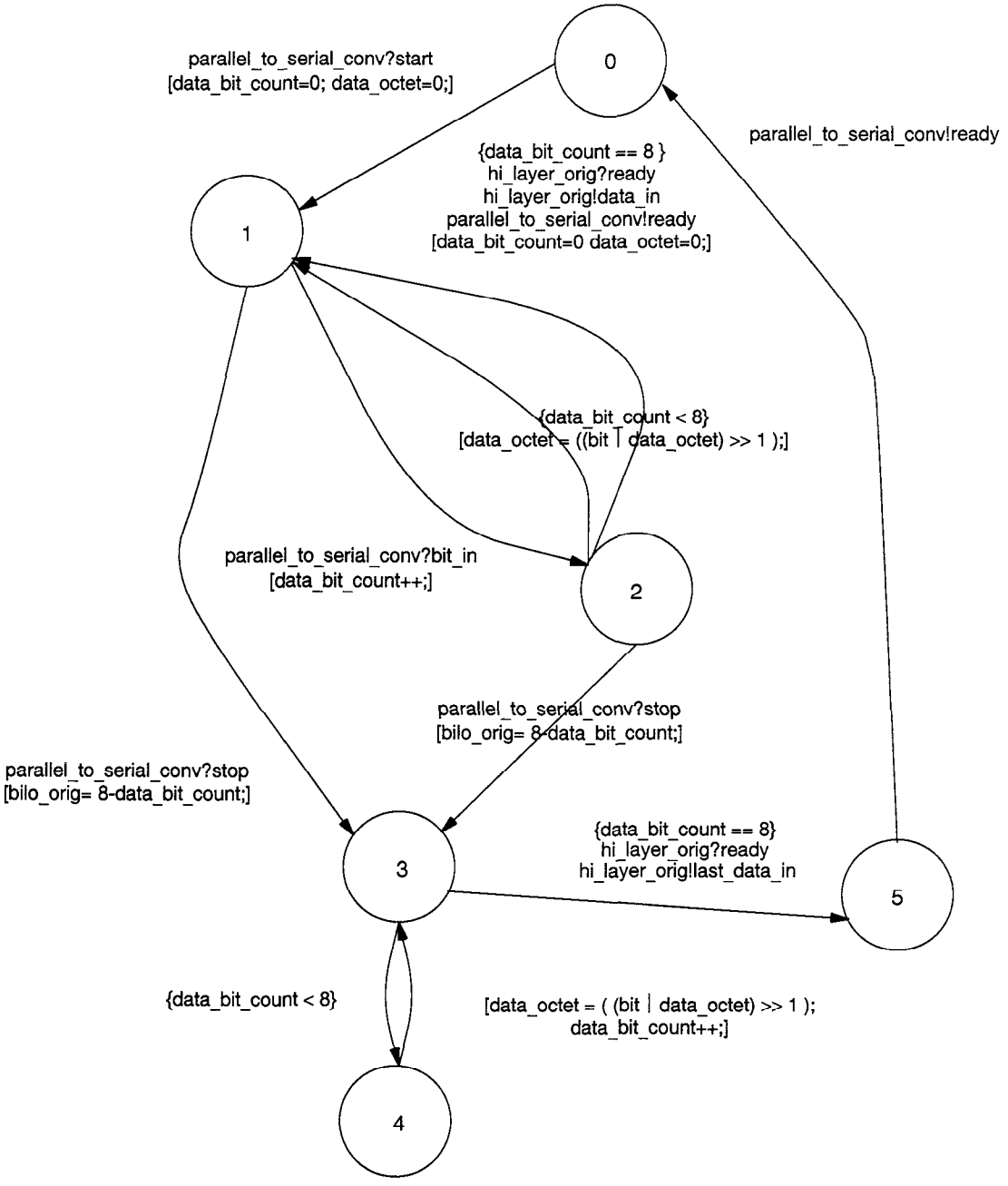


Figure C.4 – data_collector

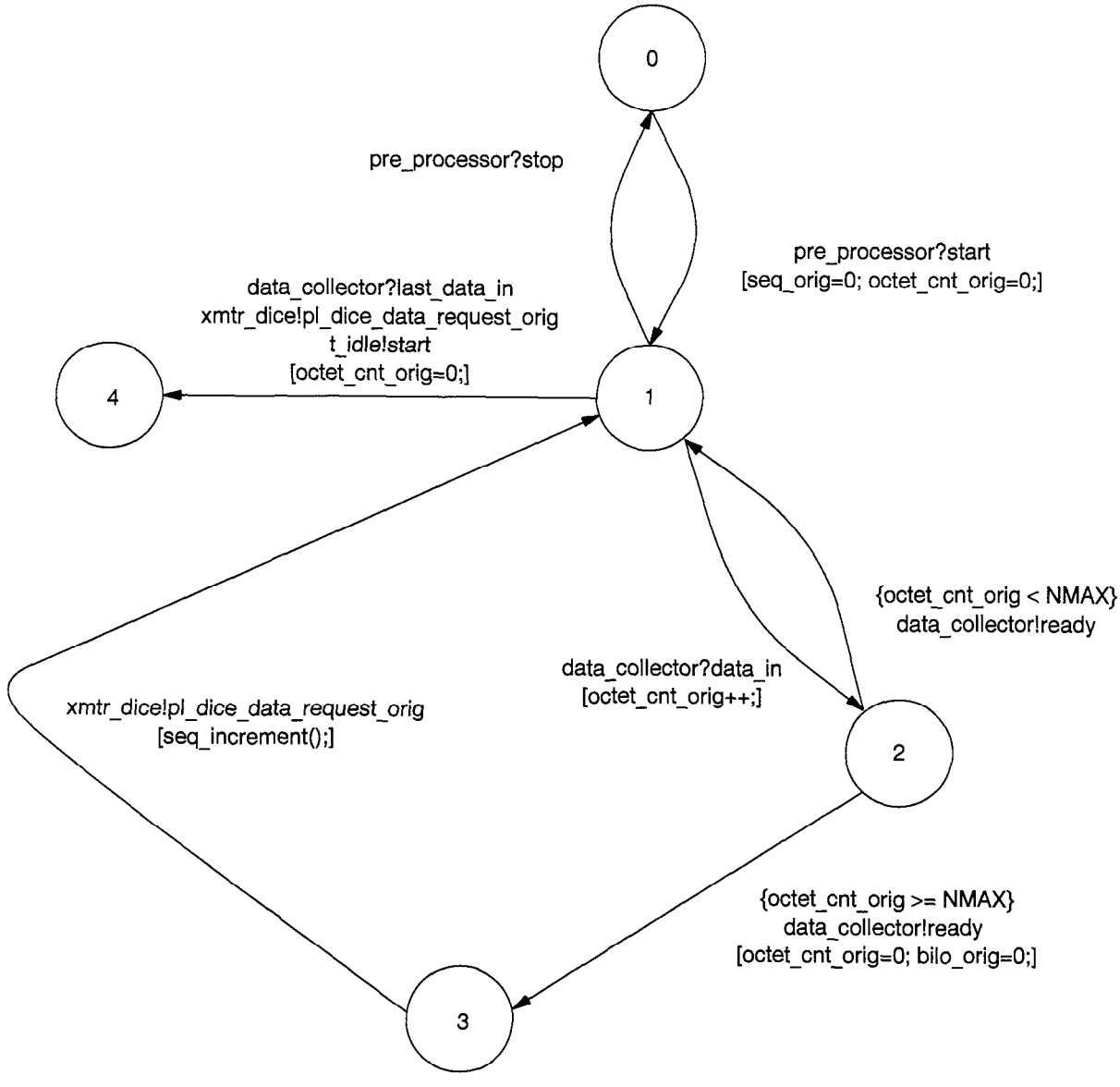


Figure C.5 – hi_layer_orig
PACKETIZE state

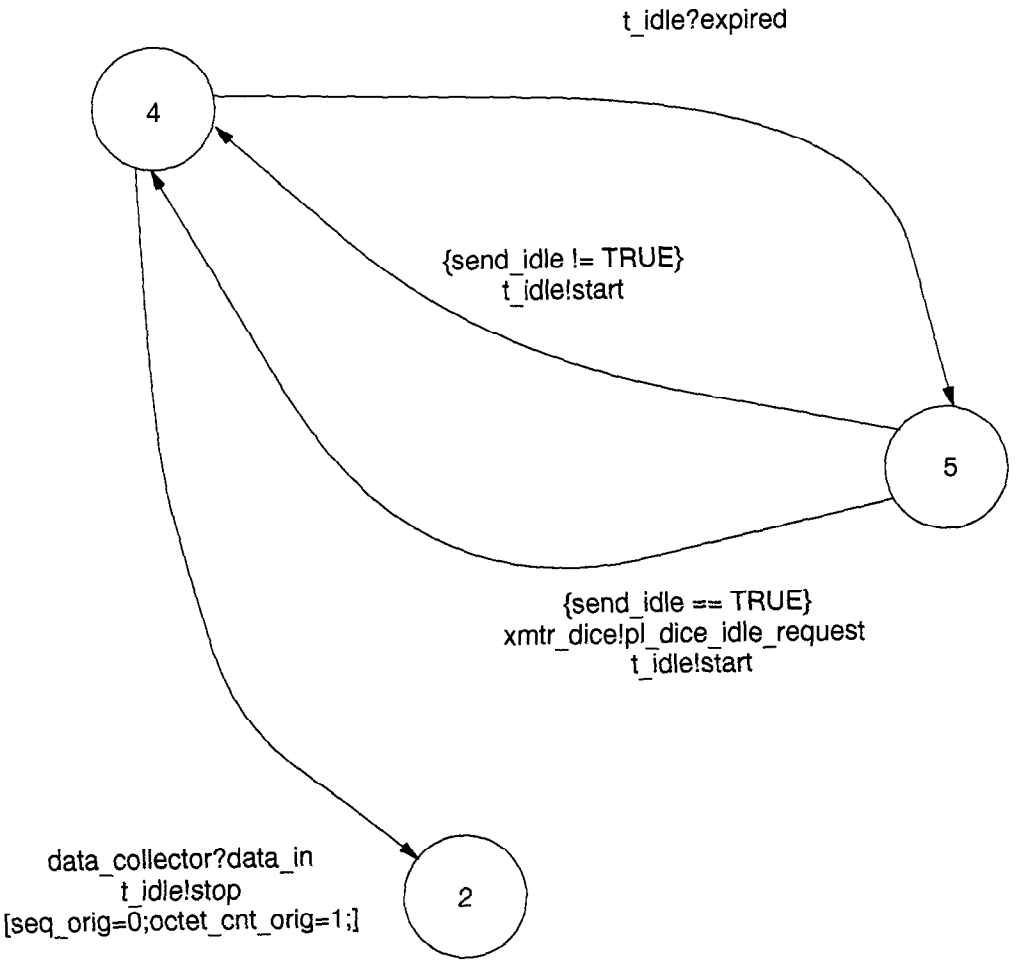


Figure C.6 – hi_layer_orig
IDLE state

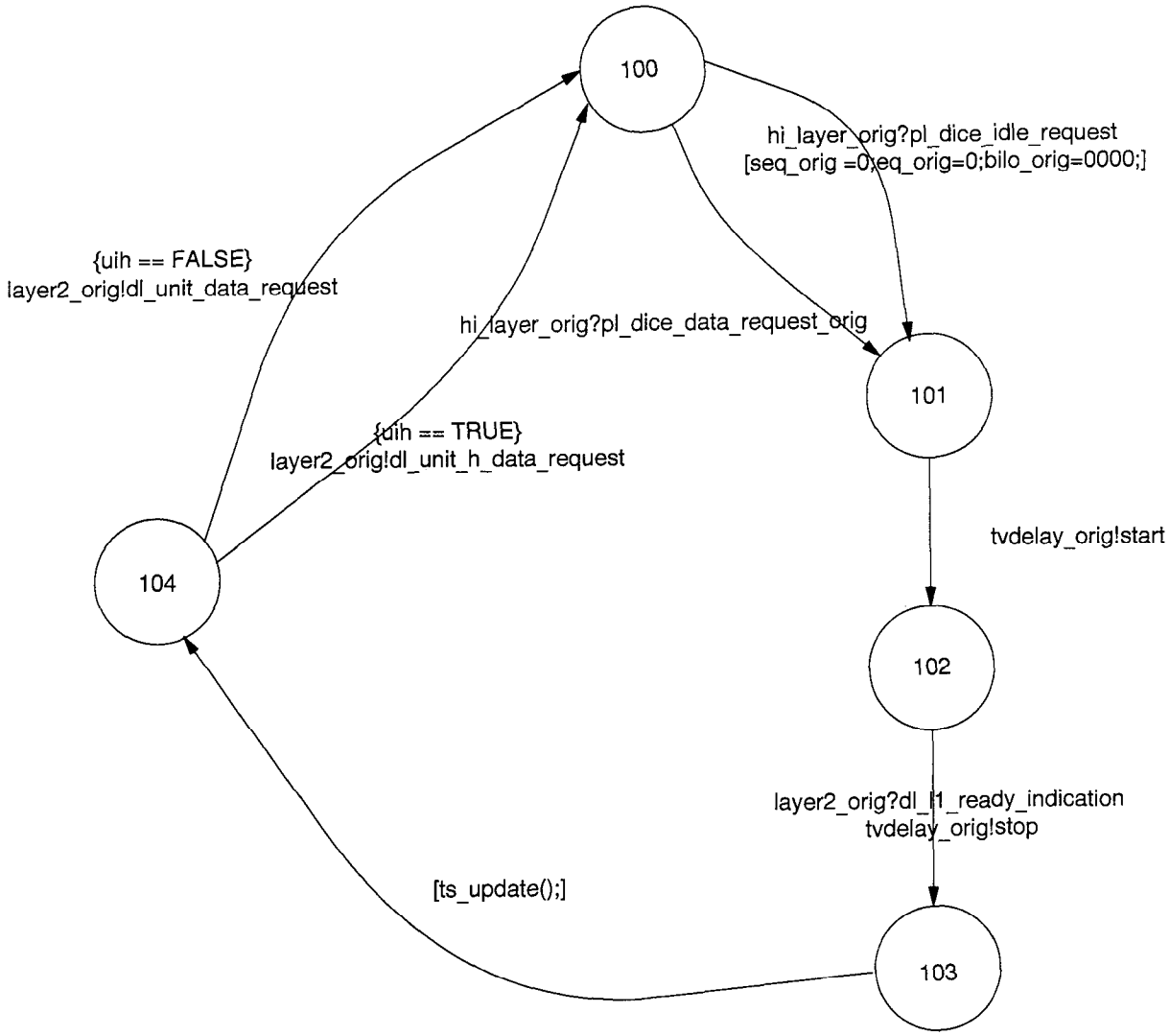


Figure C.7 – xmtr_dice

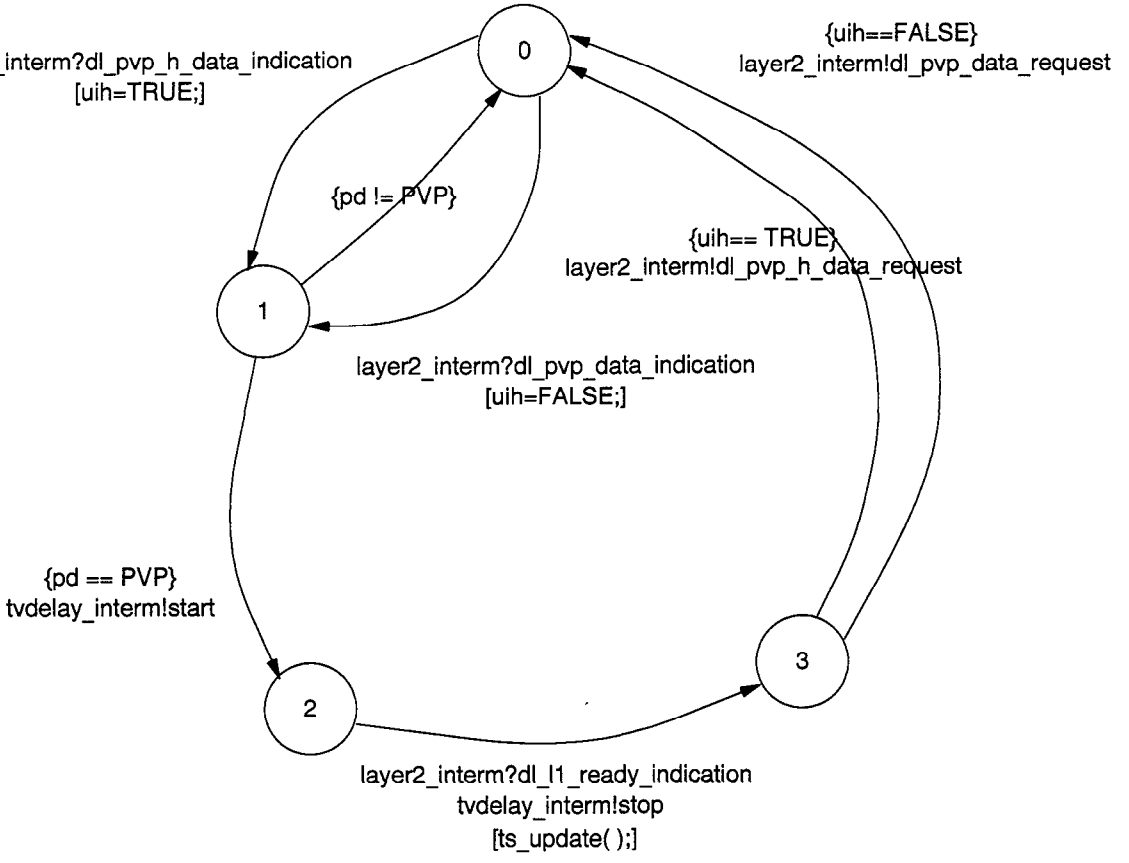


Figure C.8 – interm_pt_dice_relay

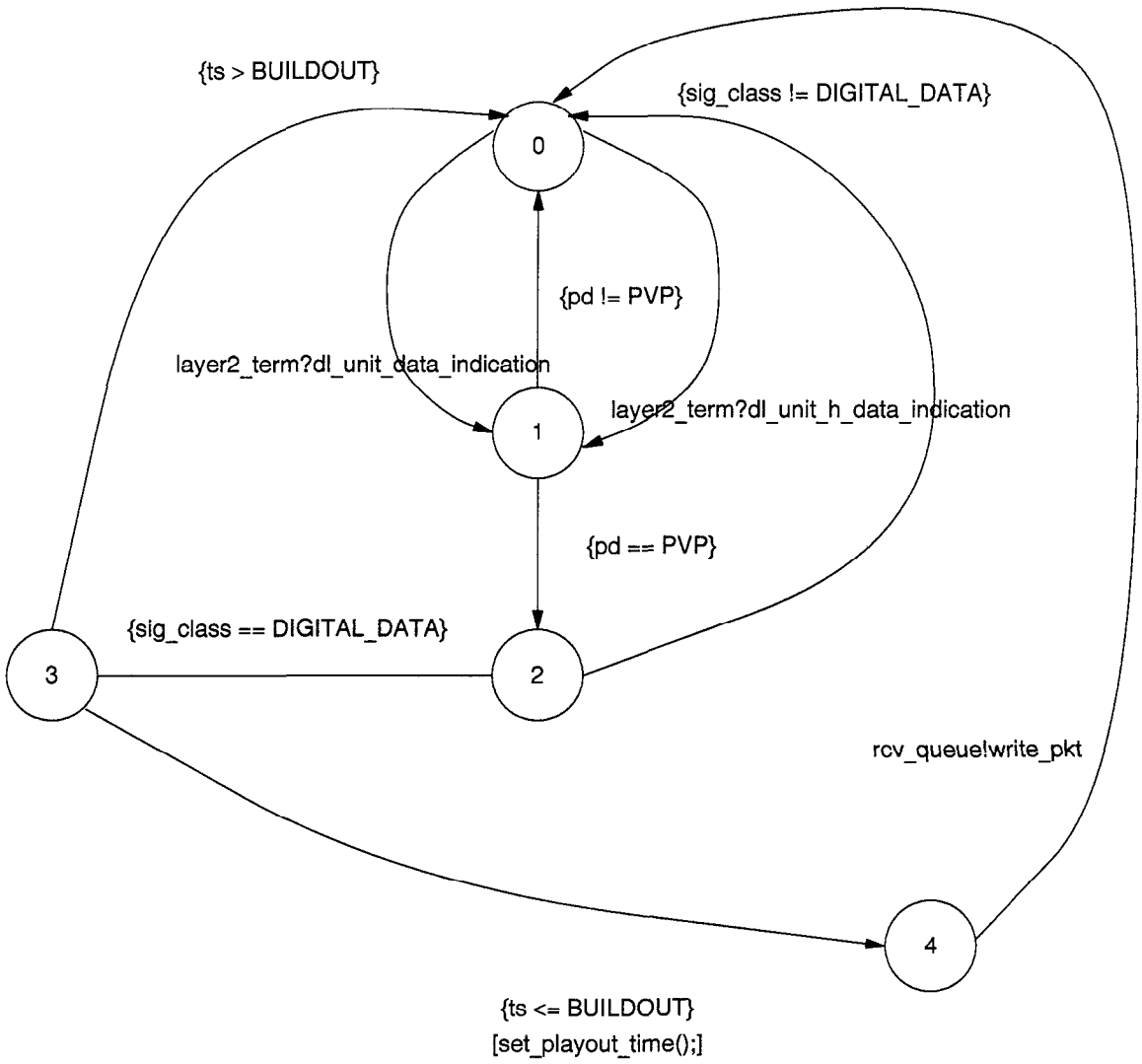


Figure C.9 – rcv_dice

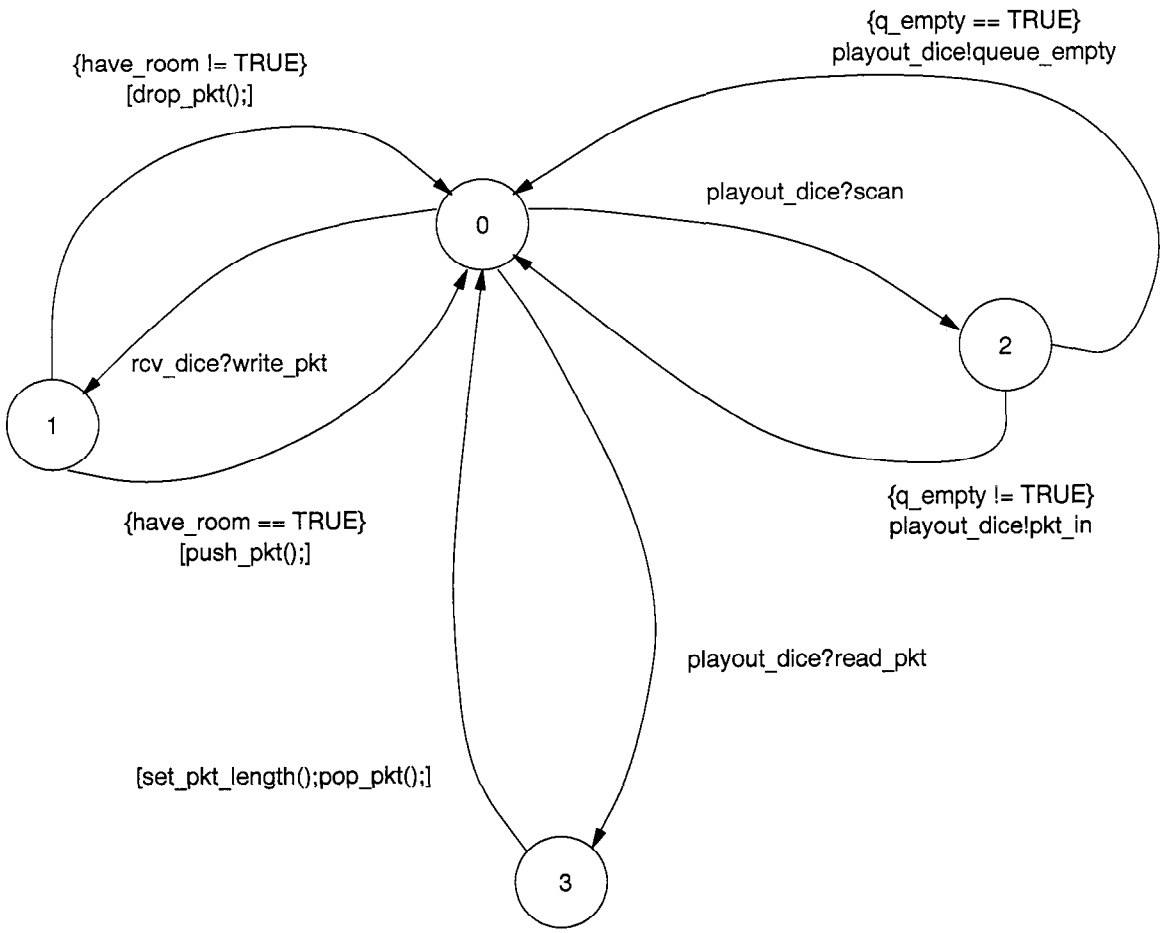


Figure C.10 – rcv_queue

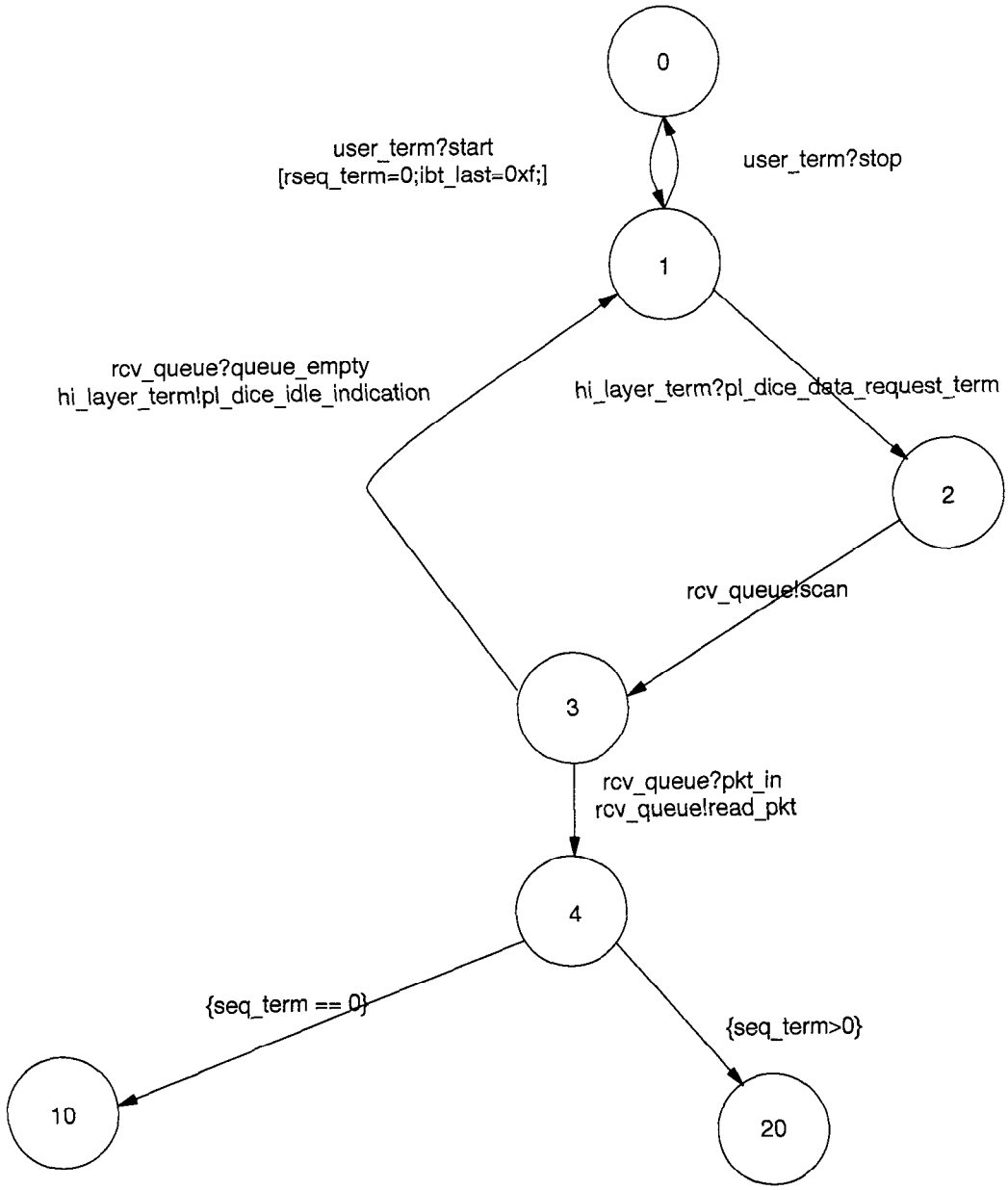


Figure C.11 – playout_dice
Part a

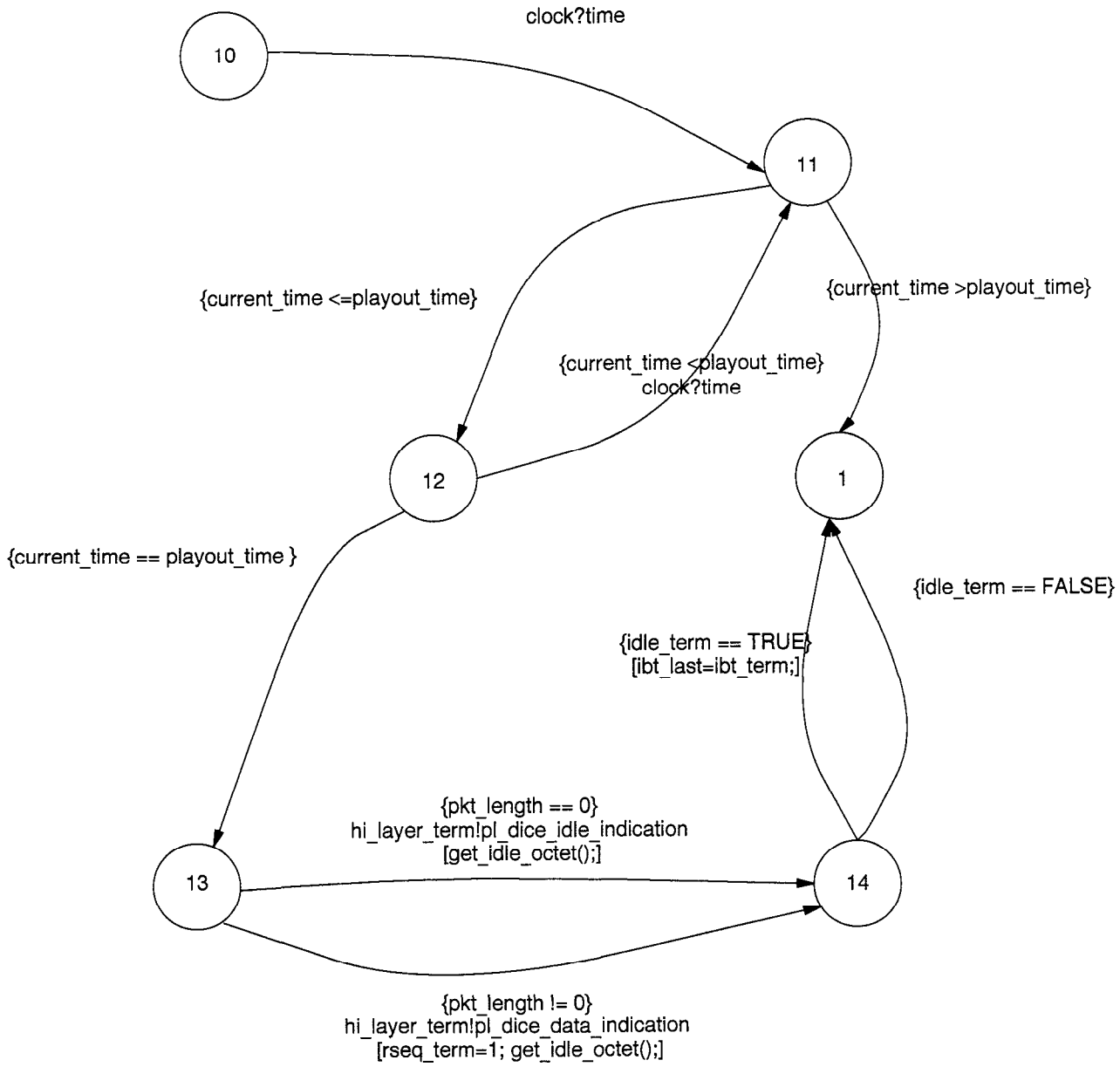


Figure C.11 (continued) – playout_dice
Part b

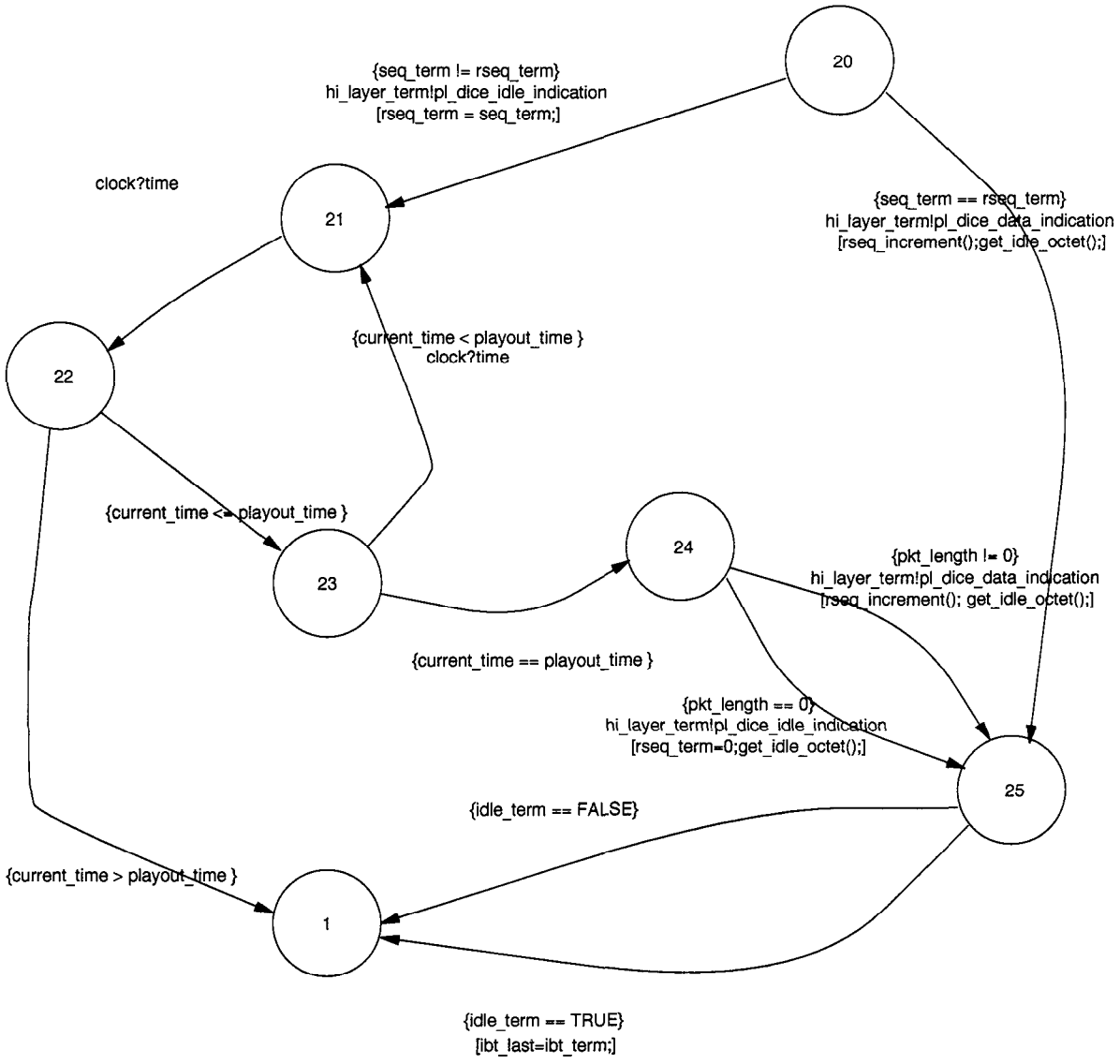


Figure C.11 (concluded) – playout_dice
Part c

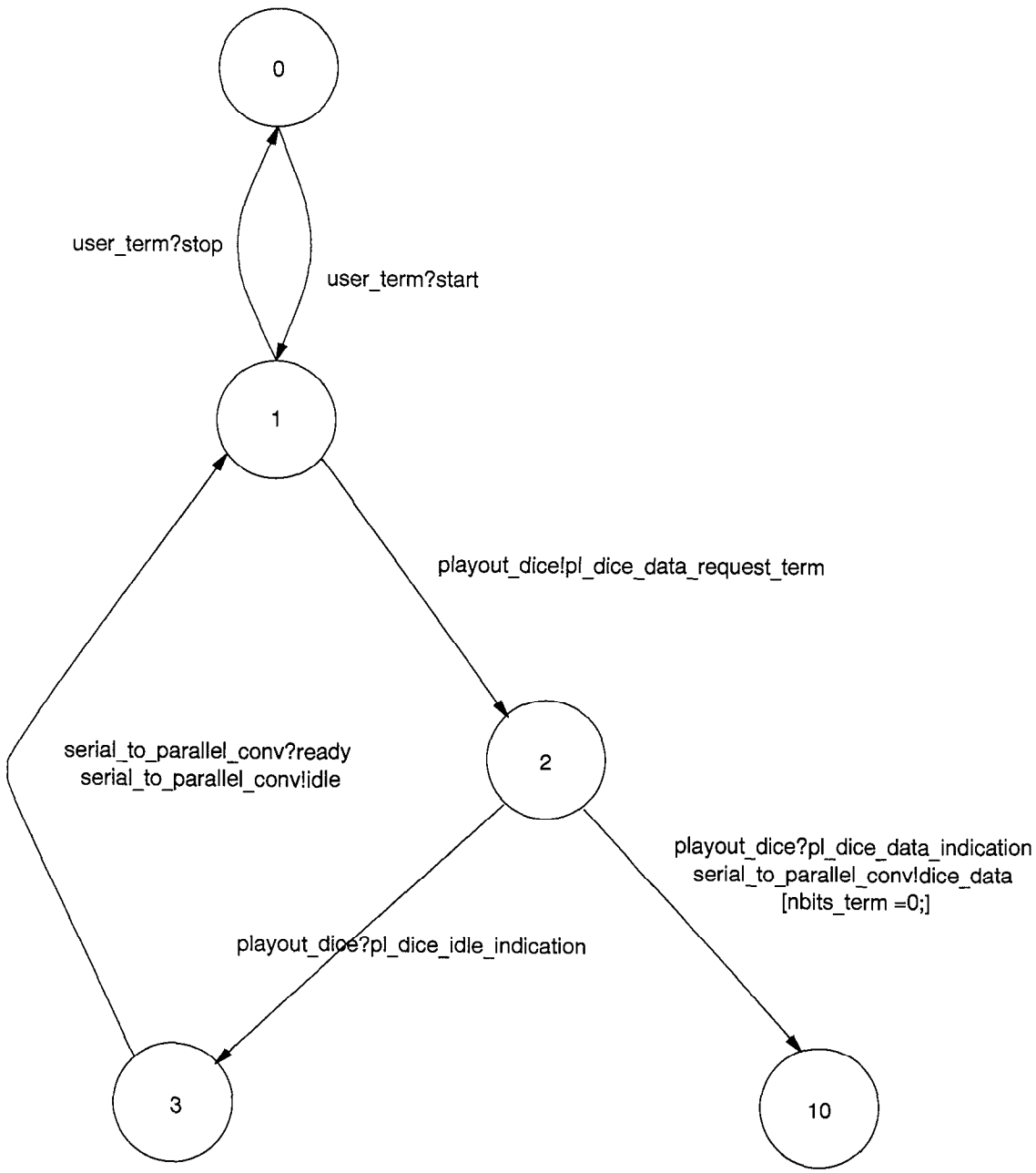


Figure C.12 – hi_layer_term
Part a

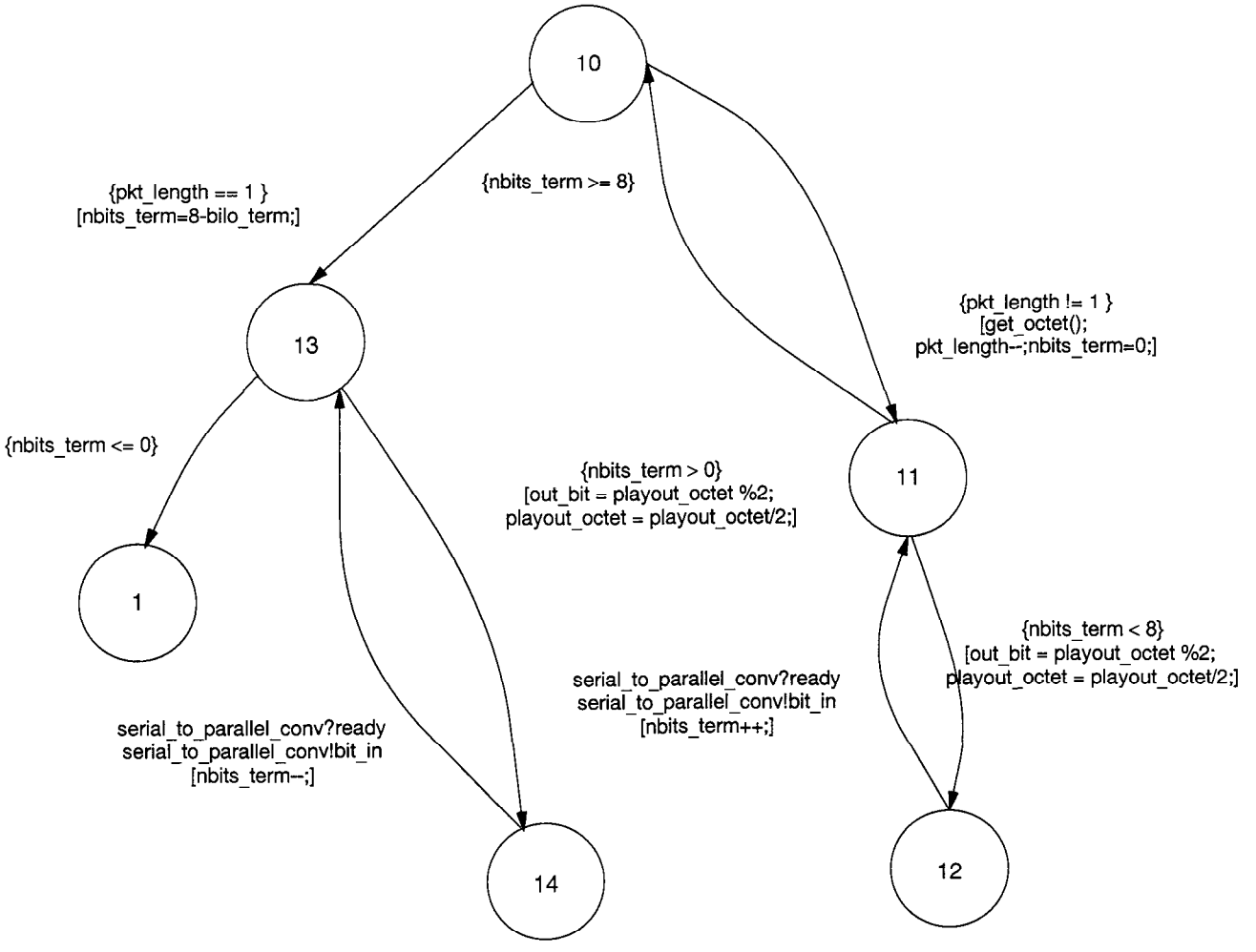


Figure C.12 (concluded) – hi_layer_term
Part b

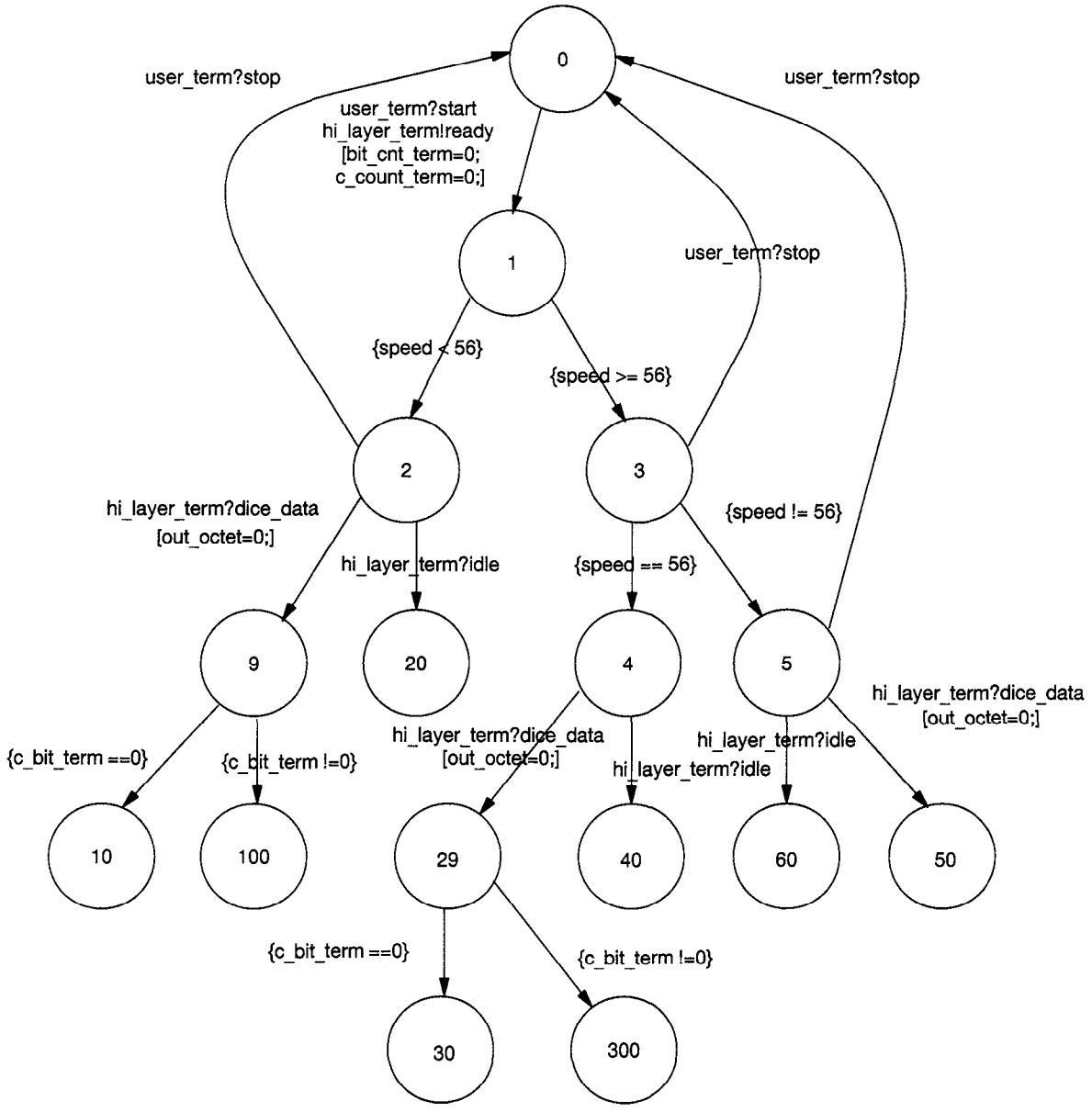


Figure C.13 – serial_to_parallel_conv
Part a

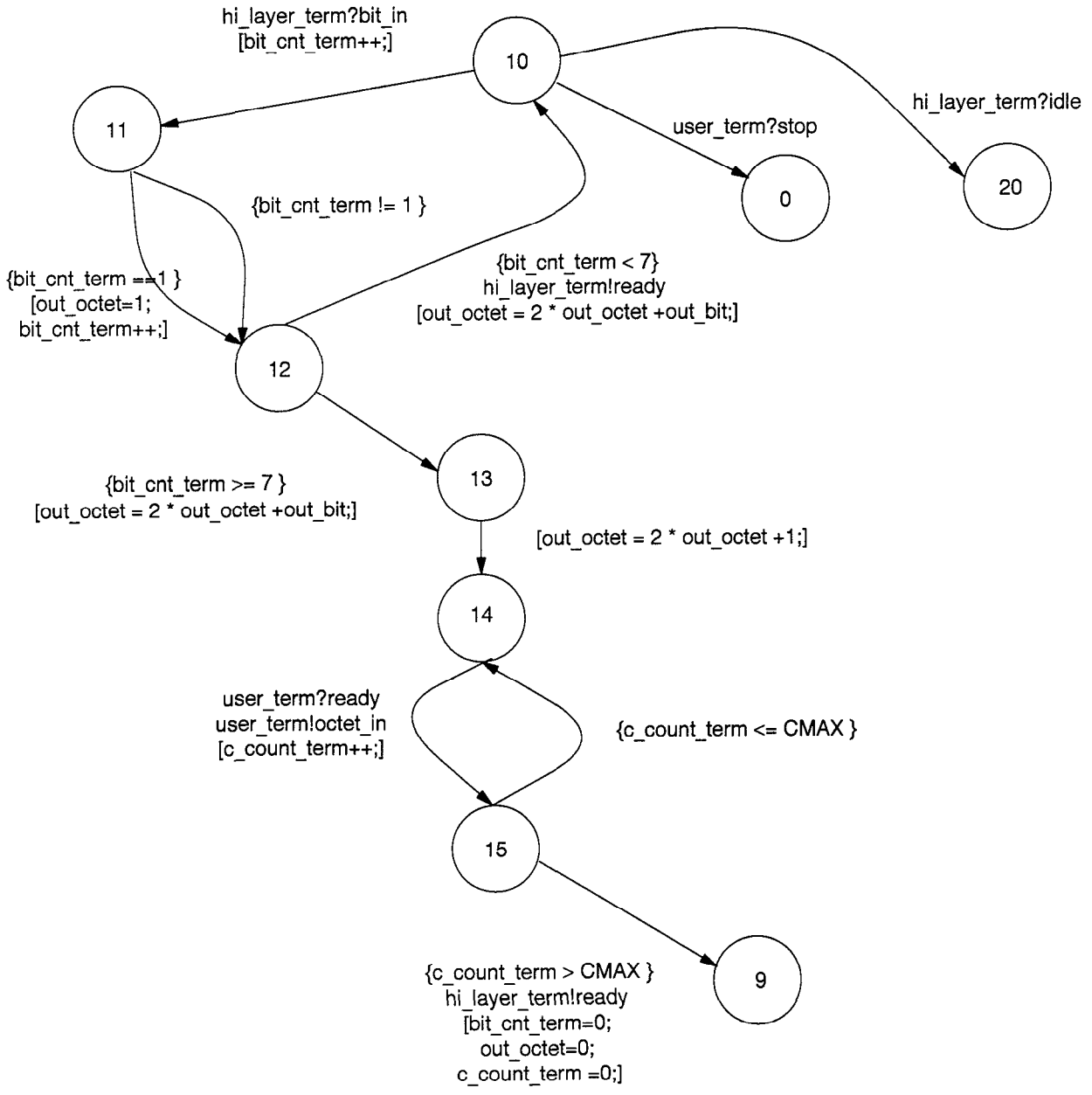


Figure C.13 (continued) – serial_to_parallel_conv
Part b

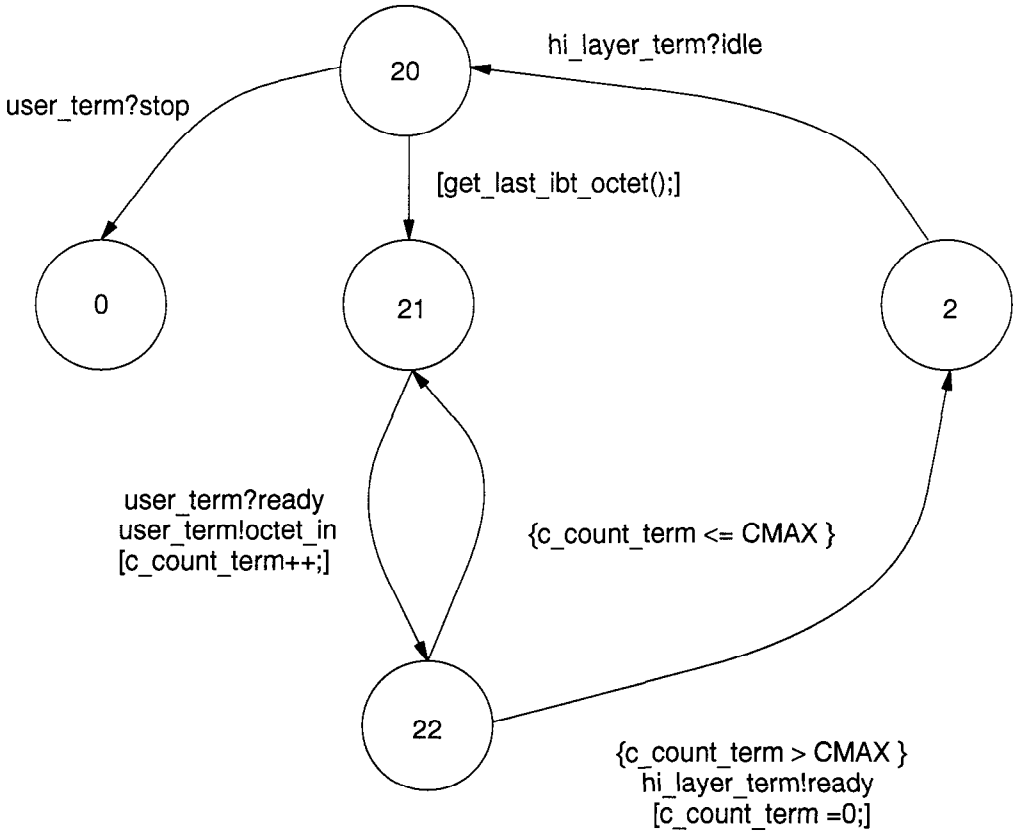


Figure C.13 (continued) – serial_to_parallel_conv
Part c

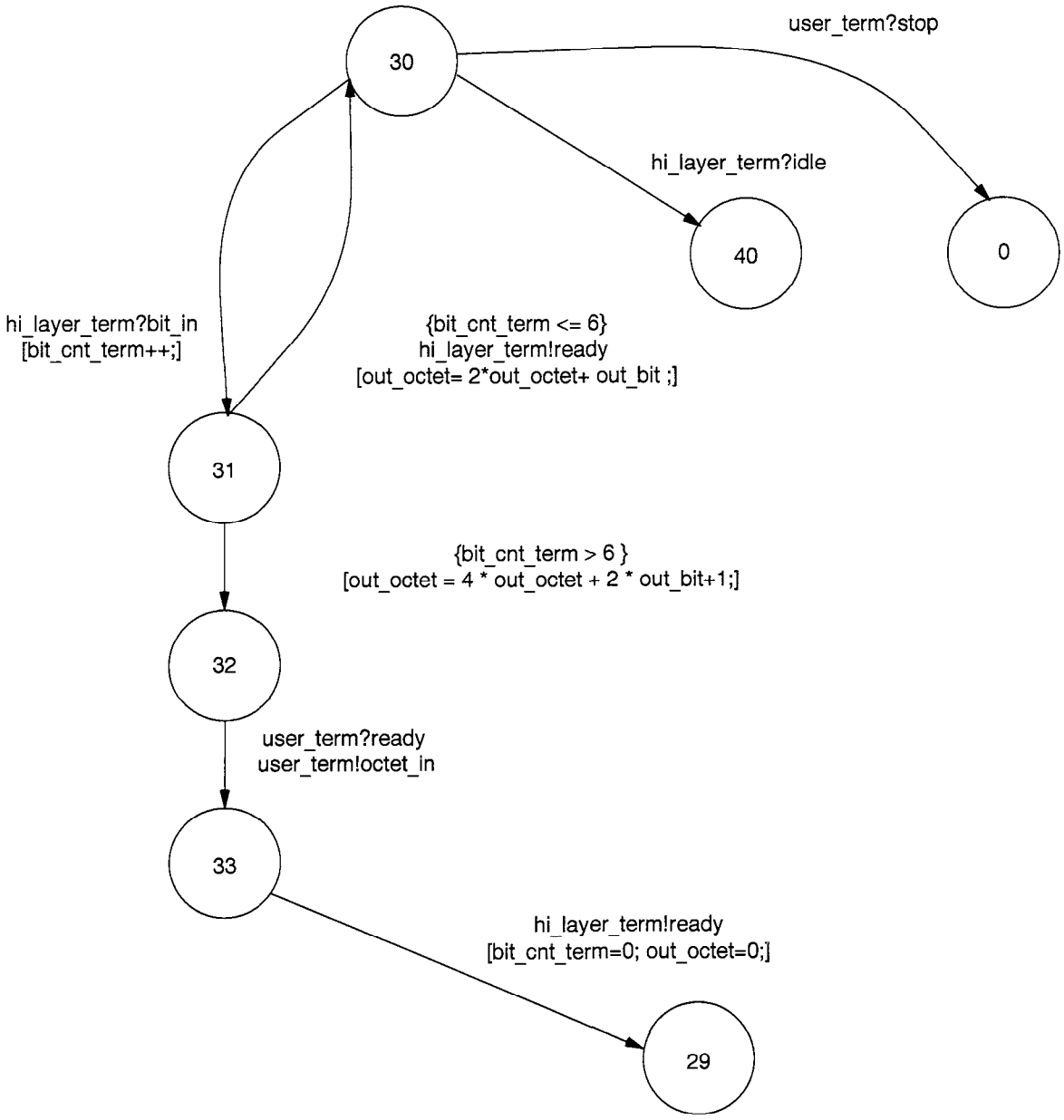


Figure C.13 (continued) – serial_to_parallel_conv
Part d

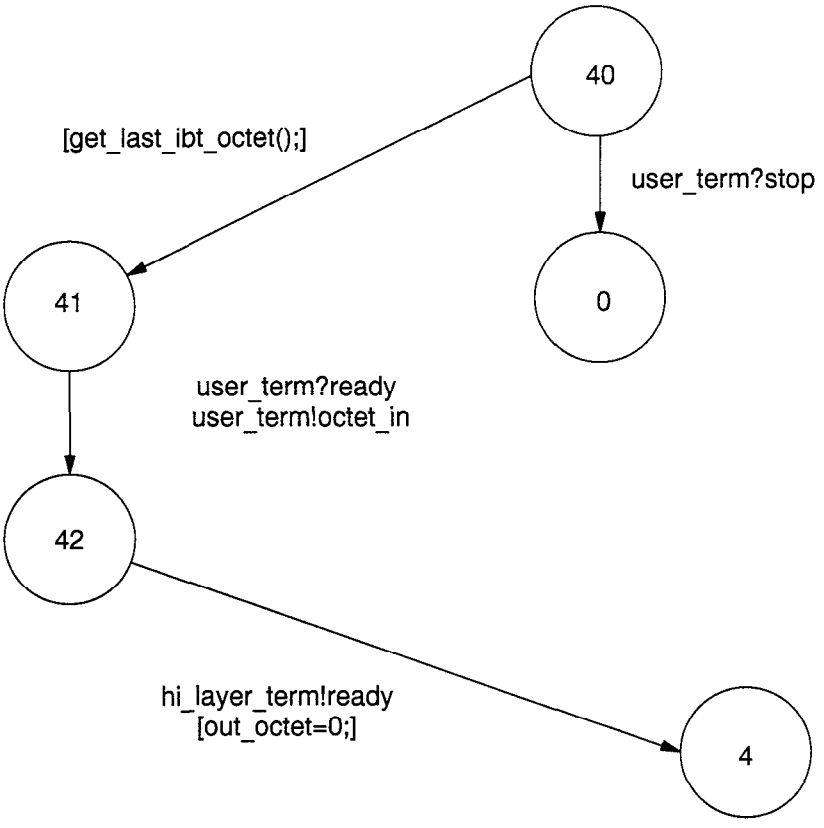


Figure C.13 (continued) – serial_to_parallel_conv
Part e

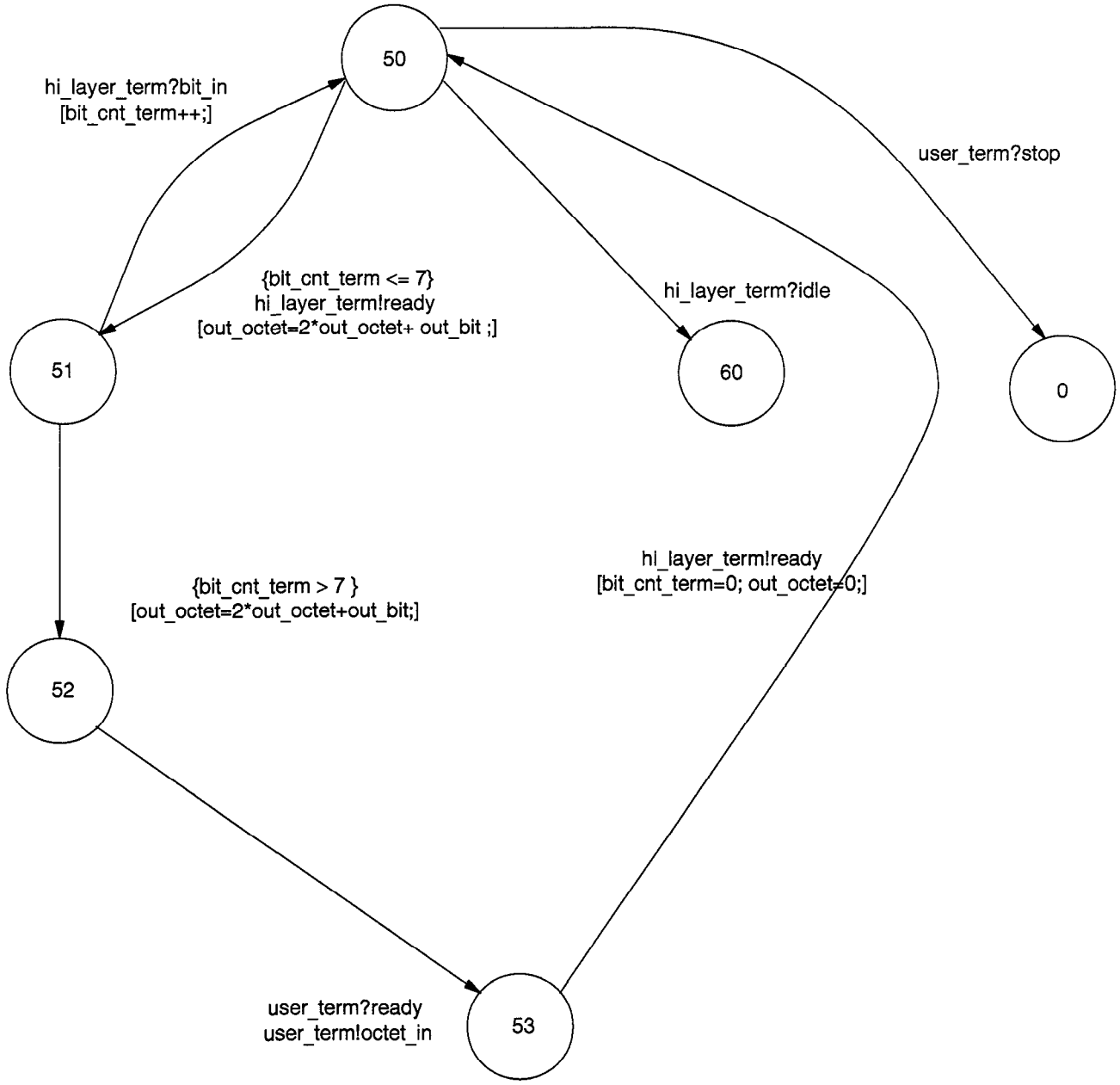


Figure C.13 (continued) – serial_to_parallel_conv
Part f

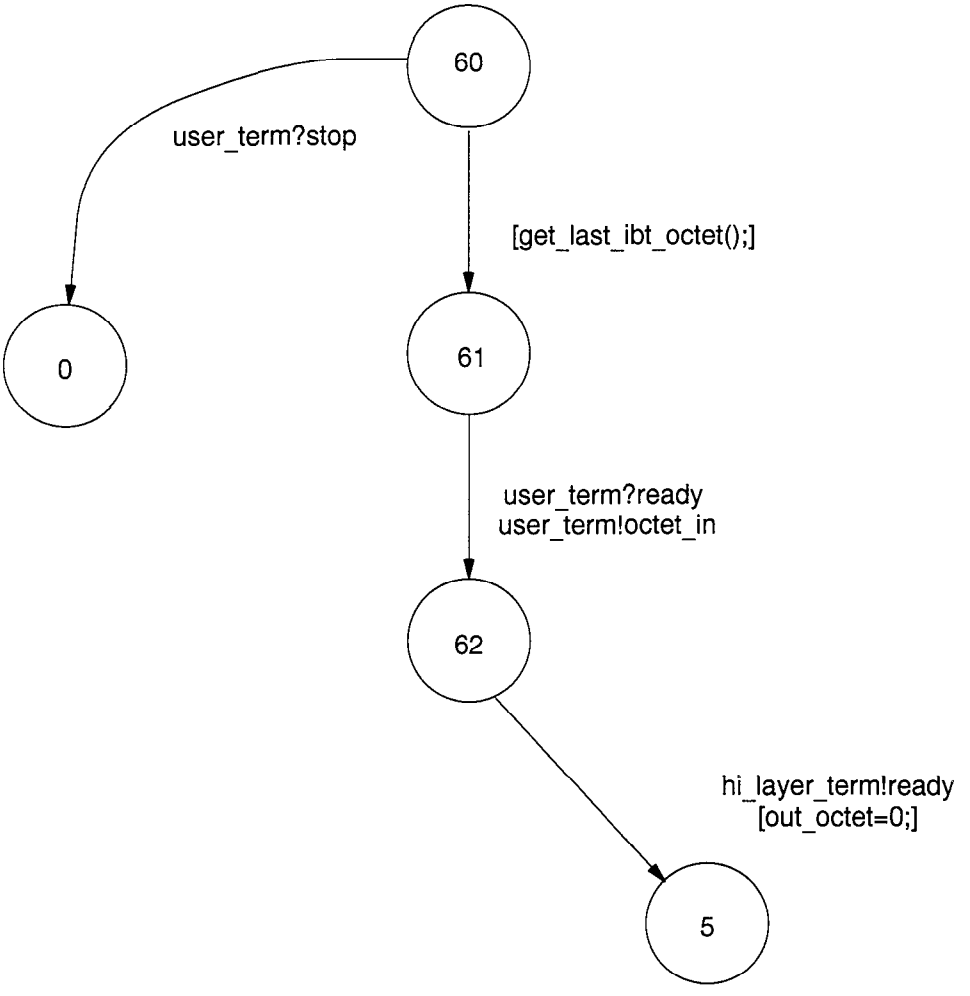


Figure C.13 (continued) – serial_to_parallel_conv
Part g

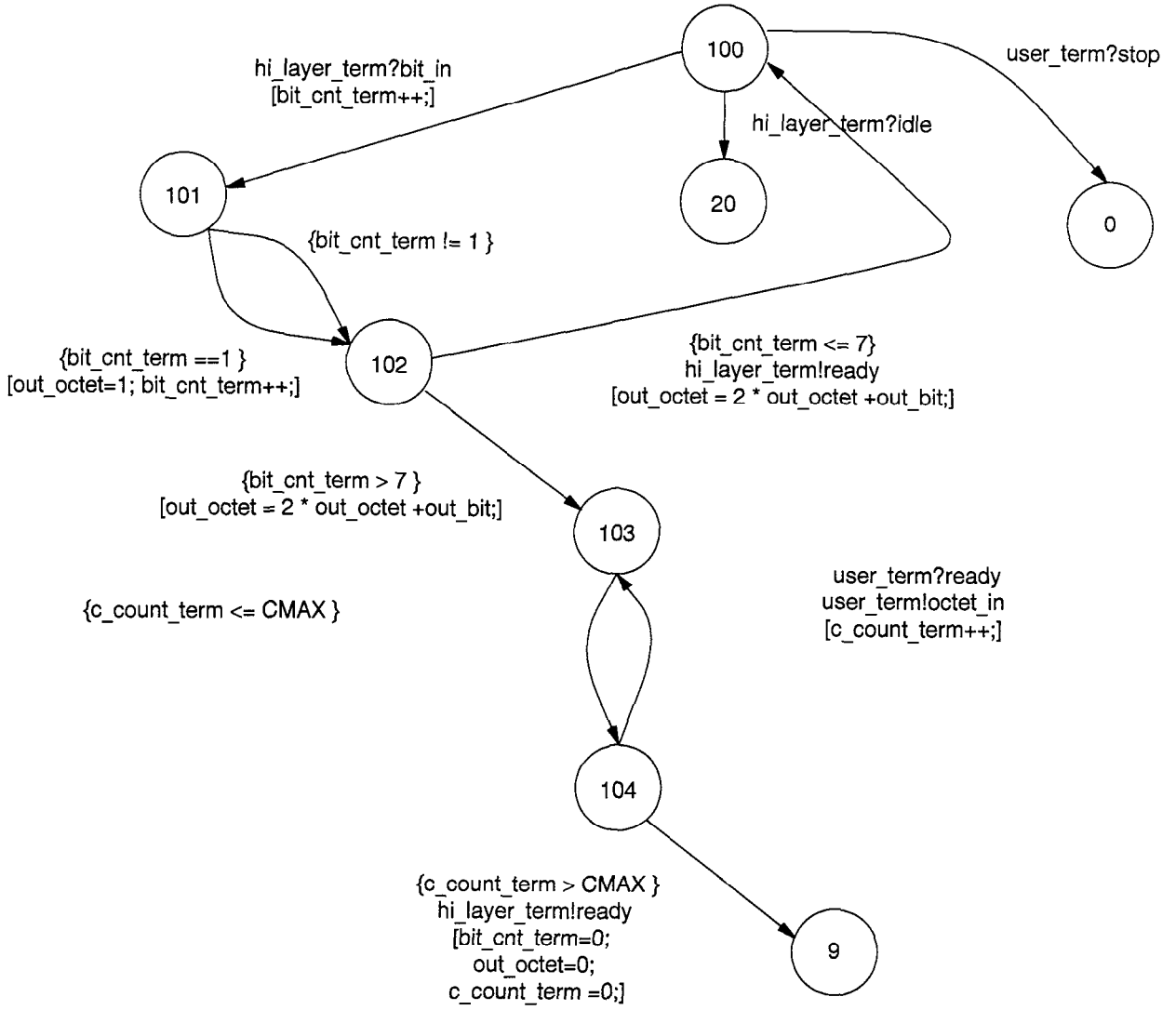


Figure C.13 (continued) – serial_to_parallel_conv
Part h

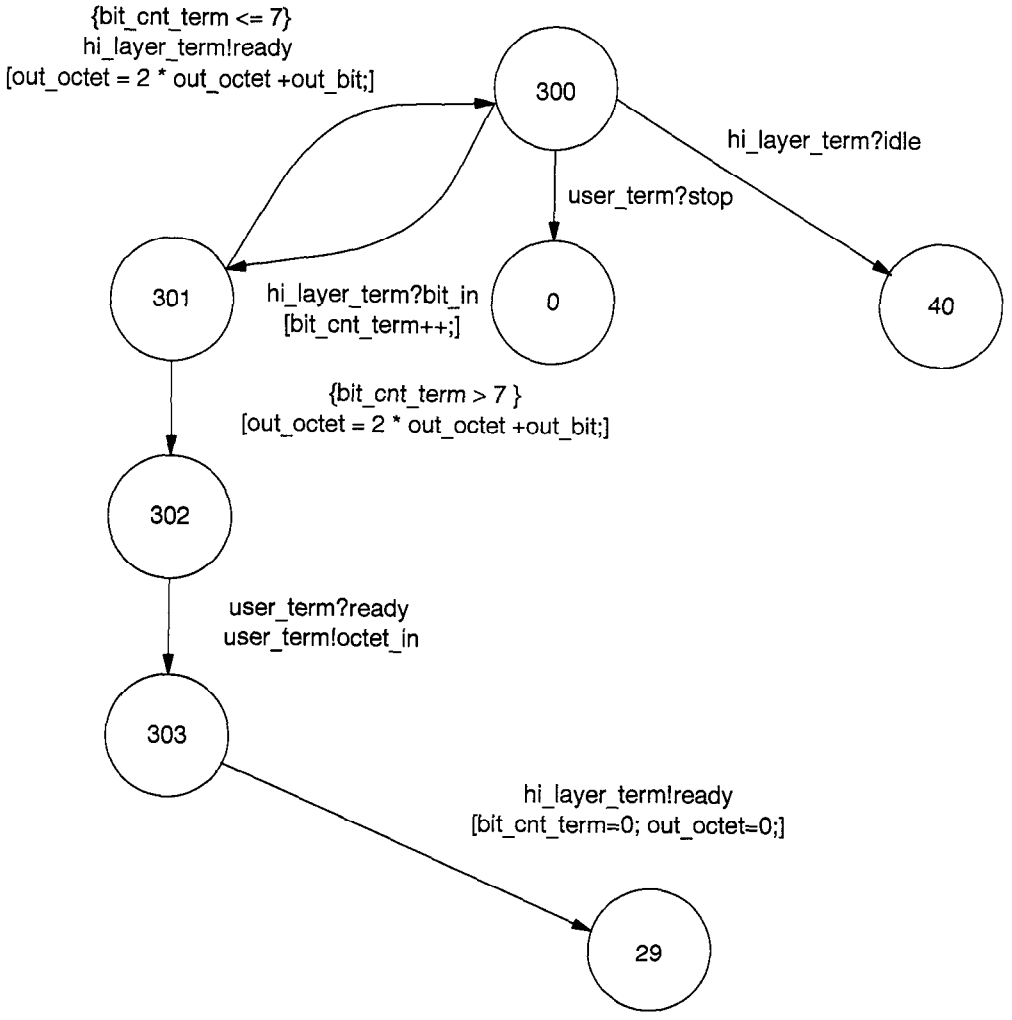


Figure C.13 (concluded) – serial_to_parallel_conv
Part i

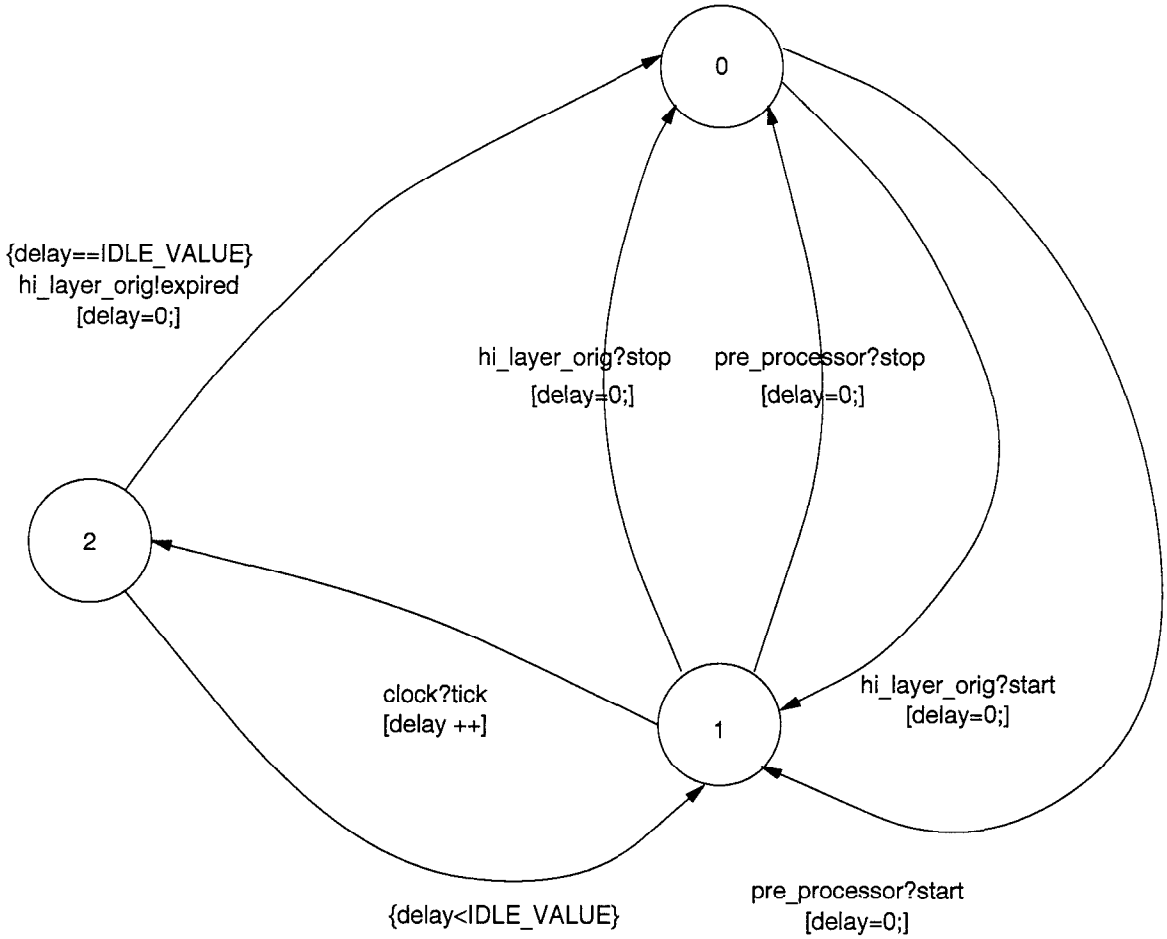


Figure C.14 – t_idle

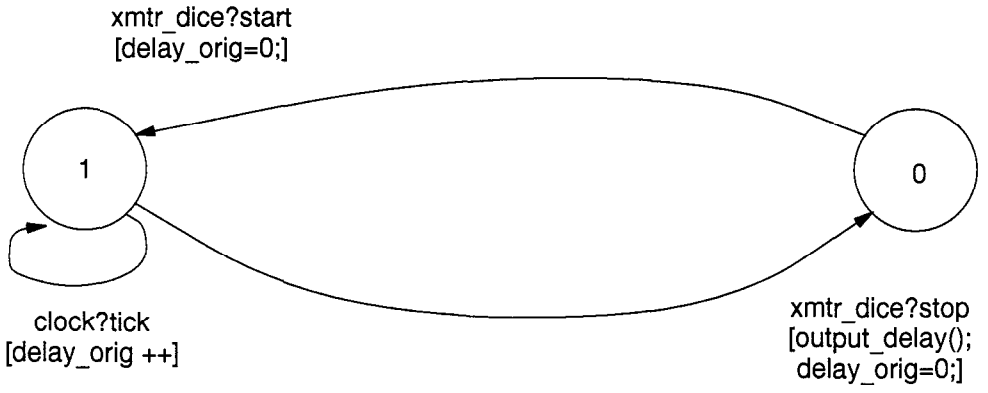


Figure C.15 – tvdelay_orig

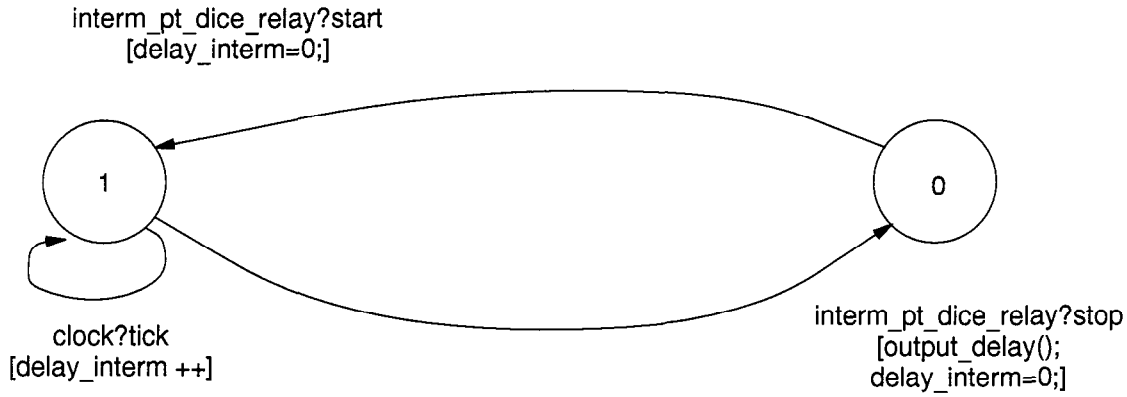


Figure C.16 – tvdelay_interm

Annex D
(informative)

Formal Description of the Virtual Data Link Capability (VDLC) Protocol

D.1 Processes in APSL description

Table D.1 lists all the processes used in the Augmented Protocol Specification Language (APSL) specification of the VDLC Protocol. It includes a brief description of each:

Table D.1

Process name	Description
abort_detector	HDLC abort detector at the originating end
data_detector	HDLC data detector at the originating end
flag_detector	HDLC flag detector at the originating end
hi_layer_orig	High layer entity at the originating end
hi_layer_term	High layer entity at the terminating end
idle_code_detector	Idle code detector at the originating end
interm_pt_vdlc_relay	VDLC intermediate node
parallel_to_serial_conv	Converter of an octet to a series of bits at the originating end
playout_vdlc	VDLC packet layer playout
pre_processor	Pre-processor at the originating end
rcv_queue	VDLC packet layer queue at the terminating end
rcv_vdlc	VDLC packet layer receiver at the terminating end
serial_to_parallel_conv	Converter of a series of bits into an octet at the terminating end
xmtr_vdlc	VDLC packet layer transmitter at the originating end
t_idle	Idle update timer
tvdelay_interm	VDLC variable delay timer at the intermediate node
tvdelay_orig	VDLC variable delay timer at the originating end
clock	Clock
layer2_interm	Layer 2 at the intermediate node
layer2_orig	Layer 2 at the originating node
layer2_term	Layer 2 at the terminating node
mgmt_term	Management entity at the terminating end
user_orig	User process at the originating end
user_term	User process at the terminating end

The following are treated as EXTERNAL processes clock,layer2_interm, layer2_orig,layer2_term, mgmt_term, user_orig, and user_term.

Table D.2 lists messages that apply for any timer·

Table D.2

Message name	Description
timer!expired	The timer has exceeded its time-out value.
timer?start	Start the timer.
timer?stop	Stop the timer and reset all variables to zero

D.2 Functions in the APSL description of VDLC

Table D.3 alphabetically lists the various functions used in the APSL specification of VDLC:

Table D.3

Function	Description
chk_idle_tbl()	At the originating endpoint, checks if 'octet' corresponds to an idle code in Table 1, then sets the variable 'idle_orig' to TRUE or FALSE accordingly.
count_bits()	Counts the number of bits in the VDLC information field.
chop_frame()	Partitions the VDLC frame into 'n_frame' frames.
drop_pkt()	Drops the current packet.
get_idle_octet()	Checks if the IBT value is a valid entry in Table 3. If it is valid, it sets 'idle_term' to TRUE and then maps the IBT value to 'idle_code' using Table 3. Otherwise, it sets 'idle_term' to FALSE.
get_last_ibt_octet()	Maps the value of 'ibt_last' into an idle code using Table 3 and puts the idle code in out_octet.
get_octet()	Reads octet of data and puts it in 'playout_octet.' The data are arranged as shown in Figure 5, i e., bit 1 is of playout_octet is the least significant bit of the data while bit 8 is the most significant bit.
ibt_update()	Updates the value of the ibt variable based on the value retrieved from the idle code table that corresponds to the idle octet.
invert_bit()	Inverts the value of the bit.
output_delay()	Outputs the delay between reception and transmission of a packet at a node. This delay is measured by the tvdelay_vdlc process for VDLC packets.
pop_pkt()	Takes a packet off the playout queue for playout and sets the terminating end variables 'eq_term', 'ibt_term' and 'seq_term' accordingly.
push_pkt()	Queues a received packet and sets the corresponding playout_time variable
rseq_increment()	Increments the receive sequence number (RSEQ) (1 to 15 with rollover to 1)
seq_increment()	Increments the sequence number seq_orig (1 to 15 with rollover to 1) at the originating end and seq_term at the terminating end
set_playout_time()	Sets the variable 'playout_time' to BUILDOUT - time stamp + current_time.
set_pkt_length()	Calculates the packet length in octets and stores it in the variable 'pkt_length'.
ts_update()	Updates the time stamp by the value of the delay measured by the 'tvdelay_orig' or 'tvdelay_interm' timers.

D.3 Primitives

Table D.4 describes the various primitives used in the APSL description:

Table D.4

Primitive	APSL representation
DL_L1_READY_INDICATION	dl_l1_ready_indication
DL_PVP_DATA_REQUEST	dl_pvp_data_request
DL_PVP_H_DATA_REQUEST	dl_pvp_h_data_request
DL_UNIT_DATA_INDICATION	dl_unit_data_indication
DL_UNIT_H_DATA_INDICATION	dl_unit_h_data_indication
DL_UNIT_DATA_REQUEST	dl_unit_data_request
DL_UNIT_H_DATA_REQUEST	dl_unit_h_data_request
PL_VDLC_DATA_INDICATION	pl_vdlc_data_indication
PL_VDLC_DATA_REQUEST	pl_vdlc_data_request_term
PL_VDLC_DATA_REQUEST(SSEQ,IBT,EQ)	pl_vdlc_data_request_orig
PL_VDLC_FLAGS_INDICATION	pl_vdlc_flags_indication
PL_VDLC_IDLE_INDICATION(IBT)	pl_vdlc_idle_indication
PL_VDLC_IDLE_REQUEST(IBT)	pl_vdlc_idle_request
PL_VDLC_ONE_FLAG_REQUEST	pl_vdlc_one_flag_request

D.4 Counters in APSL description

Table D.5 lists all the counters used in the APSL specification of the VDLC Protocol. It includes a brief description of each:

Table D.5

Counter name	What it counts
bit_cnt_term	Bits already right shifted in 'out_octet' by the serial_to_parallel converter at the terminating end
c_count_orig	Octets copied in subrate data at the originating end
c_count_term	Octets copied in subrate data at the terminating end
data_bit_count	Bits in the HDLC data detector
flag_bit_count	Bits in the HDLC flag detector
flag_cnt	Flags generated in the high layer of the terminating end
hdlc_cnt_term	Bits in the HDLC flag at to be sent at the terminating end
idle_lat_cnt	Consecutive identical idle codes arriving for the channelized side
nbits_orig	Number of bits received in the series-to-parallel converter of the originating end
nbits_term	Number of bits processed in the high layer entity of the terminating end
ndata_bits	Number of data bits in the VDLC information field as returned by count_bits
n_ones_ab	Consecutive ones received by the abort detector
n_ones_data	Consecutive ones received by the data detector
n_ones_flag	Consecutive ones received by the flag detector
n_zeros_flag	Zeros received by the flag detector
n_ones_term	Consecutive ones at the terminating end
nbits_term	Bits processed in the high layer entity of the terminating end
nflag_orig	Flags at the originating end
nflag_term	Flags at the terminating end
nframe	Frames when a long VDLC frame is chopped
octet_cnt_orig	Octets in the VDLC information field at the originating end

D.5 Internal variables

Table D.6 lists internal variables used in the APSL description:

Table D.6

Variable	Description
arriving_bit	Bit peeled from playout_octet before bit stuffing
bilo	Bit in the last octet
bit	Bit sent from the parallel_to_serial_converter to the HDLC detector
c	= 0 when control bits have been removed = 1 otherwise;
current_time	Current time as determined by a universal clock
delay	Time laps for the t_idle timer
delay_interm	Delay in the intermediate node
delay_orig	Delay in the originating end
eq_orig	Value of the delay equalization bit in the packet to be transmitted
eq_term	Value of the delay equalization bit in the packet received
data_octet	Octet of data bits collected in the HDLC data detector
first_bit	First bit to be retained from input octet
have_room	= TRUE when the packet can be stored = FALSE otherwise
ibt_orig	Value of the IBT field at the originating end
ibt_term	Value of the IBT field of the received packet at the terminating end and is used to distinguish between HDLC and non-HDLC operation
ibt_idle	The most recently updated value of the IBT code to be put in a VDLC packet
ibt_last	Value of the last IBT that corresponds to the entries of Table 3
idle_code	Idle code to be put in the output channelized stream
idle_orig	= TRUE when the input octet corresponds to an idle code = FALSE otherwise
idle_term	= TRUE when the IBT of the arriving packet corresponds to an entry in Table 3 = FALSE otherwise
input_octet	Octet arriving from the channelized side with the first bit arriving from the channelized side (the most significant bit) being in bit 1, i.e., the subrate framing bit is in bit 1 of input_octet and the control bit is in bit 8
invert_orig	= TRUE if the data bits must be inverted at the originating end = FALSE otherwise
invert_term	= TRUE if the data bits must be inverted at the terminating end = FALSE otherwise

(continued)

Table D.6 (concluded)

Variable	Description
last_bit	Last bit to be retained from the input octet
long_hdlc	= TRUE if the HDLC packets are to be segmented = FALSE otherwise
n_frame	Number of frames after chopping a long VDLC frame
octet	Copy of input_octet for the idle code detector with bit 1 being the most significant bit and bit 8 the least significant bit,
old_idle	Last received idle code from the channelized side
out_bit	Bit that hi_layer_term has peeled from 'playout_octet' and is giving to the serial_to_parallel_conv
out_octet	Octet output by the serial_to_parallel converter at the terminating end with the bits arranged as in Tables 1 and 3 so that bit 1 of the channelized pattern is the most significant bit of out_octet while bit 8 is the least significant bit
pkt_length	Length in octets of the VDLC information field in the received frame
playout_octet	Octet to be converted from parallel-to-serial representation by the high layer entity of the terminating end
playout_time	Time scheduled for playout of a received packet as calculated by set_playout_time()
pd	Protocol Discriminator
q_empty	= TRUE when playout queue is empty = FALSE otherwise
remove_control	= TRUE when control bits are to be removed = FALSE otherwise
rseq_term	RSEQ variable at the terminating end
seq_orig	SSEQ variable at the originating end
seq_term	SSEQ variable in the received packet
sig_class	Signal class
speed	Bit rate
ts	Time Stamp (updated by the ts_update function)
uih	= TRUE when the frames used are UIH = 0 when the frames used are UI

D.6 Coordination messages

Table D.7 lists the messages used in the APSL description to coordinate between various originating end processes:

Table D.7

Origin	Message	Destination
abort_detector	abort "	hi_layer_orig data_detector
flag_detector	flag "	hi_layer_orig data_detector
data_detector	data_in data_unaligned	hi_layer_orig "
hi_layer_orig	pl_vdlc_data_request_orig pl_vdlc_flag_request pl_vdlc_idle_request start stop	xmtr_vdlc " " t_idle "
hi_layer_term	error bit_in hdlc_data idle pl_vdlc_data_request_term	mgmt_term serial_to_parallel_conv " " playout_vdlc
idle_code_detector	idle_lat_max_reached ready vdlc_idle_max_reached	hi_layer_orig pre_processor hi_layer_orig
interm_pt_vdlc_relay	dl_pvp_data_request dl_pvp_h_data_request start stop	layer2_interm " tv_delay_interm "
parallel_to_serial_conv	bit_in " " ready	abort_detector data_detector flag_detector pre_processor
playout_vdlc	pl_vdlc_data_indication pl_vdlc_flags_indication pl_vdlc_one_flag_indication pl_vdlc_idle_indication read_pkt scan	hi_layer_term " " " rcv_queue "

(continued)

Table D.7 (concluded)

Origin	Message	Destination
pre_processor	data_in " start " " " " stop " " " "	idle_code_detector parallel_to_serial_conv abort_detector data_detector flag_detector idle_code_detector t_idle abort_detector data_detector flag_detector idle_code_detector t_idle
rcv_queue	queue_empty pkt_in	playout_vdlc "
rcv_vdlc	write_pkt	rcv_queue
serial_to_parallel_conv	octet_in ready	user_term hi_layer_term
t_idle	expired	hi_layer_orig
user_orig user_orig user_orig	input_in data_in start " stop "	pre-processor idle_code_detector pre-processor idle_code_detector pre-processor idle_code_detector
user_term	ready start stop	serial_to_parallel_conv " "
xmtr_vdlc	dl_unit_data_request dl_unit_h_data_request start stop	layer2_orig " tv_delay_orig "

```
/*
        various definitions
                        */

#define BUILDOUT 100
#define CMAX 5
#define FALSE 0
#define FLAG_MAX      1
#define FLAG_MIN      3
#define IDLE_LAT_MAX 2
#define IDLE_VALUE 60
#define PVP 0x44
#define TRUE  1
#define VDLC_IDLE_MAX       8
#define VMAX 133
/*
                arriving_bit      = bit peeled from playout_octet before bit stuffing,
                bilo              = bit in the last octet,
                bit               = bit sent from the parallel_to_serial_converter to the hdlc detectors
                bit_cnt_term      = counter of bits already right shifted in the
                                    serial_to_parallel converter at  the terminating end
                c                 = 1 when control bits have been removed,
                                  = 0 otherwise;
                c_count_orig      = counter for the number of octets copied in subrate data at the originating end
                c_count_term      = counter for the number of octets copied in subrate data at the terminating enc
                current_time      current time as determined by a universal clock
                data_bit_count    = counter for bits in the HDLC data detector,
                data_octet        = octet of data bits formed in  the HDLC data detector,
                delay             = time laps for the t_idle timer,
                delay_interm      = delay in the intermediate node,
                delay_orig        = delay in the originating end,
                eq_orig           = value of the delay equalization bit in the packet to be transmitted,
                eq_term                   = value of the delay equalization bit in the packet  received,
                flag_bit_count    = counter for bits in the HDLC flag detector,
                flag_cnt= counter for the number of flag to be generated in the high layer of the terminating enc
                first_bit = first bit to be retained from the input octet,
                have_room         = TRUE when the packet can be stored,
                                  = FALSE otherwise;
                ibt_idle = the most recently updated value of the IBT code to
                                    be put in a VDLC packet,
                ibt_last = value of the last IBT that corresponds to the
                                    entries of Table 3,
                ibt_orig = value of the IBT field at the originating end,
                ibt_term          = value of the IBT field at the terminating end
                                    and is used to distinguish between HDLC and non-HDLC operation,
                idle_code         idle code to be put in the output channelized stream
                idle_lat_cnt      = counter for the number of consecutive identical idle codes
                                    received from the channelized side,
                idle_orig         = TRUE when the input octet corresponds to an idle code in Table 1,
                                  = FALSE otherwise,
                idle_term         = TRUE when the IBT of the arriving packet  corresponds to a value in Table 3,
                                  = FALSE otherwise,
                input_octet       = octet arriving from the channelized side,
                invert_orig       = TRUE if the bits must be inverted at the originating endpoint,
                                  = FALSE otherwise
```

```
                invert_term     = TRUE if the bits must be inverted at the terminating endpoint,
                                = FALSE otherwise
                last_bit  = last bit to be retained from the input octet,
                long_hdlc       = TRUE if the HDLC packets are to be segmented,
                                = FALSE otherwise;
                n_frame         = number of frames after chopping a long VDLC frame,
                n_ones_ab       = counter for the number of consecutive ones received by the abort detector,
                n_ones_data     = counter for the number of consecutive ones received by the data detector,
                n_ones_flag     = counter for the number of consecutive ones received by the flag detector,
                n_ones_term     = counter for the number of consecutive ones at the terminating end,
                n_zeros_flag    = counter for the number of zeros received by the flag detector,
                nbits_orig      = number of bits received in the series-to-parallel converter of the originating end,
                nbits_term      = number of bits processed in the high layer entity of  the terminating end,
                ndata_bits      = number of data bits in the VDLC information field as returned by count_bits,
                nflag_orig      = counter for the number of flags at the originating end,
                nflag_term      = counter for the number of flags at the terminating end,
                nframe          = counter for the number of frames when the VDLC frame is chopped,
        octet           = a copy of input_octet for the idle code detector,
                octet_cnt_orig  = counter for the number of octets in the VDLC information field
                                  at the originating end,
                out_bit         = bit peeled from playout_octet,
                old_idle = last received idle code from the channelized side,
                out_octet       = octet output by the  serial_to_parallel converter at the terminating end with
                                  the bits arranged as in Tables 1 and 3 so that bit 1 of the channelized
                                  pattern is the most significant bit of out_octet while bit 8 is
                                  the least significant bit,
                pd              Protocol Discriminator
                pkt_length      = length in octets of the VDLC information field in the received frame,
                playout_octet   = octet to be converted from parallel-to-serial representation
                                  by the high layer entity of the terminating end,
                playout_time    = time scheduled for playout of a received packet,
                q_empty         = TRUE when playout queue is empty,
                                = FALSE otherwise;
                rseq_term       RSEQ variable at the terminating end
                seq_orig        SEQ variable at the originating end
                seq_term        SEQ variable in the received packet
                sig_class       signal class
                speed           bit rate
                ts              Time Stamp (updated by the ts_update function)
                uih             = TRUE when the frames used are UIH
                                = FALSE when the frames used are UI
                                                                        */

/*
                External Processes
*/
EXTERNAL clock, layer2_orig, layer2_interm, layer2_term, user_orig,  user_term, mgmt_term;
/*
                Pre-Processor at the Originating End
*/
PROCESS         pre_processor;
CONTEXT         c, c_count_orig, first_bit, input_octet, last_bit, octet, speed;

STATES          0-8;
REND  user_orig?start, user_orig?stop, user_orig?input_in,
```

```
        idle_code_detector!start, idle_code_detector!stop,
        idle_code_detector!data_in,
        idle_code_detector?ready,
        flag_detector!start, flag_detector!stop,
        abort_detector!start, abort_detector!stop,
        data_detector!start, data_detector!stop,
        parallel_to_serial_conv!data_in,
        parallel_to_serial_conv?ready,
        t_idle!start, t_idle!stop;

INITIAL STATE 0;
TRANSITIONS

        /* Figure D.1 */

0:1     WHEN user_orig?start * idle_code_detector!start * flag_detector!start
            * abort_detector!start * data_detector!start
            * t_idle!start [c_count_orig=0; octet=0; input_octet=0;],
1:0     WHEN user_orig?stop * idle_code_detector!stop * flag_detector!stop
            * abort_detector!stop * data_detector!stop * t_idle!stop,
1:2     WHEN user_orig?input_in,
        /*
           The first bit to arrive
           is the subrate framing bit and the last bit
           is the control bit;
           input_octet has the subrate framing bit in bit 1 and
           the control bit in bit 8 as shown in Table 1
           In a LAPD frame, the subrate framing bit must be
           in octet 1, since according to LAPD, bit 1 is
           transmitted first:
           i.e., the bit order must be reversed

           octet is a copy of input_octet for the idle code detector
         */
2:3     WHEN {speed >= 56},
3:4     WHEN {speed != 56} [octet=input_octet;first_bit=1; last_bit=8;c=1;],
3:4     WHEN {speed ==56} [octet=input_octet;first_bit=1; last_bit=7;c=0;],
2:5     WHEN {speed < 56} [c_count_orig ++;],
5:6     WHEN {c_count_orig!=1},
6:1     WHEN {c_count_orig < CMAX},
6:1     WHEN {c_count_orig == CMAX} [c_count_orig=0;],
5:4     WHEN {c_count_orig == 1} [octet=input_octet; first_bit=2; last_bit=7;c=0;],
4:7     WHEN idle_code_detector?ready,
7:8     WHEN parallel_to_serial_conv?ready,
8:1     WHEN idle_code_detector!data_in *
            parallel_to_serial_conv!data_in [input_octet=0;].

/*
                Idle Code Detector
*/

PROCESS     idle_code_detector;
CONTEXT     idle_orig, old_idle, octet, idle_lat_cnt;
STATES      0-8;
REND  pre_processor ?start, pre_processor ?stop,
```

```
        pre_processor?data_in, pre_processor!ready,
        hi_layer_orig!idle_lat_max_reached,
        hi_layer_orig!vdlc_idle_max_reached;

INITIAL STATE 0;
TRANSITIONS

                /* Figure D.2 (a)  */

0:1     WHEN pre_processor?start [idle_lat_cnt=0;old_idle=0xFF;],
1:0     WHEN pre_processor?stop,
1:2     WHEN pre_processor?data_in [chk_idle_tbl();],
2:1     WHEN {idle_orig == FALSE} [idle_lat_cnt=0;],
2:3     WHEN {idle_orig == TRUE},
3:1     WHEN {octet != old_idle} [old_idle = octet; idle_lat_cnt = 1;],
3:4     WHEN {octet == old_idle} [idle_lat_cnt++;],

                /* Figure D.2 (b) */

4:5     WHEN {idle_lat_cnt < IDLE_LAT_MAX},
4:6     WHEN {idle_lat_cnt >= IDLE_LAT_MAX},
6:5     WHEN {idle_lat_cnt > IDLE_LAT_MAX},
6:5     WHEN {idle_lat_cnt == IDLE_LAT_MAX} hi_layer_orig!idle_lat_max_reached,
5:1     WHEN {idle_lat_cnt < VDLC_IDLE_MAX},
5:7     WHEN {idle_lat_cnt >= VDLC_IDLE_MAX},
7:1     WHEN {idle_lat_cnt > VDLC_IDLE_MAX} pre_processor!ready,
7:1     WHEN {idle_lat_cnt == VDLC_IDLE_MAX}
          hi_layer_orig!vdlc_idle_max_reached * pre_processor!ready.
/*
                Parallel_To_Serial_Conv
*/

PROCESS     parallel_to_serial_conv;
CONTEXT     bit, first_bit, invert_orig, last_bit, nbits_orig, octet ;
STATES      0-4;
REND    pre_processor?data_in,
        pre_processor!ready,
        flag_detector!bit_in, abort_detector!bit_in,
        data_detector!bit_in;

INITIAL STATE 0;
TRANSITIONS

                /* Figure D.3  */

0:1     WHEN pre_processor?data_in [nbits_orig=1;],
1:0     WHEN {nbits_orig > last_bit } pre_processor!ready,
1:2     WHEN {nbits_orig <= last_bit },
2:1     WHEN {nbits_orig < first_bit} [nbits_orig++; octet= octet /*<< 1 */;],
2:3     WHEN {nbits_orig >= first_bit} [bit = octet& 0x8; octet= octet /* << 1 */ ;],
3:4     WHEN {invert_orig != TRUE},
3:4     WHEN {invert_orig == TRUE} [invert_bit();],
4:0     WHEN flag_detector!bit_in * abort_detector!bit_in *
          data_detector!bit_in [nbits_orig++;].
```

```
/*
                Flag Detector
*/

PROCESS     flag_detector;
CONTEXT     bit, flag_bit_count, n_ones_flag, n_zeros_flag;
STATES      0-8;
REND  pre_processor?start, pre_processor?stop, parallel_to_serial_conv?bit_in,
      data_detector!flag, hi_layer_orig!flag;

INITIAL STATE 0;
TRANSITIONS

                /* Figure D.4 */


0:1     WHEN pre_processor?start [flag_bit_count=0; n_ones_flag=0;n_zeros_flag=0;],
1:0     WHEN pre_processor?stop,
1:2     WHEN parallel_to_serial_conv?bit_in,
2:4     WHEN {bit==0} [flag_bit_count++;],
4:1     WHEN {flag_bit_count == 1} [flag_bit_count=1;n_zeros_flag=1;n_ones_flag=0;],
4:7     WHEN {flag_bit_count > 1},
7:1     WHEN {flag_bit_count < 8} [flag_bit_count=1;n_zeros_flag = 0; n_ones_flag=0;],
7:8     WHEN {flag_bit_count == 8} [n_zeros_flag++;],
8:1     WHEN {n_zeros_flag != 2} [flag_bit_count =1; n_ones_flag=0; n_zeros_flag=1;],
2:3     WHEN {bit==1}[n_ones_flag++;flag_bit_count++;],
3:1     WHEN {flag_bit_count < 7},
3:5     WHEN {flag_bit_count >= 7},
5:1     WHEN {flag_bit_count > 7} [flag_bit_count =0; n_ones_flag=0; n_zeros_flag=0;],
5:6     WHEN {flag_bit_count == 7},
6:1     WHEN {n_ones_flag!=6} [flag_bit_count =0; n_ones_flag=0; n_zeros_flag=0;],
6:1     WHEN {n_ones_flag == 6},
8:1     WHEN {n_zeros_flag == 2} hi_layer_orig!flag * data_detector!flag
          [flag_bit_count =1; n_ones_flag=0;n_zeros_flag=1;].


/*
                Abort Detector
*/

PROCESS     abort_detector;
CONTEXT     bit, n_ones_ab;
STATES      0-3;
REND  pre_processor?start, pre_processor?stop, parallel_to_serial_conv?bit_in,
      data_detector!abort, hi_layer_orig!abort;

INITIAL STATE 0;
TRANSITIONS

                /* Figure D.5 */


0:1     WHEN pre_processor?start [n_ones_ab=0;],
1:0     WHEN pre_processor?stop,
1:2     WHEN parallel_to_serial_conv?bit_in,
```

```
2:1     WHEN {bit==0} [n_ones_ab=0; ],
2:3     WHEN {bit==1} [n_ones_ab++;],
3:1     WHEN {n_ones_ab != 7},
3:1     WHEN {n_ones_ab == 7} hi_layer_orig!abort *data_detector!abort.


/*
                Data Detector
*/

PROCESS     data_detector;
CONTEXT     bit, data_bit_count, n_ones_data, data_octet;
STATES      0-7;
REND  pre_processor?start, pre_processor?stop, parallel_to_serial_conv?bit_in,
      flag_detector?flag, abort_detector?abort,
      hi_layer_orig!data_in, hi_layer_orig!unaligned_data;

INITIAL STATE 0;
TRANSITIONS

                /* Figure D.6 */


0:1     WHEN pre_processor?start [data_bit_count=0; n_ones_data=0;data_octet=0;],
1:0     WHEN pre_processor?stop,
1:2     WHEN parallel_to_serial_conv?bit_in,
2:3     WHEN {bit==0} [n_ones_data=0;],
2:3     WHEN {bit==1} [n_ones_data++;],
3:4     WHEN {n_ones_data == 6},
3:4     WHEN {n_ones_data != 6} [ data_bit_count++; data_octet = ( (bit |data_octet) /* >>1 */ ); n_ones_data =0;],
4:1     WHEN abort_detector?abort [data_bit_count=0;n_ones_data=0;data_octet=0;],
4:5     WHEN flag_detector?flag,
5:6     WHEN {data_bit_count ==7},
5:6     WHEN {data_bit_count !=7} hi_layer_orig!unaligned_data,
6:1     WHEN [data_bit_count=0;n_ones_data=0;data_octet=0;],
4:7     WHEN {DEFAULT},
7:1     WHEN {data_bit_count == 8 } hi_layer_orig!data_in
          [data_bit_count = 0; n_ones_data=0; data_octet=0;],
7:1     WHEN {data_bit_count !=8}.

/*
                high_layer_orig
*/

PROCESS     hi_layer_orig;
CONTEXT     ibt_orig, long_hdlc, octet_cnt_orig, nflag_orig, ndata_bits,
      nframe, n_frame,  seq_orig, eq_orig;
STATES      0-701;
REND  flag_detector?flag, data_detector?data_in, data_detector?unaligned_data,
      abort_detector?abort,
      idle_code_detector?idle_lat_max_reached,
      idle_code_detector?vdlc_idle_max_reached,
      t_idle!start, t_idle?expired, t_idle!stop,
      xmtr_vdlc!pl_vdlc_flag_request,
```

```
        xmtr_vdlc!pl_vdlc_data_request_orig, xmtr_vdlc!pl_vdlc_idle_request;

INITIAL STATE 0;
TRANSITIONS

                /* Figure D.7 */

                /* IDLE_STATE */

0:700   WHEN idle_code_detector?idle_lat_max_reached [ibt_update();],
0:700   WHEN t_idle?expired,
700:701         WHEN xmtr_vdlc!pl_vdlc_idle_request,
701:0   WHEN t_idle!start,
0:100   WHEN flag_detector?flag * t_idle!stop [octet_cnt_orig=0;ndata_bits=0; nflag_orig = 0; seq_orig= 0;],



                /* Figure D.8 */

                /* AWAIT_STATE */


100:101         WHEN flag_detector?flag  [nflag_orig ++;],
101:100         WHEN {nflag_orig < FLAG_MAX},
101:102         WHEN {nflag_orig == FLAG_MAX} t_idle!start [ ibt_orig = 2; eq_orig=0;],
102:150         WHEN xmtr_vdlc!pl_vdlc_flag_request,
100:103         WHEN abort_detector?abort [ibt_orig = 2; eq_orig= 0;],
103:120         WHEN xmtr_vdlc!pl_vdlc_flag_request,
100:200         WHEN data_detector?data_in  * xmtr_vdlc!pl_vdlc_data_request_orig [octet_cnt_orig =1;],


                /* Figure D.9 (a) */

                /* PACKETIZE_STATE */
200:201         WHEN data_detector?data_in,
201:200         WHEN [octet_cnt_orig++;],
200:202         WHEN flag_detector?flag,
202:200         WHEN {octet_cnt_orig< 4} [drop_pkt();],
202:203         WHEN {octet_cnt_orig >= 4},
203:300         WHEN data_detector?unaligned_data [drop_pkt();],
203:204         WHEN data_detector?data_in,
204:206         WHEN {octet_cnt_orig > VMAX},
204:205         WHEN {octet_cnt_orig <= VMAX} [ibt_orig = 0; eq_orig= 0; seq_increment();],
205:300         WHEN xmtr_vdlc!pl_vdlc_data_request_orig,
206:300         WHEN {long_hdlc != TRUE} [drop_pkt();],
206:207         WHEN {long_hdlc == TRUE} [chop_frame(); nframe = 1;],
200:120         WHEN abort_detector?abort [drop_pkt();],


                /* Figure D.9 (b) */

207:208         WHEN {nframe == 1} [ibt_orig =1; eq_orig=1; seq_orig= 0;],
208:207         WHEN xmtr_vdlc!pl_vdlc_data_request_orig [nframe++;],
207:209         WHEN {nframe > 1},
209:210         WHEN {nframe < n_frame} [ibt_orig=1;eq_orig=1;nframe++;seq_increment();],
```

```
210:207         WHEN xmtr_vdlc!pl_vdlc_data_request_orig,
209:211         WHEN {nframe == n_frame} [ibt_orig=0;eq_orig=0;seq_increment();],
211:300         WHEN xmtr_vdlc!pl_vdlc_data_request_orig,

                /* Figure D.10 */

                /* ABORT_STATE */

120:300         WHEN flag_detector?flag,
120:121         WHEN idle_code_detector?vdlc_idle_max_reached [ibt_update();],
121:122         WHEN xmtr_vdlc!pl_vdlc_idle_request,
122:0   WHEN t_idle!start,

                /* Figure D.11 */

                /* FLAG_WAIT_STATE */

300:303         WHEN data_detector?data_in [octet_cnt_orig=1;],
303:200         WHEN xmtr_vdlc!pl_vdlc_data_request_orig,
300:120         WHEN abort_detector?abort,
300:100         WHEN flag_detector?flag [nflag_orig = 1;],
300:301         WHEN idle_code_detector?vdlc_idle_max_reached [ibt_update();],
301:0   WHEN xmtr_vdlc!pl_vdlc_idle_request *t_idle!start,

                /* Figure D.12 */

                /* FLAG_IDLE_STATE */

150:200         WHEN data_detector?data_in * t_idle!stop [seq_orig=0;octet_cnt_orig =1;],
150:120         WHEN abort_detector?abort * t_idle!stop,
150:160         WHEN t_idle?expired,
160:150         WHEN t_idle!start * xmtr_vdlc!pl_vdlc_idle_request [ibt_orig =2;].
/*
        Originating End  XMTR -- Figure D.13
                        */

PROCESS         xmtr_vdlc;
CONTEXT seq_orig,uih,bilo;
STATES          100-104;
REND    hi_layer_orig?pl_vdlc_idle_request, hi_layer_orig?pl_vdlc_data_request_orig,
        tvdelay_orig!start, tvdelay_orig!stop,
        layer2_orig?dl_l1_ready_indication,
        layer2_orig!dl_unit_data_request,
        layer2_orig!dl_unit_h_data_request;

INITIAL STATE 100;
TRANSITIONS

100:101         WHEN  hi_layer_orig?pl_vdlc_data_request_orig,
100:101         WHEN  hi_layer_orig?pl_vdlc_idle_request [seq_orig =0;eq_orig=0;bilo=0;],
101:102         WHEN  tvdelay_orig!start,
102:103         WHEN  layer2_orig?dl_l1_ready_indication * tvdelay_orig!stop,
103:104         WHEN  [ts_update();] ,
104:100         WHEN  {uih == FALSE}layer2_orig!dl_unit_data_request,
104:100         WHEN  {uih == TRUE}layer2_orig!dl_unit_h_data_request.
```

```
/*
        Intermediate Point -- Figure D.14 --
                                                */

PROCESS     interm_pt_vdlc_relay;
CONTEXT pd, uih;
STATES              0-3;
REND  layer2_interm?dl_pvp_data_indication,
        layer2_interm?dl_pvp_h_data_indication,
        layer2_interm!dl_pvp_data_request,
        layer2_interm!dl_pvp_h_data_request,
        layer2_interm?dl_l1_ready_indication,
        tvdelay_interm!start, tvdelay_interm!stop;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  layer2_interm?dl_pvp_h_data_indication [uih = TRUE;],
0:1     WHEN  layer2_interm?dl_pvp_data_indication [uih = FALSE;],
1:0     WHEN  {pd != PVP},
1:2     WHEN  {pd == PVP} tvdelay_interm!start,
2:3     WHEN  layer2_interm?dl_l1_ready_indication * tvdelay_interm!stop [ts_update();],
3:0     WHEN  {uih == FALSE} layer2_interm!dl_pvp_data_request,
3:0     WHEN  {uih == TRUE} layer2_interm!dl_pvp_h_data_request.


/*
        Receiver Terminating End      -- Figure D.15
                                    */
PROCESS     rcv_vdlc;
CONTEXT pd,sig_class,ts;
STATES      0-4;
REND  layer2_term?dl_unit_h_data_indication,
        layer2_term?dl_unit_data_indication,  rcv_queue!write_pkt;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  layer2_term?dl_unit_h_data_indication,
0:1     WHEN  layer2_term?dl_unit_data_indication,
1:0     WHEN  {pd != PVP},
1:2     WHEN  {pd == PVP},
2:0     WHEN  {sig_class != 3},
2:3     WHEN  {sig_class == 3},
3:0     WHEN  {ts > BUILDOUT},
3:4     WHEN  {ts <= BUILDOUT} [set_playout_time();],
4:0     WHEN  rcv_queue!write_pkt.


/*
        Receiver Queue          -- Figure D.16
                                */
PROCESS     rcv_queue;
CONTEXT have_room,q_empty;
STATES      0-3;
```

```
REND  rcv_vdlc?write_pkt, playout_vdlc?scan,
      playout_vdlc!pkt_in, playout_vdlc!queue_empty,
      playout_vdlc?read_pkt;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  rcv_vdlc?write_pkt,
1:0     WHEN  {have_room == TRUE} [push_pkt();],
1:0     WHEN  {have_room != TRUE} [drop_pkt();],
0:2     WHEN  playout_vdlc?scan,
2:0     WHEN  {q_empty == TRUE} playout_vdlc!queue_empty,
2:0     WHEN  {q_empty != TRUE} playout_vdlc!pkt_in,
0:3     WHEN  playout_vdlc?read_pkt,
3:0     WHEN  [set_pkt_length();pop_pkt();].



/*
      Playout VDLC -- Figure D.17 (a)
                                    */
PROCESS     playout_vdlc;
CONTEXT current_time, eq_term, ibt_last, ibt_term, idle_term, pkt_length, playout_time, rseq_term, seq_term;
STATES      0-6,10-14,20-26,30-31,40-43;
REND  hi_layer_term?pl_vdlc_data_request_term,
      hi_layer_term!pl_vdlc_flags_indication,
      hi_layer_term!pl_vdlc_one_flag_indication,
      hi_layer_term!pl_vdlc_idle_indication,
      hi_layer_term!pl_vdlc_data_indication,
      rcv_queue!scan, rcv_queue?pkt_in,rcv_queue?queue_empty,
      rcv_queue!read_pkt,
      user_term?start, user_term?stop, clock?time;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  user_term?start [rseq_term=0;ibt_last=0xf;],
1:0     WHEN  user_term?stop,
1:2     WHEN  hi_layer_term?pl_vdlc_data_request_term,
2:3     WHEN  rcv_queue!scan,
3:1     WHEN  rcv_queue?queue_empty * hi_layer_term!pl_vdlc_idle_indication,
3:4     WHEN  rcv_queue?pkt_in * rcv_queue!read_pkt,
4:6     WHEN  {eq_term == 1},
4:5     WHEN  {eq_term == 0},
5:40    WHEN  {seq_term>0},
5:30    WHEN  {seq_term == 0},
6:20    WHEN  {seq_term>0},
6:10    WHEN  {seq_term == 0},

/*
      -- Figure D.17 (b) --

   First packet of a long frame
                .       */
```

```
10:11   WHEN  clock?time,
11:1    WHEN  {current_time >playout_time},
11:12   WHEN  {current_time <=playout_time},
12:11   WHEN  {current_time <playout_time} clock?time,
12:13   WHEN  {current_time == playout_time },
13:14   WHEN  {pkt_length != 0} hi_layer_term!pl_vdlc_data_indication [rseq_term=1; get_idle_octet();],
13:14   WHEN  {pkt_length == 0} hi_layer_term!pl_vdlc_idle_indication [get_idle_octet();],
14:1    WHEN  {idle_term == FALSE},
14:1    WHEN  {idle_term == TRUE} [ibt_last=ibt_term;],

/*
        -- Figure D.17 (c) --

      All packets except the first and the last of a long frame
                        */
20:26   WHEN  {seq_term == rseq_term} hi_layer_term!pl_vdlc_data_indication [rseq_increment();get_idle_octet(
20:21   WHEN  {seq_term != rseq_term},
21:22   WHEN  {ibt_term <= 1} hi_layer_term!pl_vdlc_flags_indication [rseq_term = seq_term;],
21:22   WHEN  {ibt_term > 1} hi_layer_term!pl_vdlc_one_flag_indication [rseq_term = seq_term;],
22:23   WHEN  clock?time,
23:1    WHEN  {current_time > playout_time },
23:24   WHEN  {current_time <= playout_time },
24:22   WHEN  {current_time < playout_time } clock?time,
24:25   WHEN  {current_time == playout_time },
25:26   WHEN  {pkt_length != 0} hi_layer_term!pl_vdlc_data_indication
                [rseq_increment(); get_idle_octet();],
25:26   WHEN  {pkt_length == 0} hi_layer_term!pl_vdlc_idle_indication [rseq_term=0;get_idle_octet();],
26:1    WHEN  {idle_term == FALSE},
26:1    WHEN  {idle_term == TRUE} [ibt_last=ibt_term;],
/*
        -- Figure D.17 (d) --
      This section applies to idle_update packets and to the
      first packet after the transition from FLAG_IDLE state to
      PACKETIZE state
                        */

30:31   WHEN  {pkt_length != 0} hi_layer_term!pl_vdlc_data_indication
                [rseq_term=1; get_idle_octet();],
30:31   WHEN  {pkt_length == 0} hi_layer_term!pl_vdlc_idle_indication [rseq_term=0;get_idle_octet();],
31:1    WHEN  {idle_term == FALSE},
31:1    WHEN  {idle_term == TRUE} [ibt_last=ibt_term;],

/*
        -- Figure D.17 (e) --
       A short frame and the last frame from a long frame
                        */

40:43   WHEN  {seq_term == rseq_term} hi_layer_term!pl_vdlc_data_indication [rseq_increment();get_idle_octet(
40:41   WHEN  {seq_term != rseq_term},
41:42   WHEN  {ibt_term <= 1} hi_layer_term!pl_vdlc_flags_indication [rseq_term = seq_term;],
41:42   WHEN  {ibt_term > 1} hi_layer_term!pl_vdlc_one_flag_indication [rseq_term = seq_term;],
42:43   WHEN  {pkt_length != 0} hi_layer_term!pl_vdlc_data_indication
                [rseq_increment(); get_idle_octet();],
42:43   WHEN  {pkt_length == 0} hi_layer_term!pl_vdlc_idle_indication [rseq_term=0;get_idle_octet();],
43:1    WHEN  {idle_term == FALSE},
```

```
43:1    WHEN  {idle_term == TRUE} [ibt_last=ibt_term;].

/*
        High layer entity at the terminating end

        -- Figure D.18 (a) --
                        */

PROCESS     hi_layer_term;
CONTEXT     invert_term, n_ones_term, pkt_length, ibt_term, n_flag_term, nbits_term,
        playout_octet, out_bit, arriving_bit, flag_cnt;
STATES      0-2,100-105,200-202,300,310-316,350-351,360-372;
REND    playout_vdlc?pl_vdlc_idle_indication, playout_vdlc?pl_vdlc_data_indication,
        playout_vdlc!pl_vdlc_data_request_term,
        playout_vdlc?pl_vdlc_one_flag_indication,
        playout_vdlc?pl_vdlc_flags_indication,
        user_term?start, user_term?stop,
        mgmt_term!error,
        serial_to_parallel_conv?ready,
        serial_to_parallel_conv!idle,
        serial_to_parallel_conv!hdlc_data,
        serial_to_parallel_conv!bit_in;
INITIAL STATE 0;
TRANSITIONS

0:1     WHEN user_term?start [n_ones_term=0;],
1:0     WHEN user_term?stop,
1:2     WHEN playout_vdlc!pl_vdlc_data_request_term,
2:300   WHEN playout_vdlc?pl_vdlc_data_indication * serial_to_parallel_conv!hdlc_data
            [nbits_term =0;],
2:100   WHEN playout_vdlc?pl_vdlc_flags_indication
            * serial_to_parallel_conv!hdlc_data [n_ones_term=0;flag_cnt=FLAG_MIN;],
2:100   WHEN playout_vdlc?pl_vdlc_one_flag_indication
            * serial_to_parallel_conv!hdlc_data [n_ones_term=0;flag_cnt=1;],
2:200   WHEN playout_vdlc?pl_vdlc_idle_indication [n_ones_term=0;],
/*
        -- Figure D.18 (b) --

       PLAY_FLAG STATE

      Send one or more flags bit by bit
                                */
100:1   WHEN {flag_cnt <=0 },
100:101 WHEN {flag_cnt>0} [playout_octet = 0x7e; nbits_term =0;],
101:102         WHEN {nbits_term < 8} [out_bit = playout_octet %2;
                playout_octet = playout_octet/2;],
102:101         WHEN  serial_to_parallel_conv?ready *
            serial_to_parallel_conv!bit_in [nbits_term++;],
101:100 WHEN {nbits_term >= 8} [flag_cnt--;],
/*
        -- Figure D.18 (c) --

                                */
200:201         WHEN {ibt_term > 2},
201:1   WHEN serial_to_parallel_conv?ready * serial_to_parallel_conv!idle,
```

```
200:100         WHEN {ibt_term <= 2} serial_to_parallel_conv!hdlc_data [flag_cnt=1;],
/*
        -- Figure D.18 (d) --

      PLAY_HDLC_DATA STATE

      Send data octet bit by bit to the serial to parallel converter
         without bit stuffing
*/
300:310         WHEN {ibt_term == 0 },
300:350         WHEN {ibt_term != 0},
350:360         WHEN {ibt_term == 1},
350:351         WHEN {ibt_term != 1},
351:1   WHEN {ibt_term != 2} mgmt_term!error,
351:370         WHEN {ibt_term == 2},

     /*
        FLAG_PKT
      */

370:1   WHEN {pkt_length <= 0 },
370:371         WHEN {pkt_length > 0 } [get_octet(); pkt_length--;nbits_term=0;],
371:372         WHEN {nbits_term < 8} [out_bit = playout_octet %2;
                playout_octet = playout_octet/2;],
372:371         WHEN  serial_to_parallel_conv?ready *
             serial_to_parallel_conv!bit_in [nbits_term++;],
371:370 WHEN {nbits_term >= 8},
/*
        -- Figure D.18 (e) --

      PLAY_HDLC_DATA STATE (cont)

      Send data octet bit by bit to the serial to parallel converter
         with bit stuffing and append flag
                                  */

310:100         WHEN {pkt_length == 0 } [n_ones_term=0; flag_cnt =1;],
310:311         WHEN {pkt_length != 0 } [get_octet(); pkt_length--; nbits_term=0;],
311:312         WHEN {nbits_term < 8} [arriving_bit = playout_octet %2;
                playout_octet = playout_octet/2;],
312:313         WHEN {arriving_bit ==1 } [n_ones_term++;],
312:315         WHEN {arriving_bit != 1 } [n_ones_term=0;],
313:315         WHEN {n_ones_term != 5},
313:314         WHEN {n_ones_term ==5} [out_bit =0; n_ones_term=0;],
314:315         WHEN serial_to_parallel_conv?ready * serial_to_parallel_conv!bit_in,
315:316         WHEN  [out_bit = arriving_bit;],
316:311         WHEN  serial_to_parallel_conv?ready *
             serial_to_parallel_conv!bit_in [nbits_term++;],
311:310 WHEN {nbits_term >= 8},

/*
        -- Figure D.18 (f) --

      PLAY_HDLC_DATA STATE (cont)
```

```
        Send data octet bit by bit to the serial to parallel converter
          with bit stuffing
                                        */

360:1   WHEN {pkt_length == 0 },
360:361         WHEN {pkt_length != 0} [get_octet(); pkt_length--;nbits_term=0;],
361:362         WHEN {nbits_term < 8} [arriving_bit = playout_octet %2;
                playout_octet = playout_octet/2;],
362:363         WHEN {arriving_bit ==1 } [n_ones_term++;],
362:365         WHEN {arriving_bit !=1 } [n_ones_term=0;],
363:365         WHEN {n_ones_term != 5},
363:364         WHEN {n_ones_term ==5} [out_bit =0; n_ones_term =0;],
364:365         WHEN serial_to_parallel_conv?ready * serial_to_parallel_conv!bit_in,
365:366         WHEN  [out_bit = arriving_bit;],
366:361         WHEN  serial_to_parallel_conv?ready *
              serial_to_parallel_conv!bit_in [nbits_term++;],
361:360 WHEN {nbits_term >= 8}.



/*
        -- Figure D.19 (a) --
                                        */

PROCESS       serial_to_parallel_conv;
CONTEXT speed, invert_term, out_octet, bit_cnt_term,c_count_term ;
STATES        0-5,10-16,20-24,30-34, 40-44,50-54,60-64;
REND  user_term?start, user_term?stop, user_term!octet_in, user_term?ready,
        hi_layer_term!ready,
        hi_layer_term?idle,
        hi_layer_term?hdlc_data,
        hi_layer_term?bit_in;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN user_term?start * hi_layer_term!ready [bit_cnt_term=0;c_count_term=0;],
1:2     WHEN {speed < 56},
2:0     WHEN user_term?stop,
2:10    WHEN hi_layer_term?hdlc_data [out_octet=0;],
2:20    WHEN hi_layer_term?idle,
1:3     WHEN {speed >= 56},
3:0     WHEN user_term?stop,
3:4     WHEN {speed == 56},
4:30    WHEN hi_layer_term?hdlc_data [out_octet=0;],
4:40    WHEN hi_layer_term?idle,
3:5     WHEN {speed != 56},
5:0     WHEN user_term?stop,
5:50    WHEN hi_layer_term?hdlc_data [out_octet=0;],
5:60    WHEN hi_layer_term?idle,

/*
        -- Figure D.19 (b) --
```

```
    HDLC operation at Speed less than 56 kbit/s
                                    */

10:11   WHEN hi_layer_term?bit_in [bit_cnt_term++;],
10:20   WHEN hi_layer_term?idle,
11:12   WHEN {invert_term == TRUE} [invert_bit();],
11:12   WHEN {invert_term != TRUE},
12:13   WHEN {bit_cnt_term <=1 } [out_octet=1; bit_cnt_term++;],
12:13   WHEN {bit_cnt_term > 1 },
13:10   WHEN {bit_cnt_term < 7} hi_layer_term!ready
            [out_octet = 2 * out_octet +out_bit;],
13:14   WHEN {bit_cnt_term >= 7 } [out_octet = 2 * out_octet +out_bit;],
14:15   WHEN [out_octet = 2 * out_octet +1;],
15:16   WHEN user_term?ready * user_term!octet_in [c_count_term++;],
16:15   WHEN {c_count_term < CMAX },
16:10   WHEN {c_count_term >= CMAX } hi_layer_term!ready [bit_cnt_term=0; out_octet=0; c_count_term =0;],

/*
        -- Figure D.19 (c) --

    Idle Operation at less than 56 kbit/s
                                    */

20:23   WHEN {bit_cnt_term ==0 } [get_last_ibt_octet();],
20:21   WHEN {bit_cnt_term != 0},
21:22   WHEN {bit_cnt_term <= 7} [out_octet = 2 * out_octet + 1;],
22:21   WHEN [bit_cnt_term ++;],
21:23   WHEN {bit_cnt_term > 7 } user_term?ready * user_term!octet_in
            [c_count_term=1;bit_cnt_term=0;],
23:24   WHEN {c_count_term < CMAX } user_term?ready *user_term!octet_in,
24:23   WHEN [c_count_term++;],
23:2    WHEN {c_count_term >= CMAX } [c_count_term=0; out_octet=0;],

/*
        -- Figure D.19 (d) --

    HDLC operation at Speed  of 56 kbit/s
                                    */

30:31   WHEN hi_layer_term?bit_in [bit_cnt_term++;],
30:40   WHEN hi_layer_term?idle,
31:32   WHEN {invert_term == TRUE} [invert_bit();],
31:32   WHEN {invert_term != TRUE},
32:30   WHEN {bit_cnt_term <= 6} hi_layer_term!ready
          [out_octet=2*out_octet+ out_bit ;],
32:33   WHEN {bit_cnt_term > 6 } [out_octet = 4 * out_octet + 2 * out_bit+1;],
33:34   WHEN user_term?ready * user_term!octet_in,
34:30   WHEN hi_layer_term!ready [bit_cnt_term=0; out_octet=0;],

/*
        -- Figure D.19 (e) --

    Idle Operation at 56 kbit/s
                                    */
```

```
40:43   WHEN {bit_cnt_term ==0 } [get_last_ibt_octet();],
40:41   WHEN {bit_cnt_term != 0},
41:42   WHEN {bit_cnt_term <= 7} [out_octet = 2 * out_octet + 1;],
42:41   WHEN [bit_cnt_term ++;],
41:43   WHEN {bit_cnt_term > 7 },
43:44   WHEN user_term?ready * user_term!octet_in,
44:4    WHEN hi_layer_term!ready [bit_cnt_term=0; out_octet=0;],

/*
        -- Figure D.19 (f) --

     HDLC Operation  for 64 kbit/s
                                */

50:51   WHEN hi_layer_term?bit_in [bit_cnt_term++;],
50:60   WHEN hi_layer_term?idle,
51:52   WHEN {invert_term == TRUE} [invert_bit();],
51:52   WHEN {invert_term != TRUE},
52:50   WHEN {bit_cnt_term <= 7} hi_layer_term!ready
          [out_octet=2*out_octet+ out_bit ;],
52:53   WHEN {bit_cnt_term > 7 } [out_octet=2*out_octet+out_bit;],
53:54   WHEN user_term?ready * user_term!octet_in,
54:50   WHEN hi_layer_term!ready [bit_cnt_term=0; out_octet=0;],

/*
        -- Figure D.19 (g) --

     Idle Operation at 64 kbit/s
                                */

60:63   WHEN {bit_cnt_term ==0 } [get_last_ibt_octet();],
60:61   WHEN {bit_cnt_term != 0},
61:62   WHEN {bit_cnt_term <= 7} [out_octet = 2 * out_octet + 1;],
62:61   WHEN [bit_cnt_term ++;],
61:63   WHEN {bit_cnt_term > 7 },
63:64   WHEN user_term?ready * user_term!octet_in,
64:5    WHEN hi_layer_term!ready [bit_cnt_term=0; out_octet=0;].


/*
        T_IDLE -- Figure D.20 --
                                */
PROCESS      t_idle;
CONTEXT delay;
STATES       0-2;
REND  hi_layer_orig?start, hi_layer_orig!expired, hi_layer_orig?stop,
        pre_processor?start, pre_processor?stop, clock?tick;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  hi_layer_orig?start [delay=0;],
0:1     WHEN  pre_processor?start [delay=0;],
1:2     WHEN  clock?tick  [delay++;],
2:1     WHEN  {delay < IDLE_VALUE} ,
```

```
2:0     WHEN {delay == IDLE_VALUE} hi_layer_orig!expired [delay=0;],
1:0     WHEN pre_processor?stop [delay=0;],
1:0     WHEN hi_layer_orig?stop [delay=0;].

/*
        TVDELAY_ORIG -- Figure D.21 --
                                        */
PROCESS     tvdelay_orig;
CONTEXT delay_orig;
STATES      0-1;
REND  xmtr_vdlc?start, xmtr_vdlc?stop, clock?tick;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN xmtr_vdlc?start [delay_orig=0;],
1:1     WHEN clock?tick [delay_orig++;],
1:0     WHEN xmtr_vdlc?stop[output_delay();delay_orig=0;].

/*
        TVDELAY_INTERM -- Figure D.22 --
                                        */
PROCESS     tvdelay_interm;
CONTEXT delay_interm;
STATES      0-1;
REND  interm_pt_vdlc_relay?start, interm_pt_vdlc_relay?stop,
        clock?tick;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN interm_pt_vdlc_relay?start [delay_interm=0;],
1:1     WHEN clock?tick [delay_interm++;],
1:0     WHEN interm_pt_vdlc_relay?stop[output_delay();delay_interm=0;].
```

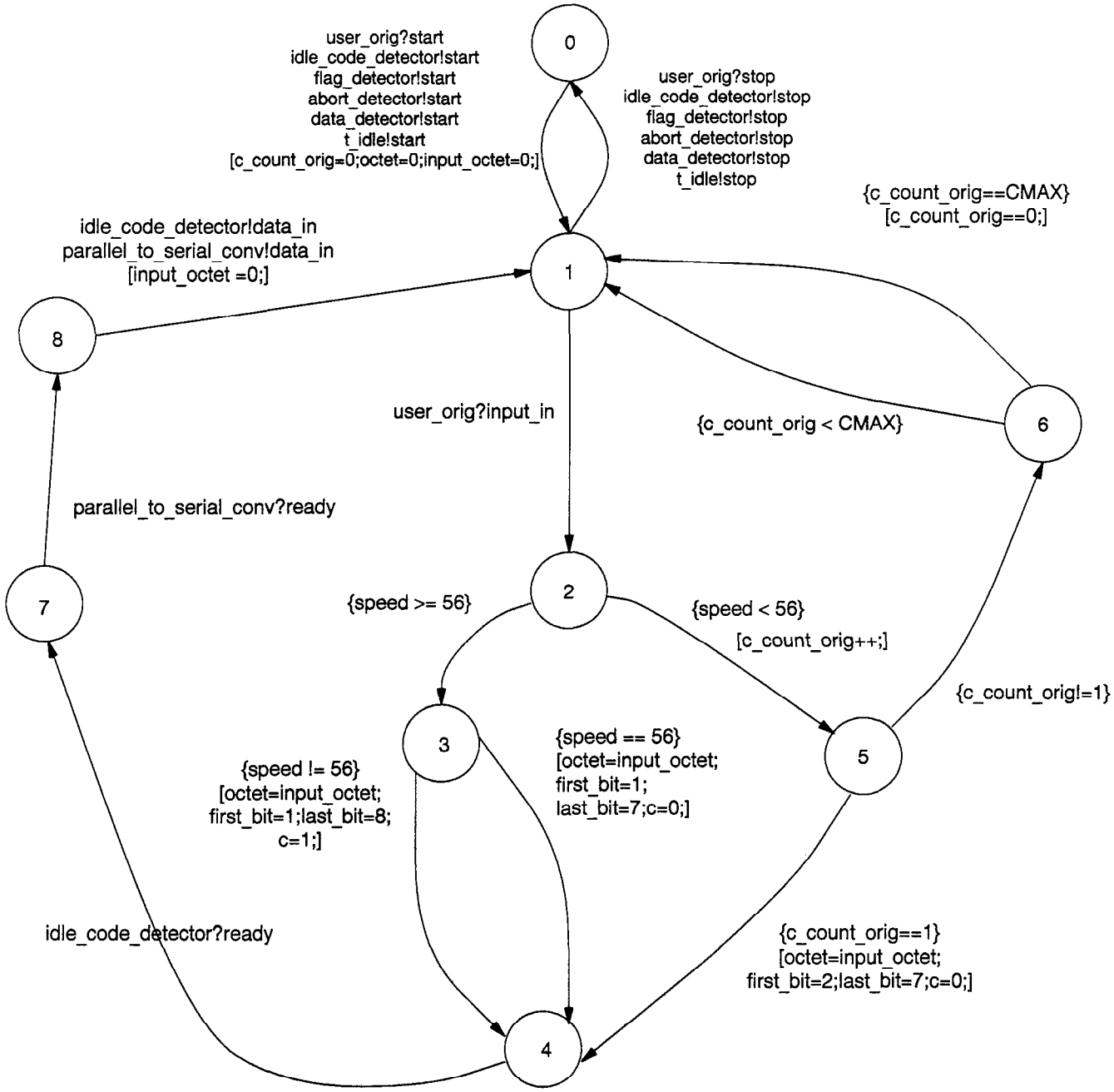


Figure D.1 – pre_processor

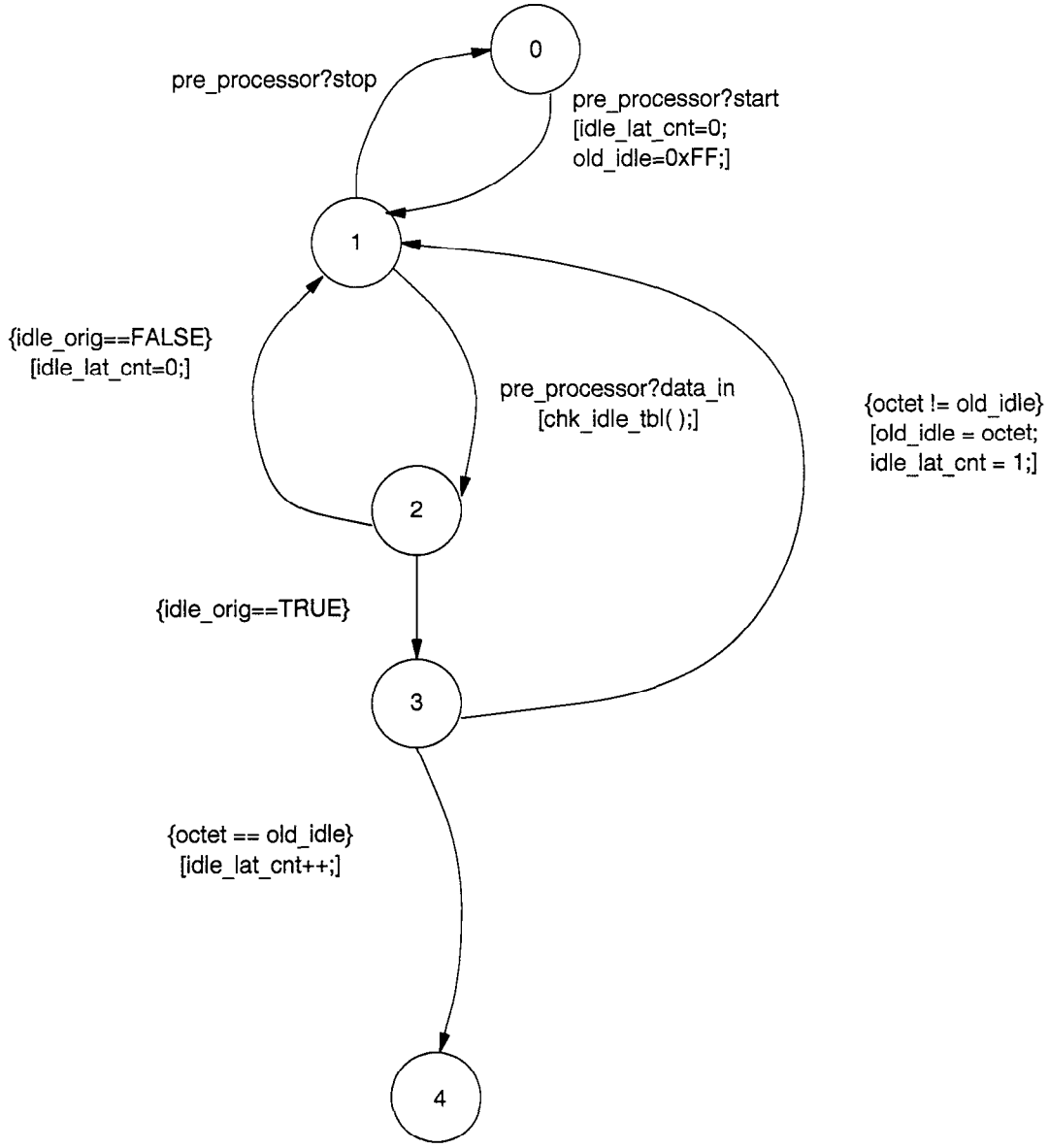


Figure D.2 – idle_code_detector
Part a

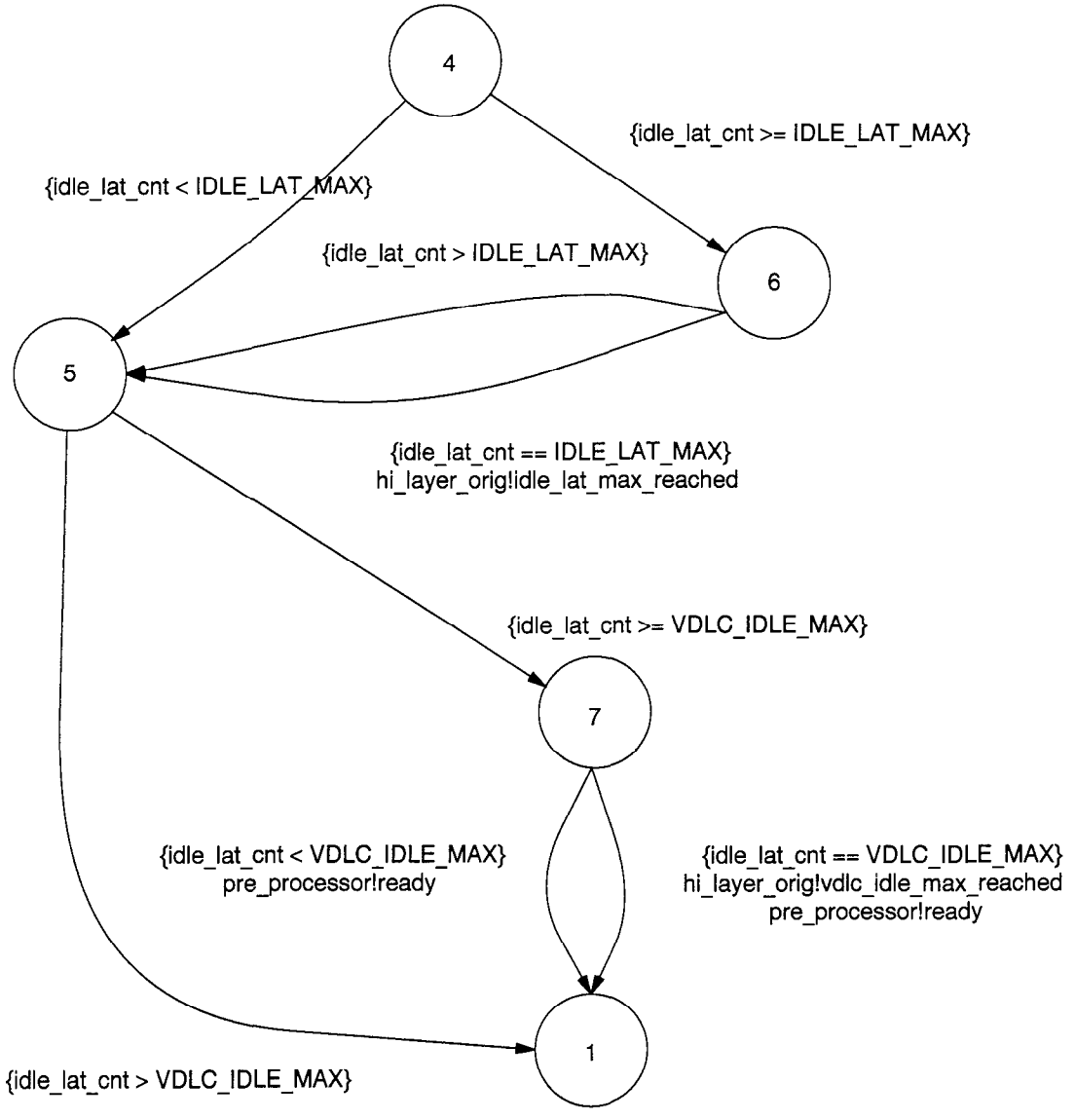


Figure D.2 (concluded) – idle_code_detector
Part b

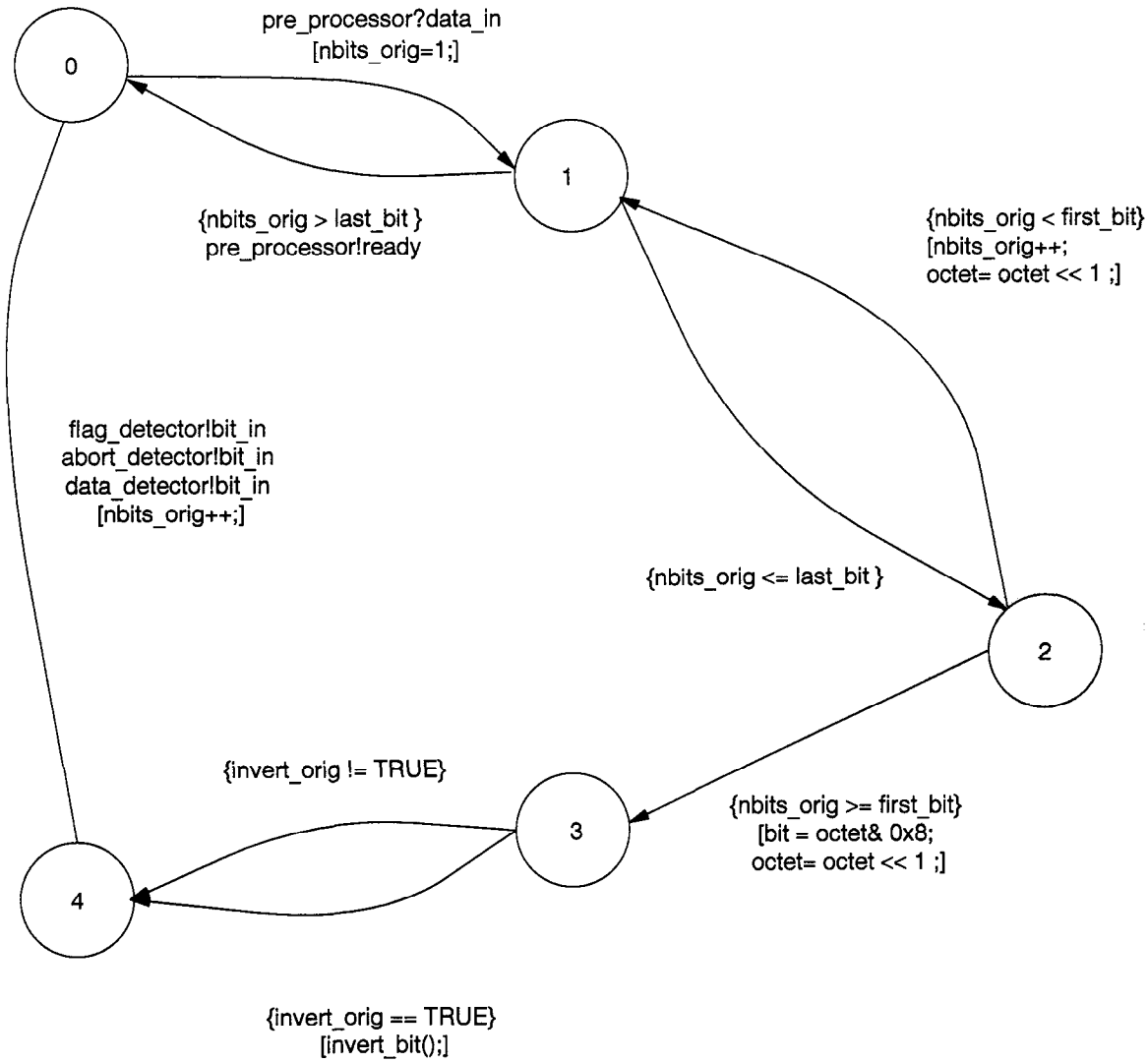


Figure D.3 – parallel_to_serial_conv

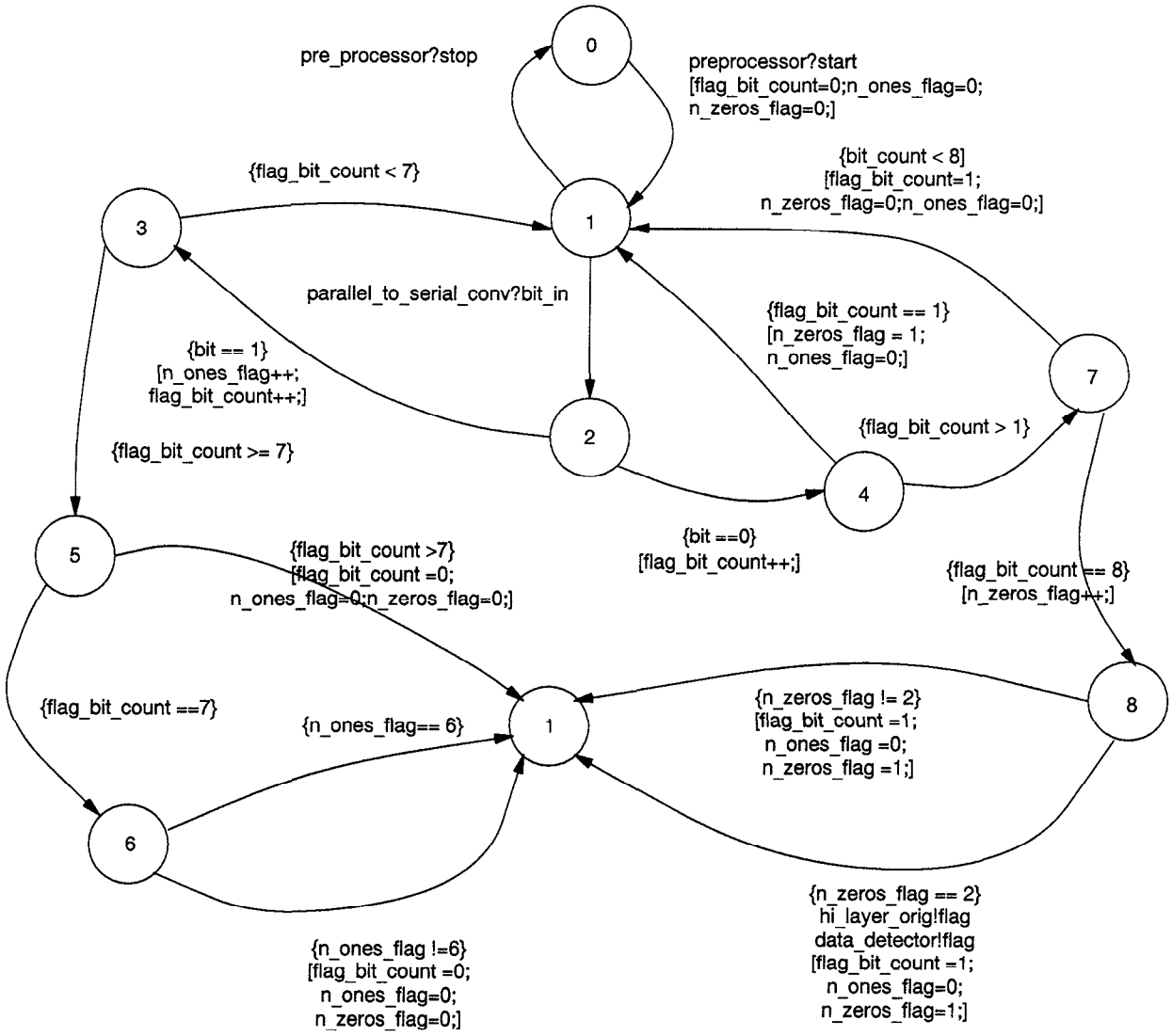


Figure D.4 – flag_detector

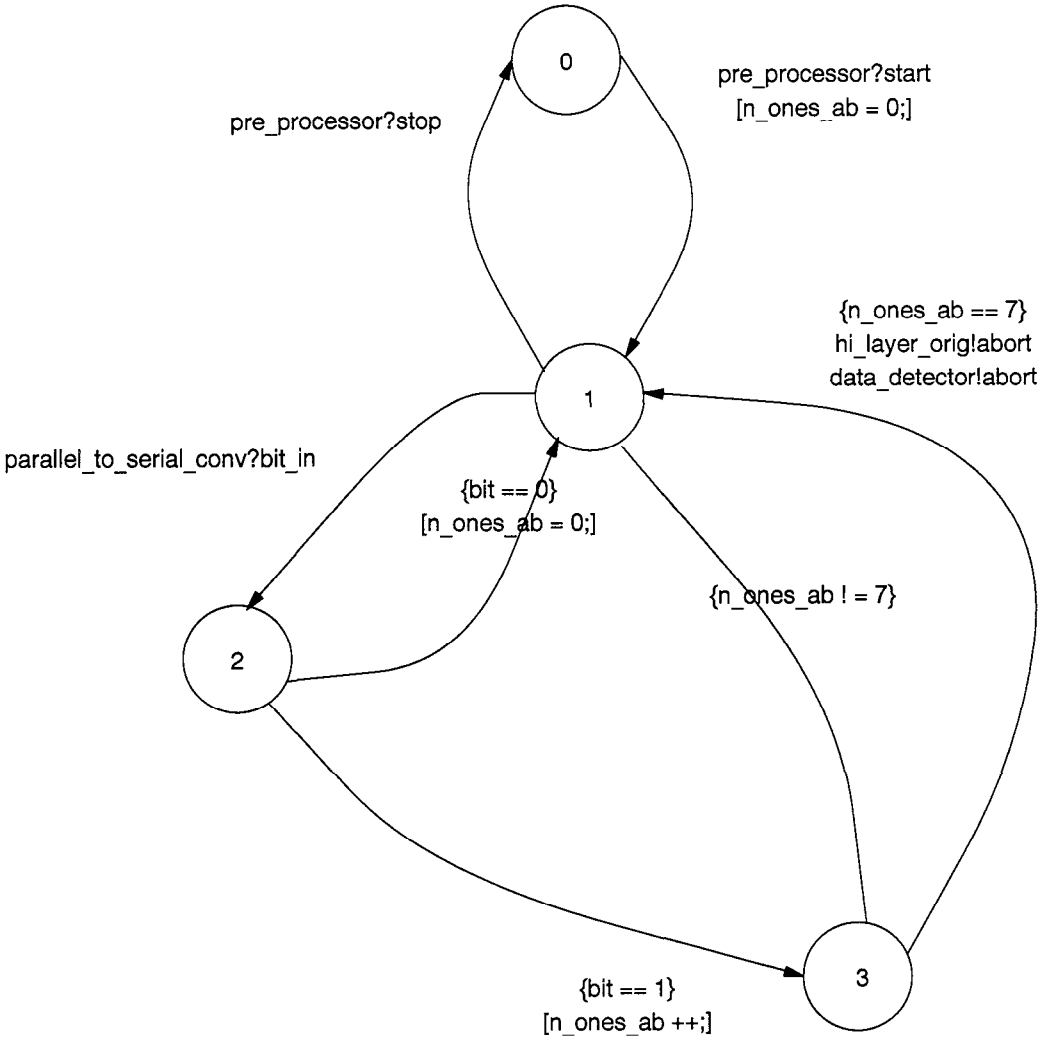


Figure D.5 – abort_detector

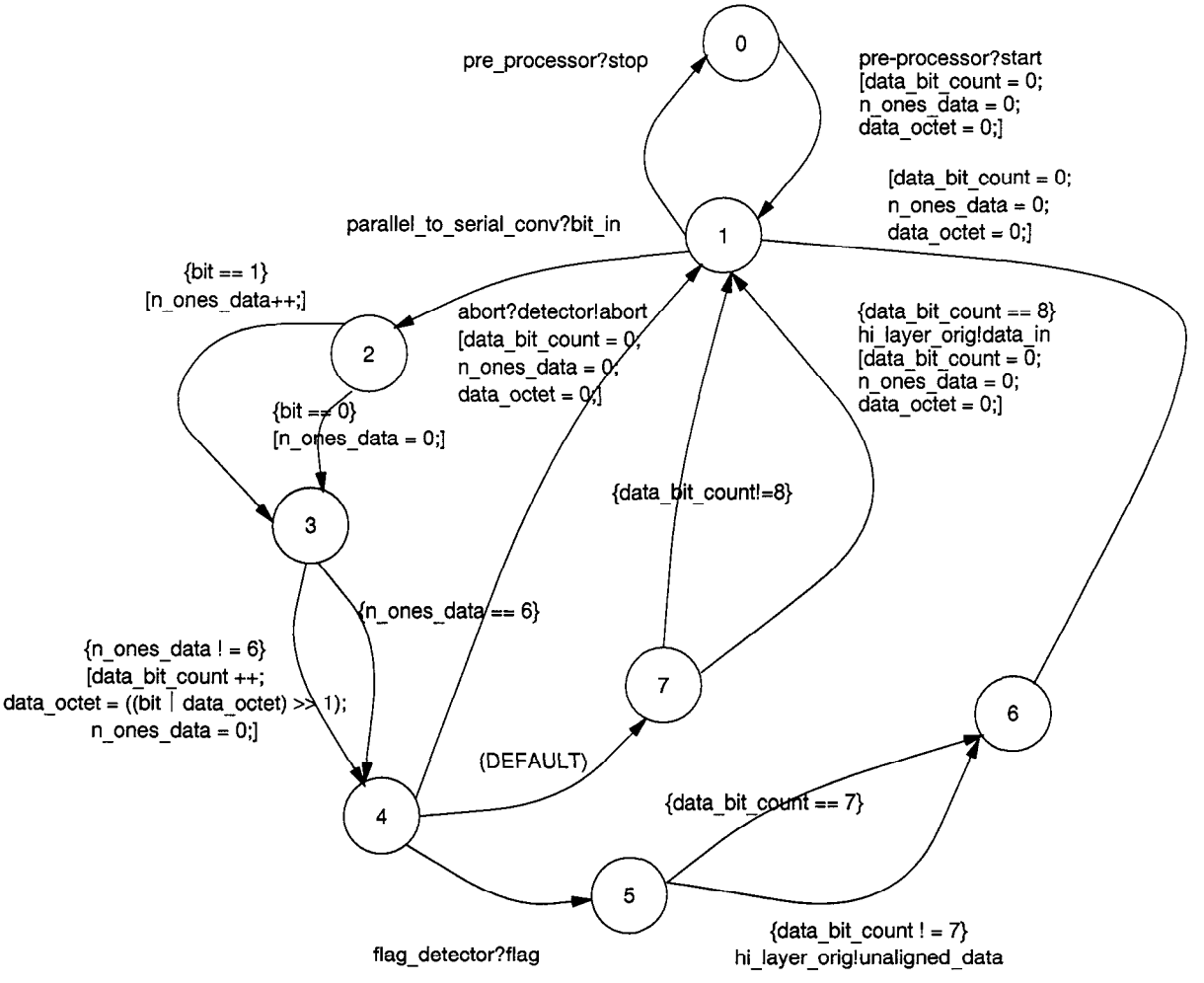


Figure D.6 – data_detector

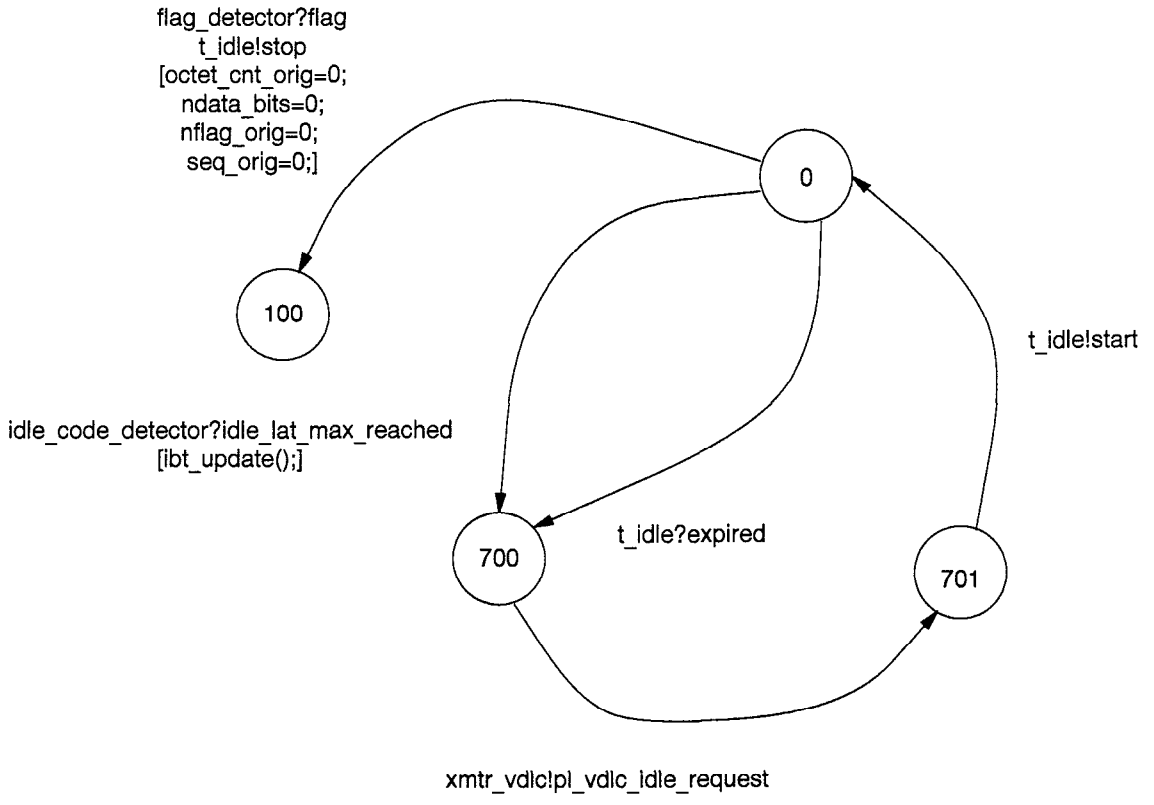


Figure D.7 – hi_layer_orig IDLE_STATE

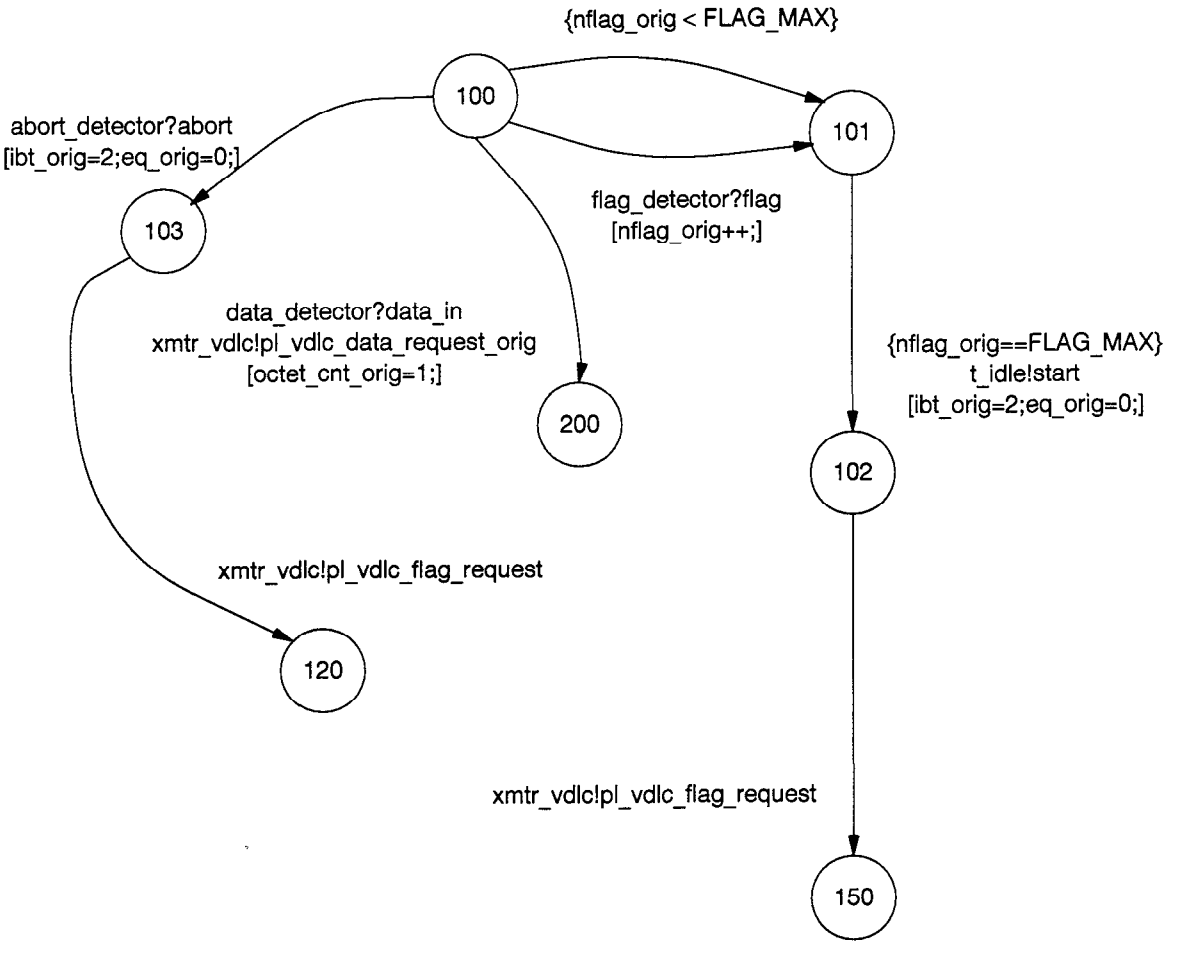


Figure D.8 – hi_layer_orig
AWAIT_STATE

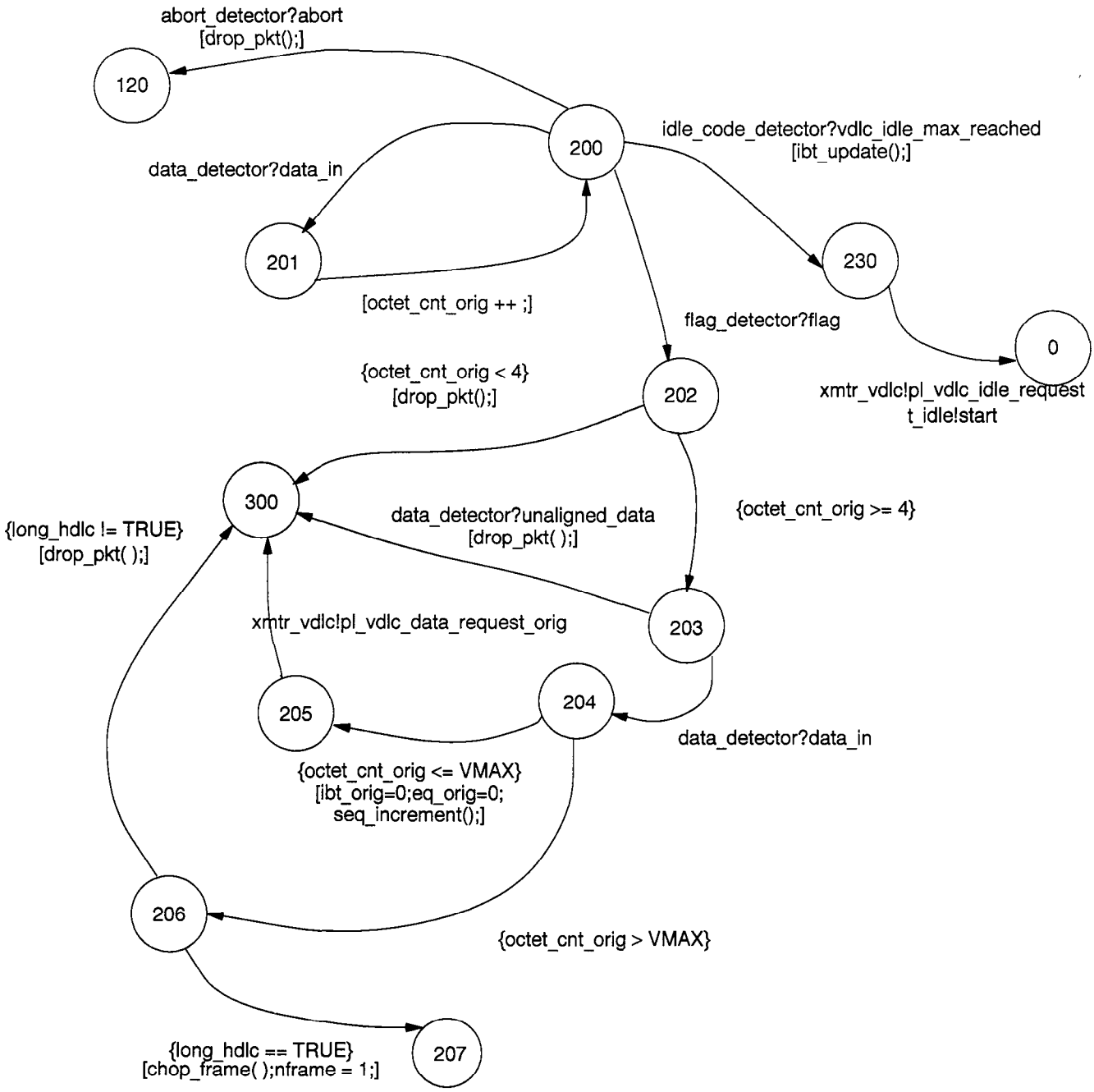


Figure D.9 – hi_layer_orig
PACKETIZE STATE
Part a

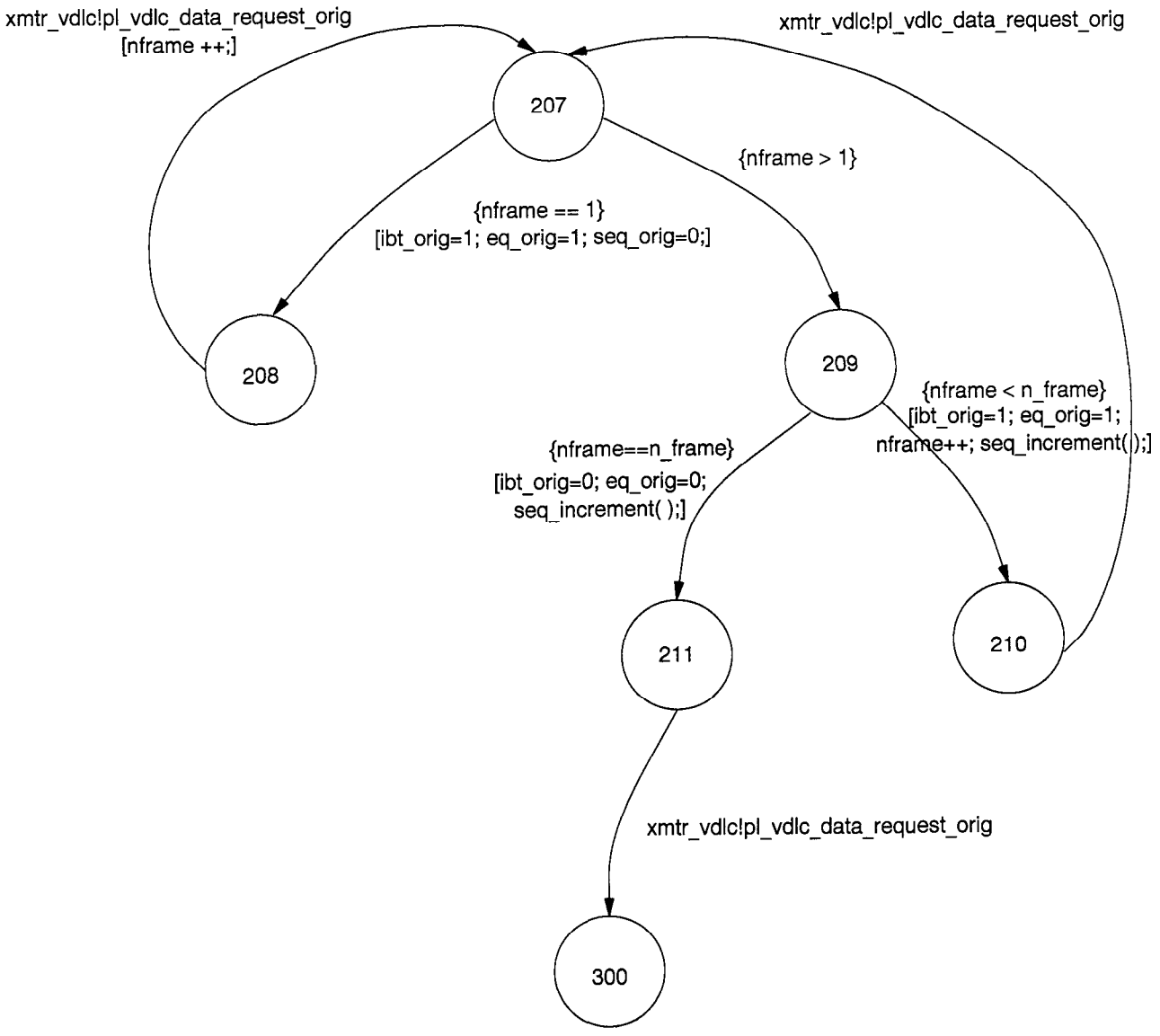


Figure D.9 (concluded) – hi_layer_orig
PACKETIZE STATE
Part b

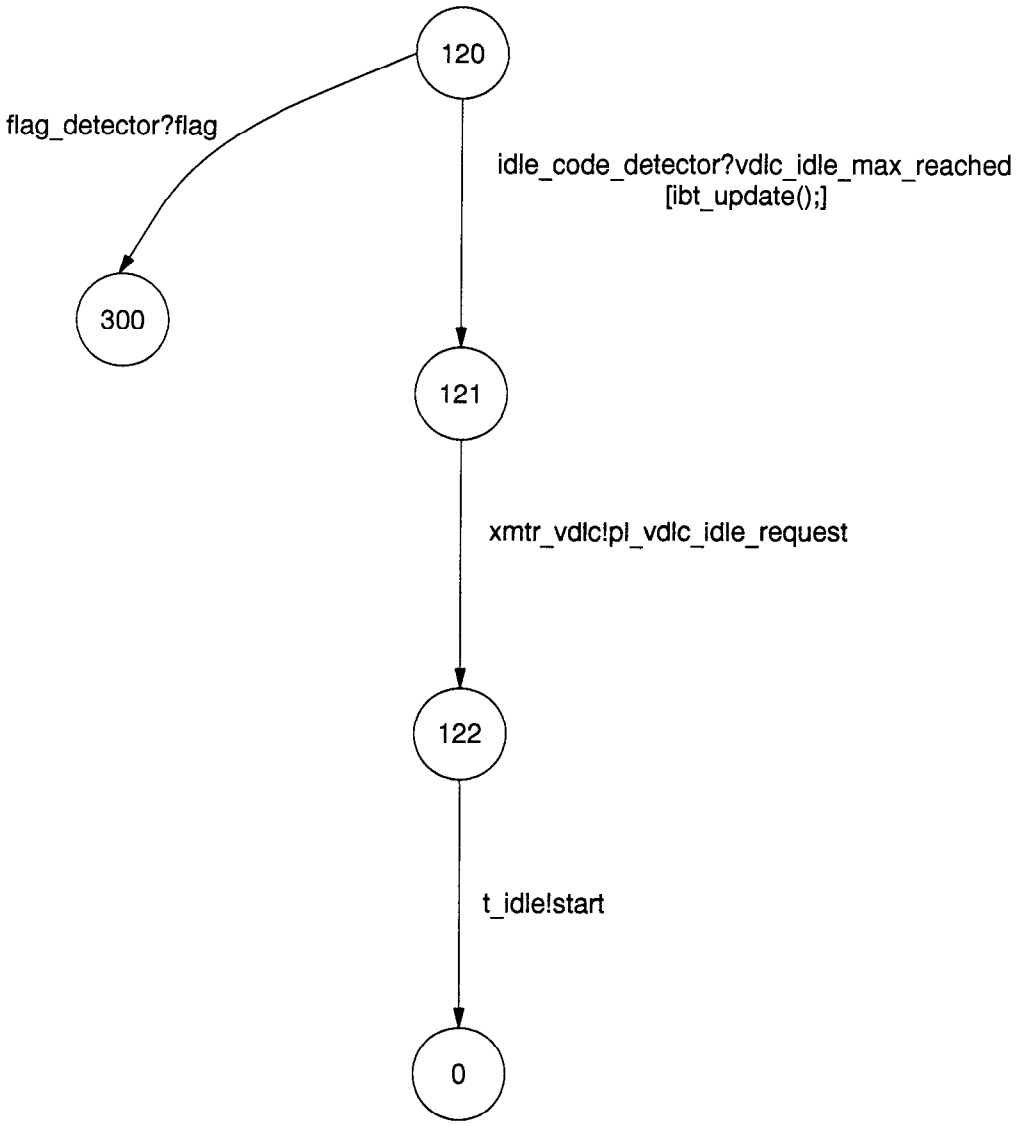


Figure D.10 – hi_layer_orig
ABORT_STATE

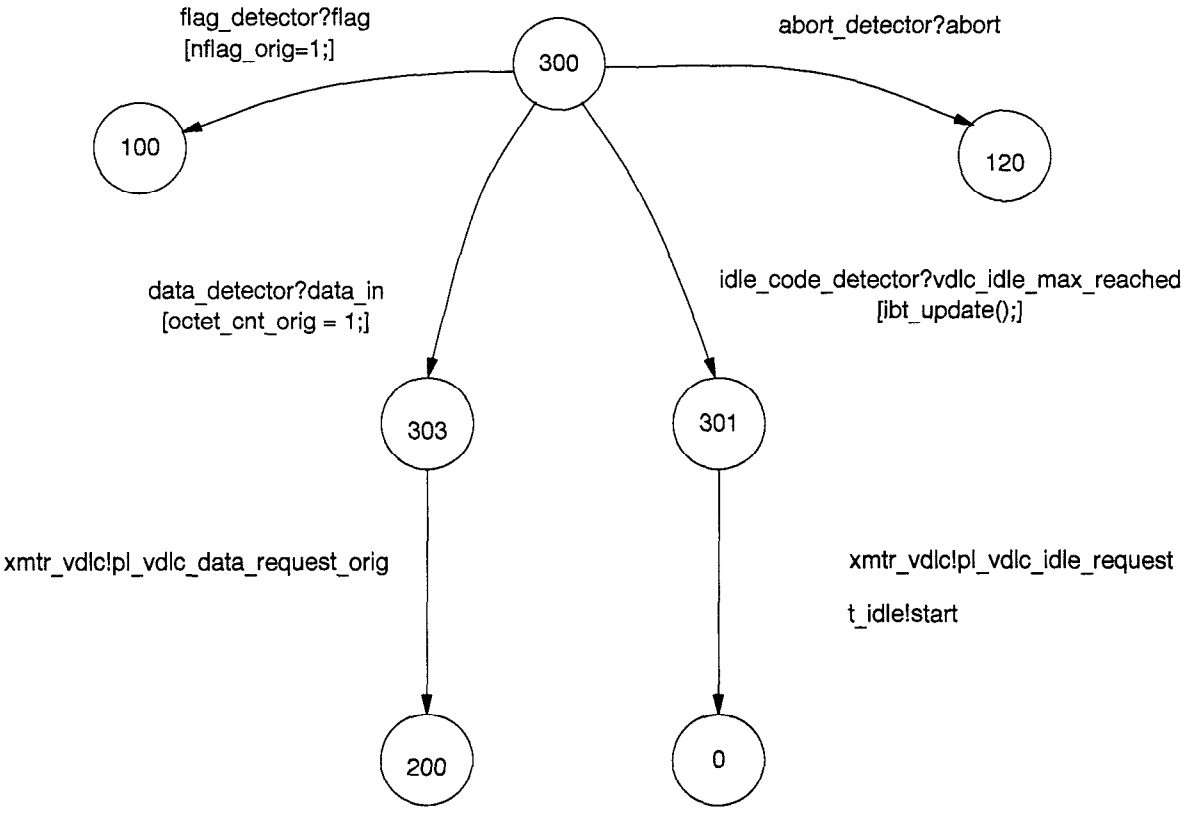


Figure D.11 – hi_layer_orig
FLAG_WAIT_STATE

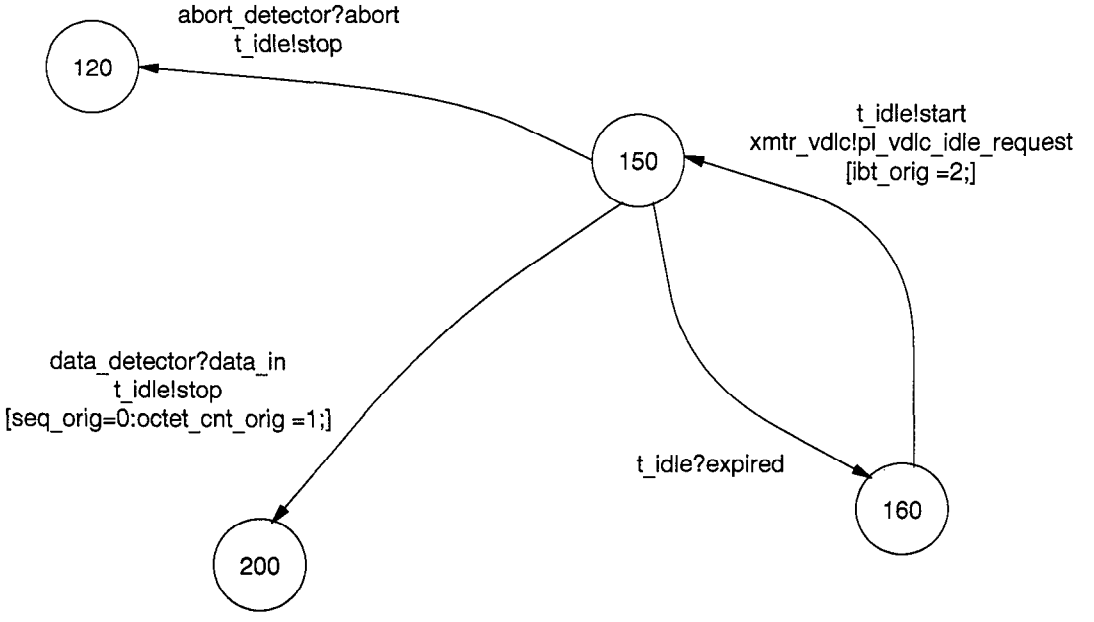


Figure D.12 – hi_layer_orig
FLAG_IDLE_STATE

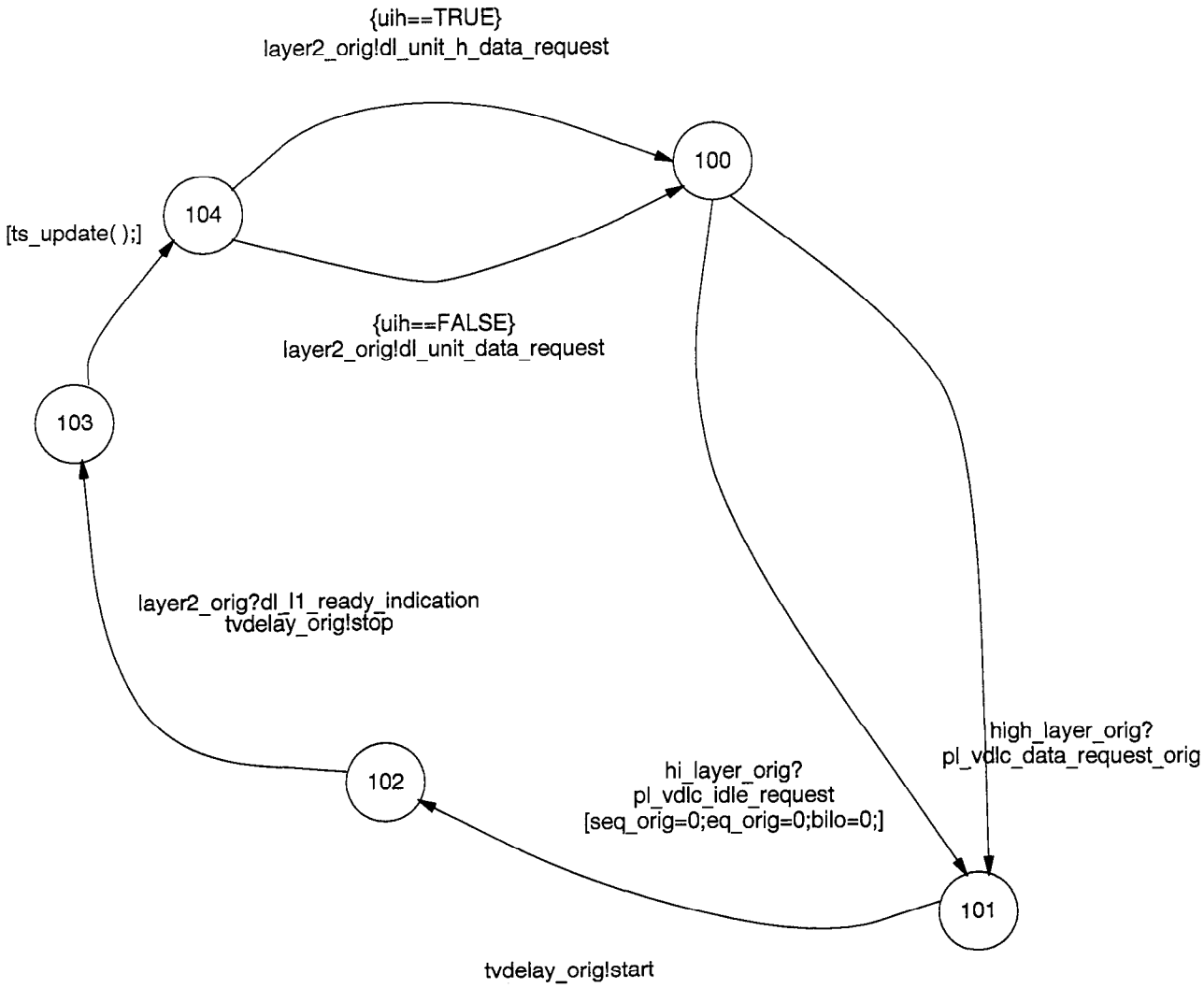


Figure D.13 – xmtr_vdlc

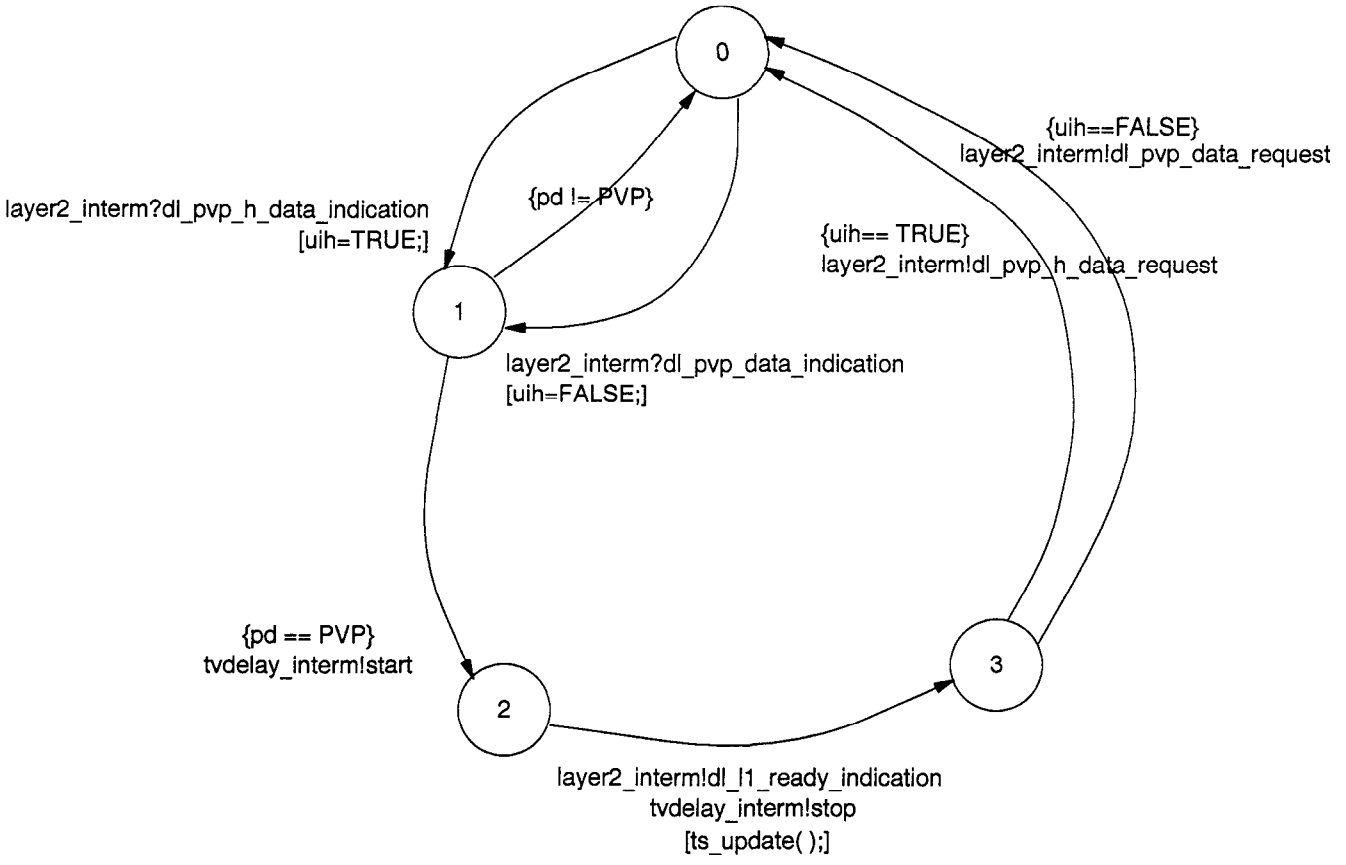


Figure D.14 – Interm_pt_vdlc_relay

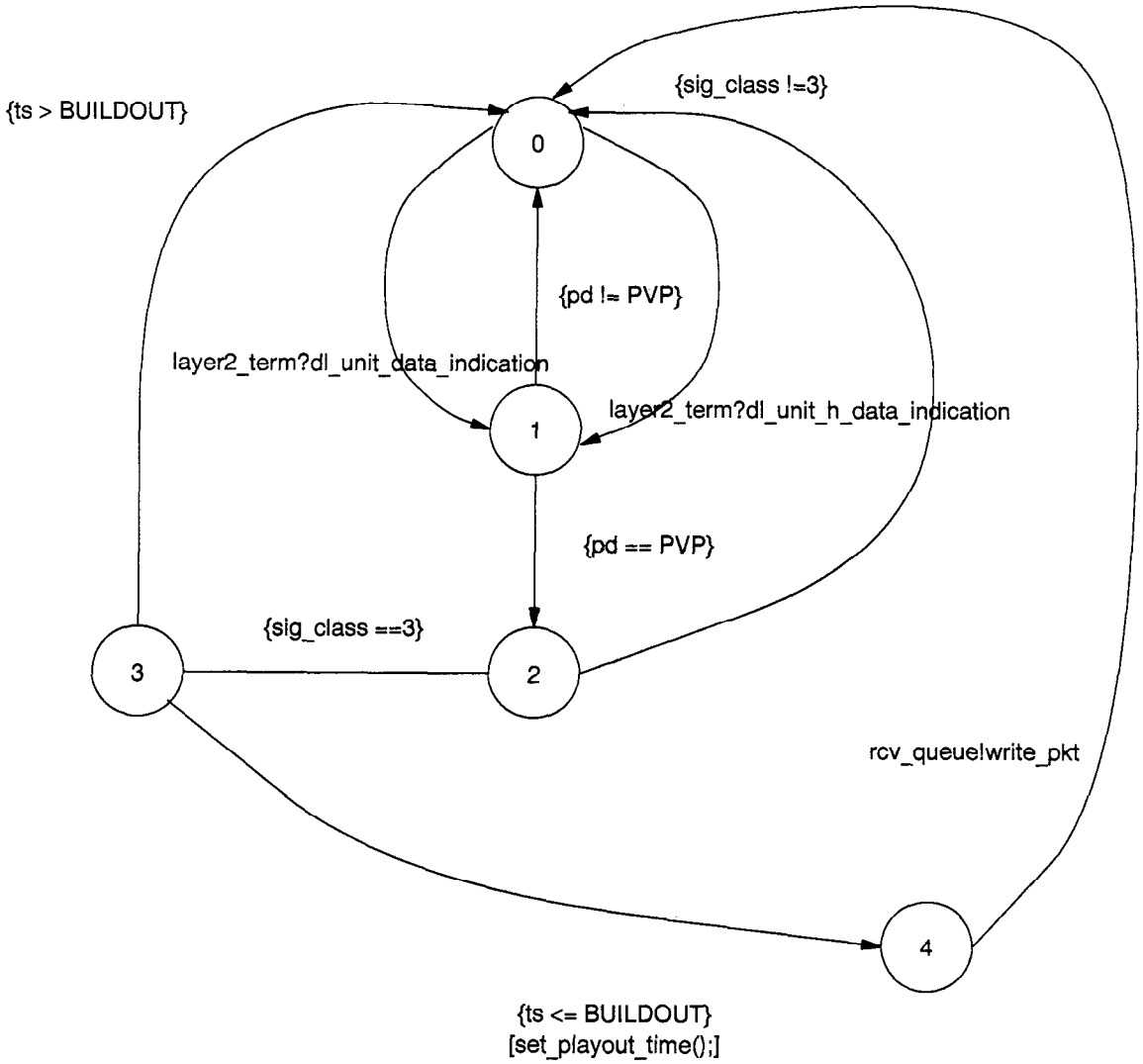


Figure D.15 – rcv_vdlc

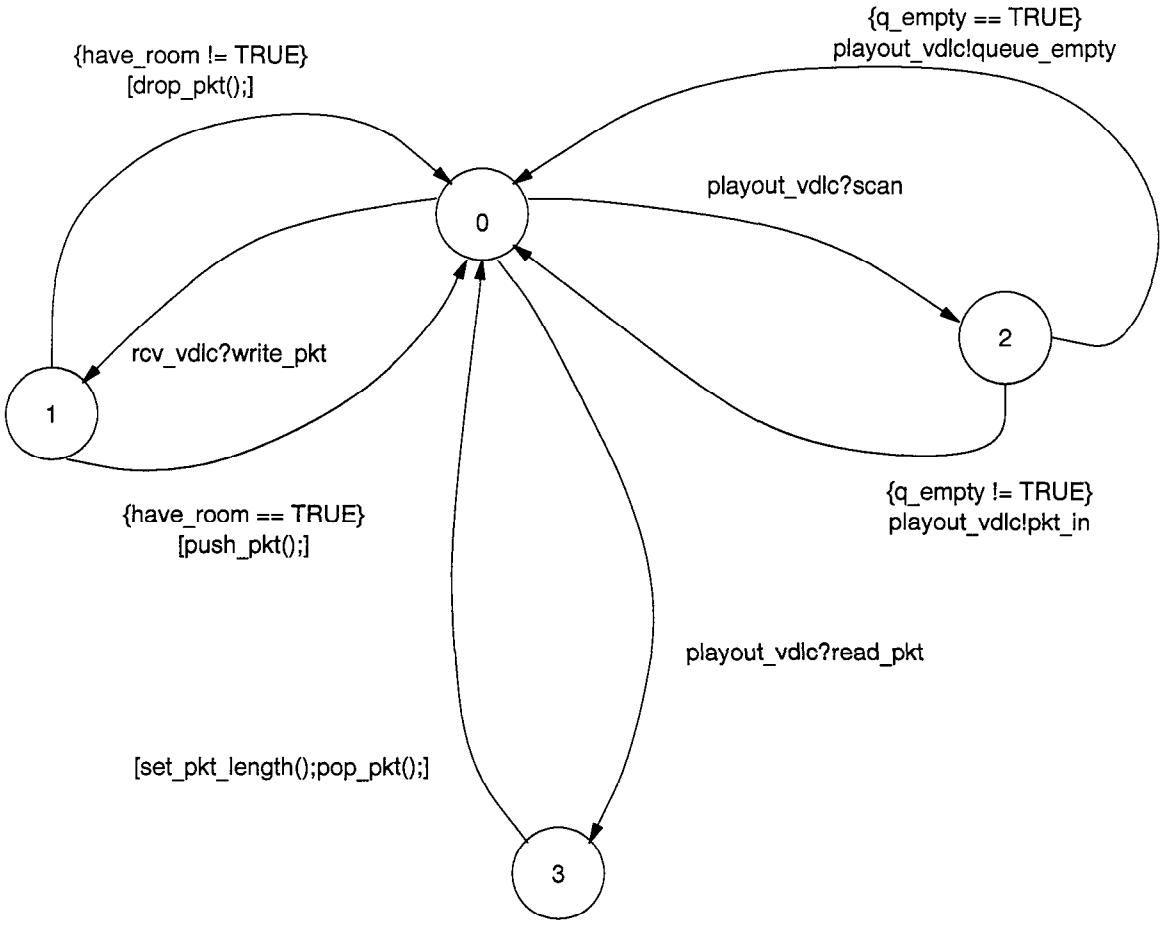


Figure D.16 – rcv_queue

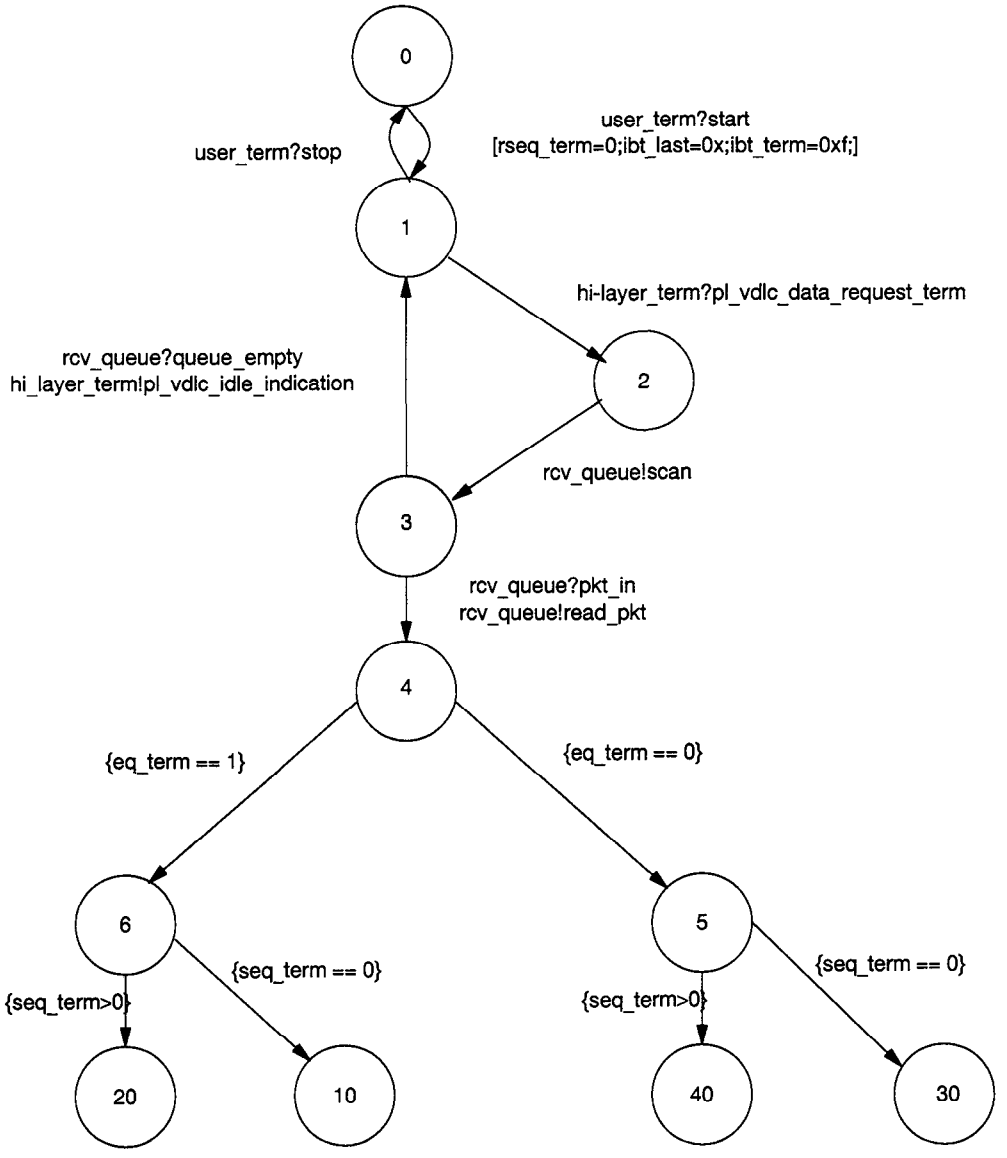


Figure D.17 – playout_vdlc
Part a

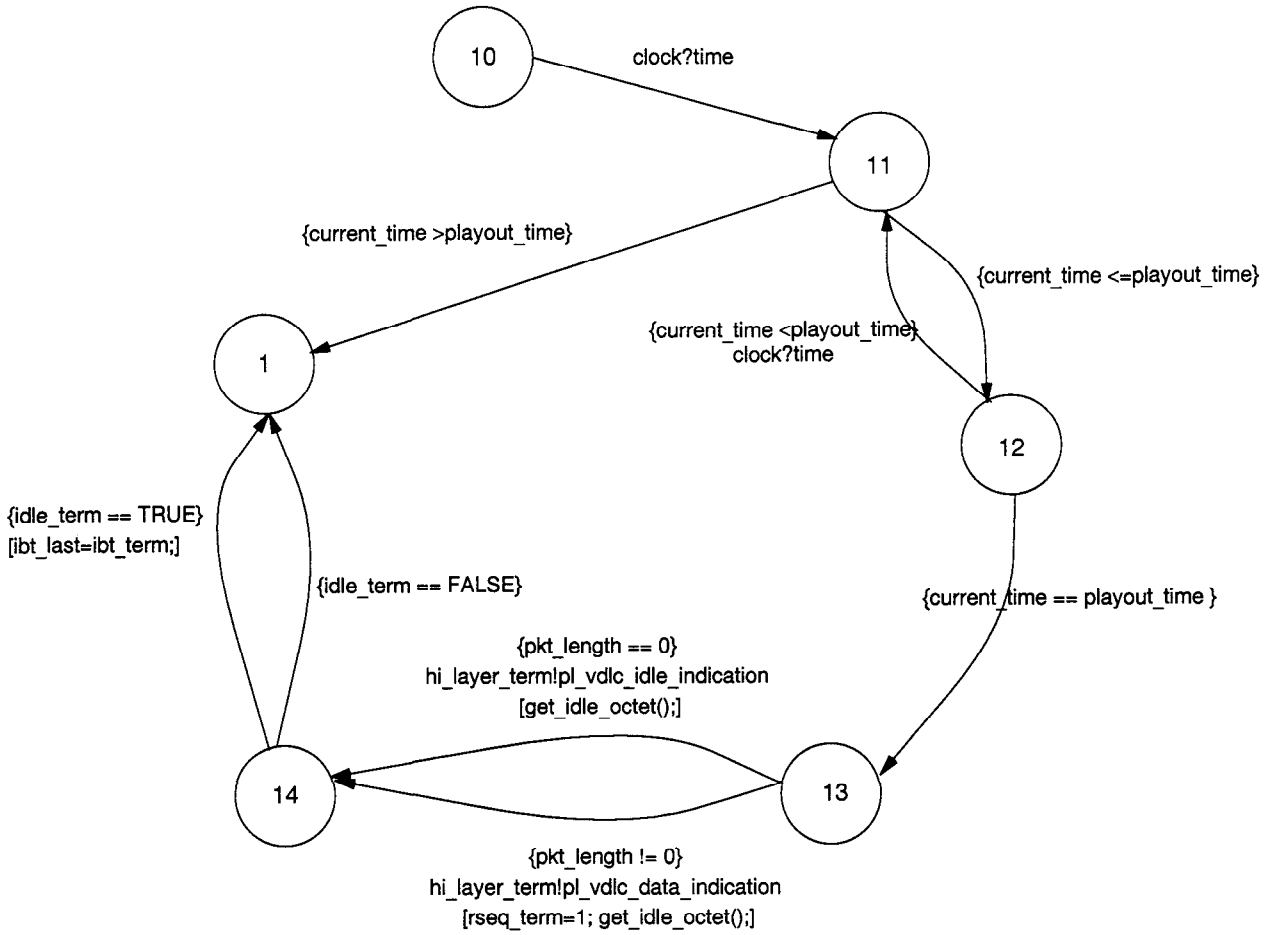


Figure D.17 (continued) – playout_vdlc
Part b

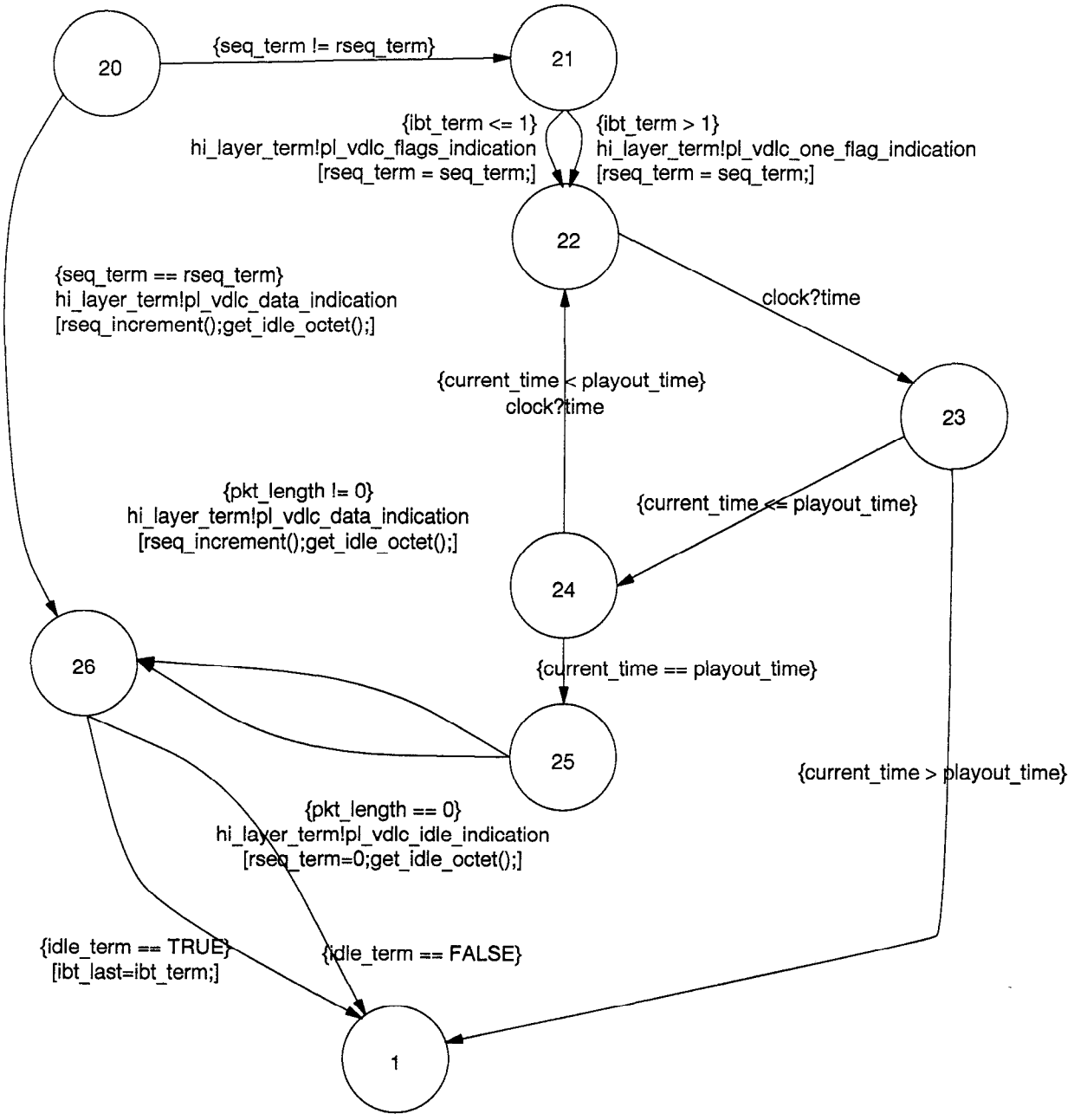


Figure D.17 (continued) – playout_vdlc
Part c

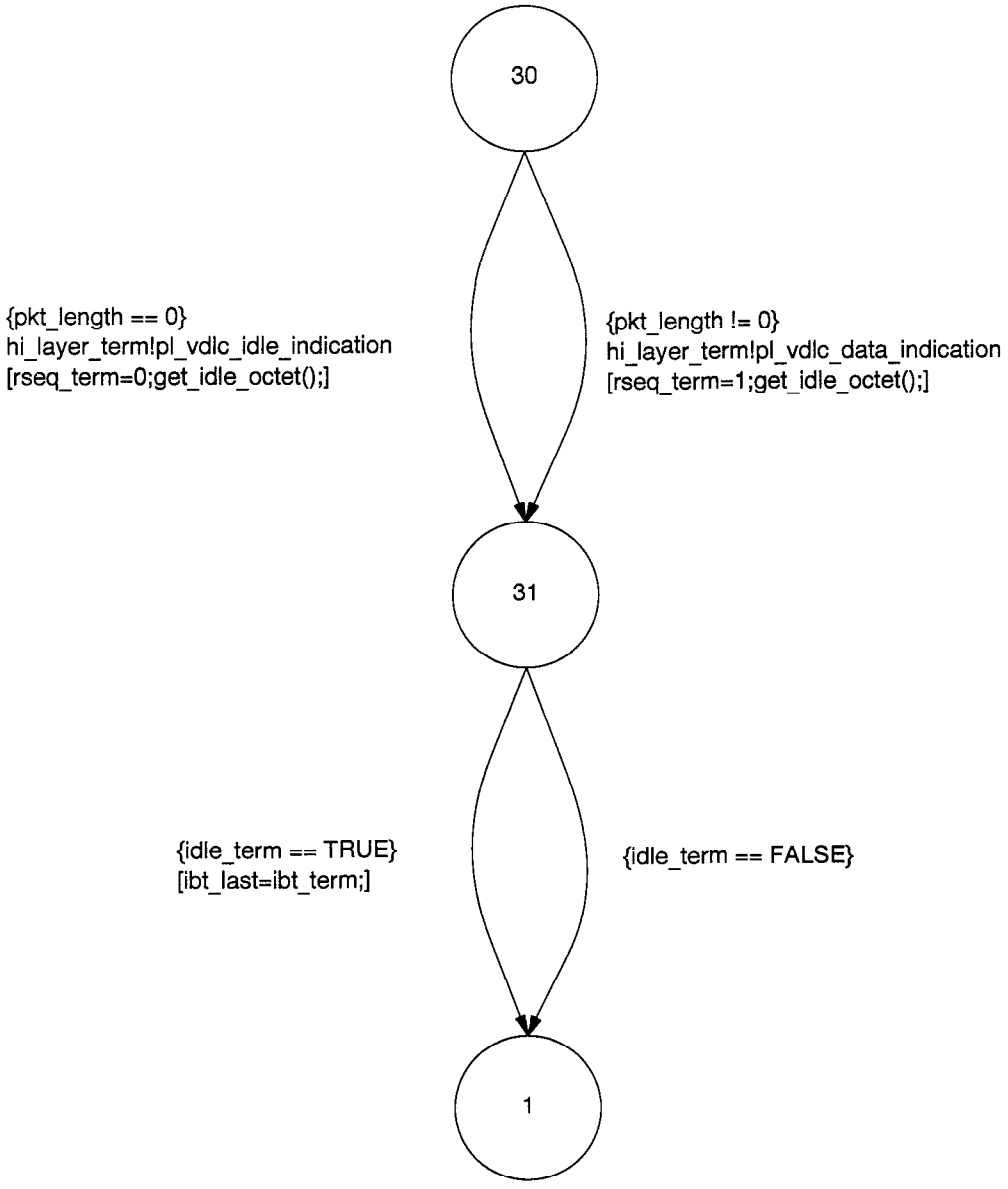


Figure D.17 (continued) – playout_vdlc
Part d

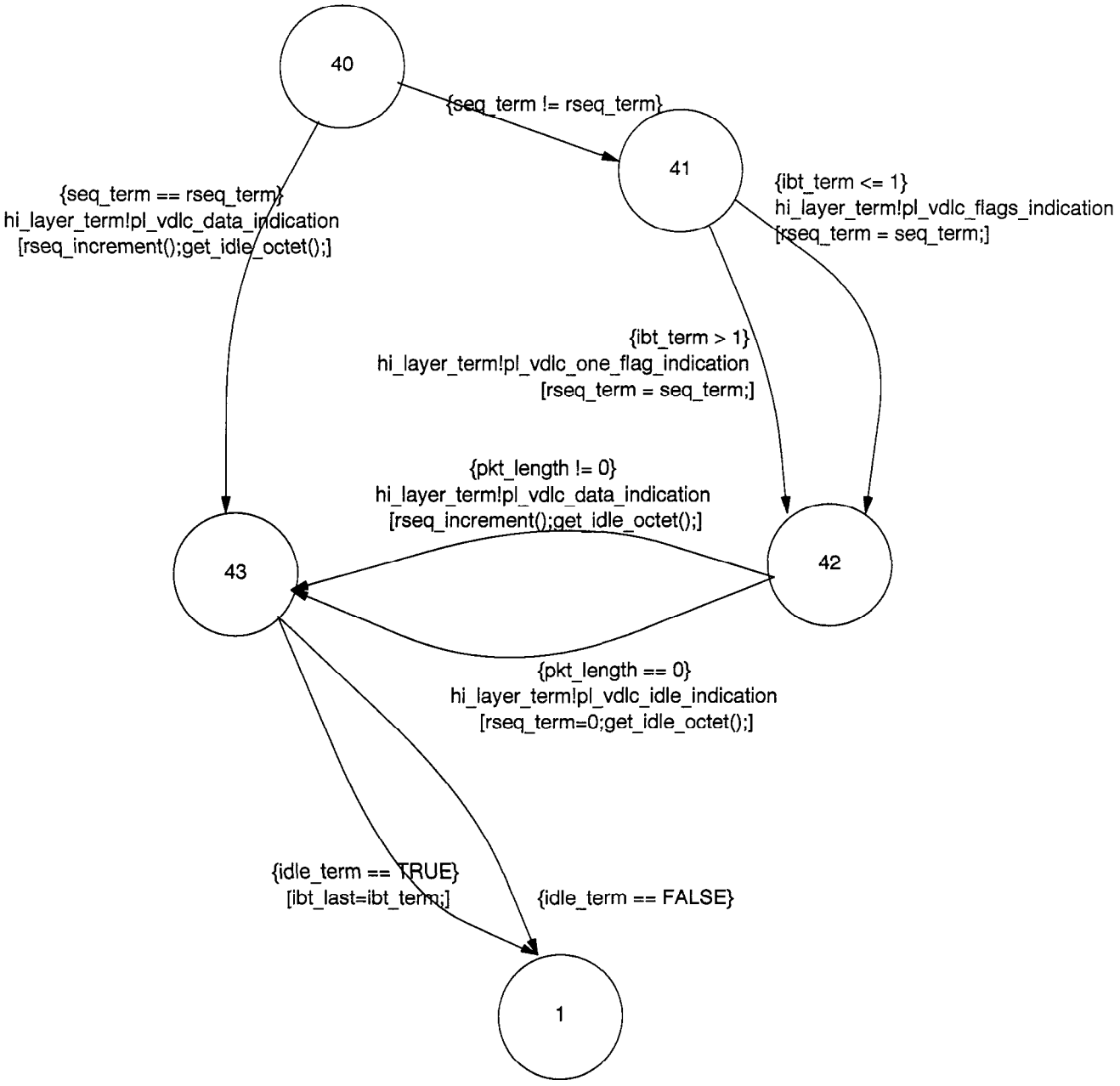


Figure D.17 (concluded) – playout_vdlc
Part e

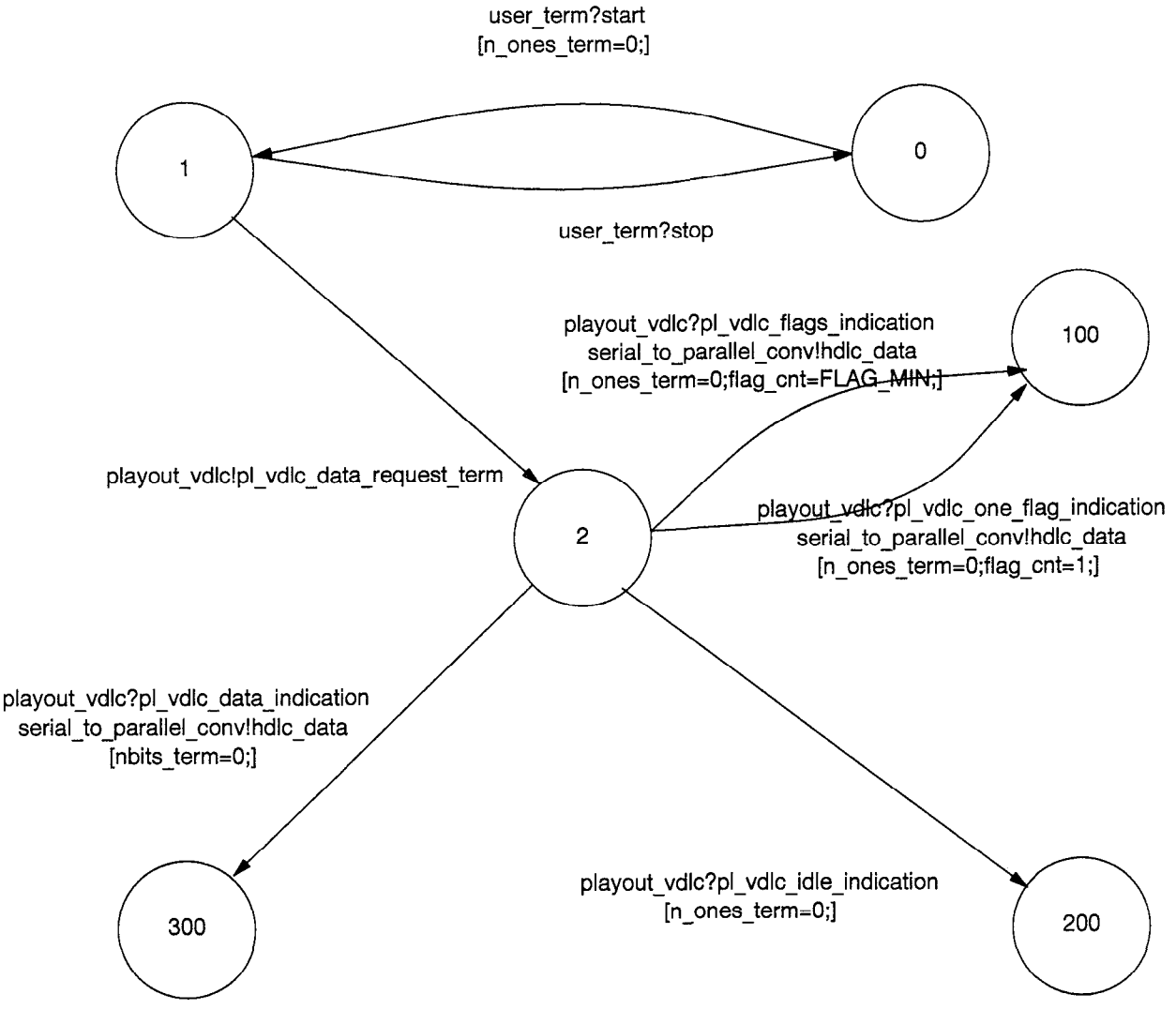


Figure D.18 – hi_layer_term
Part a

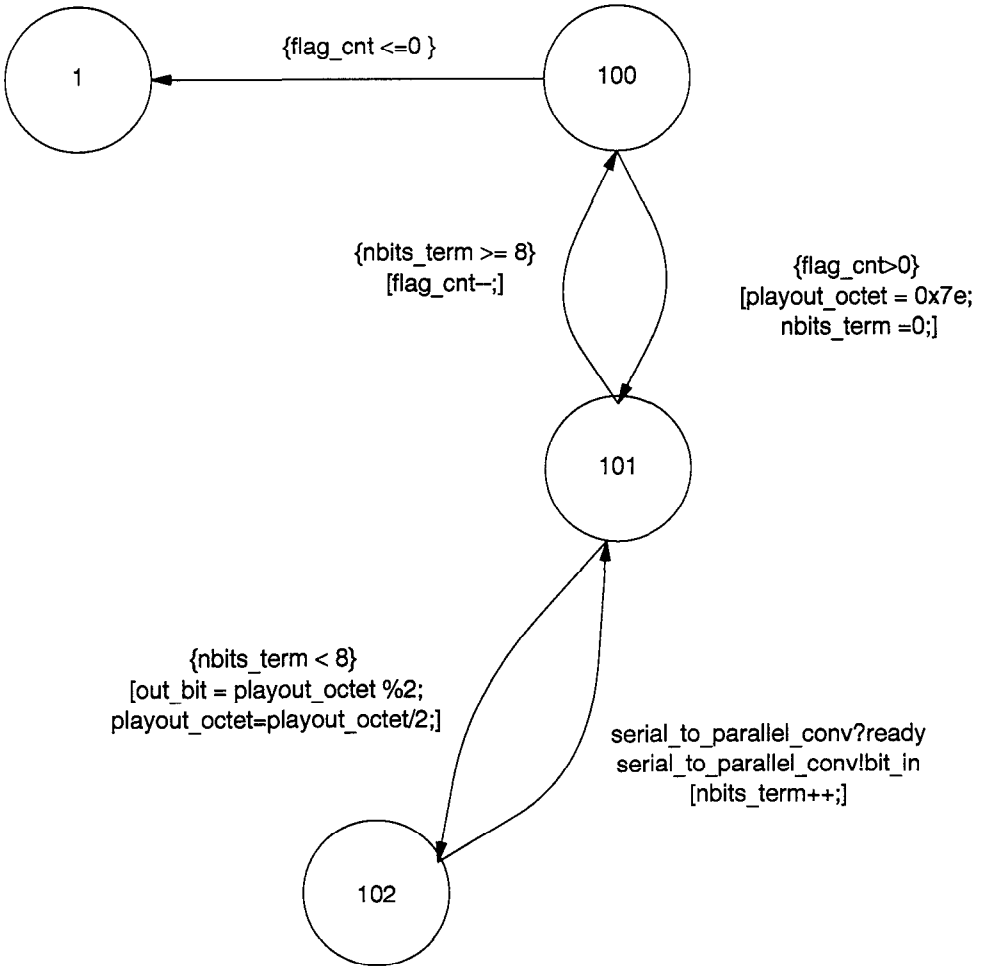


Figure D.18 (continued) – hi_layer_term
Part b

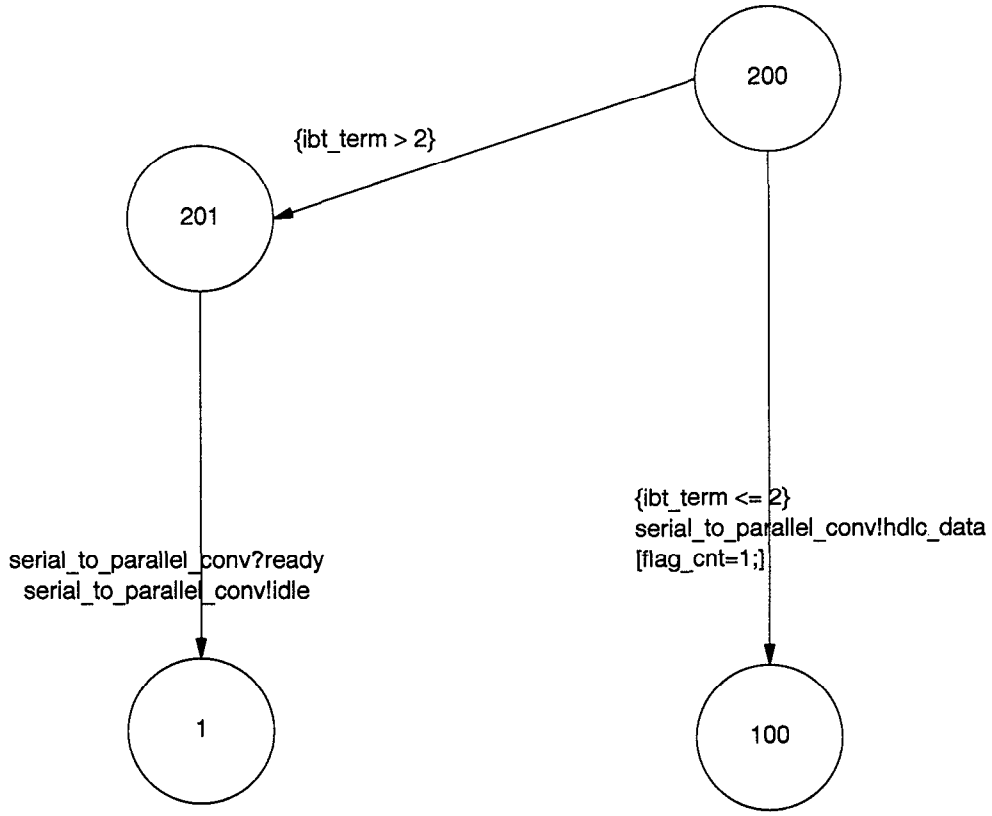


Figure D.18 (continued) – hi_layer_term
Part c

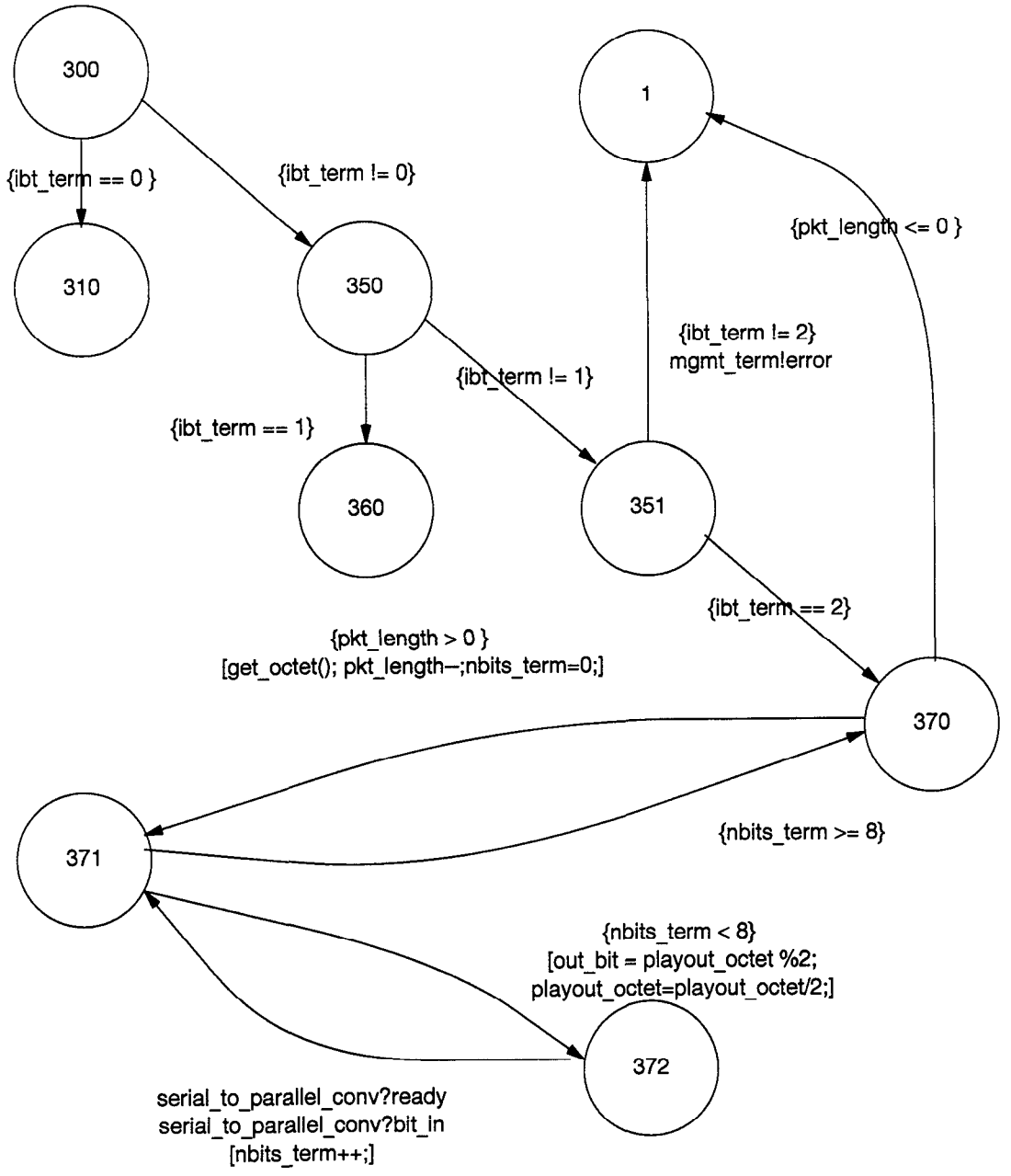


Figure D.18 (continued) – hi_layer_term
Part d

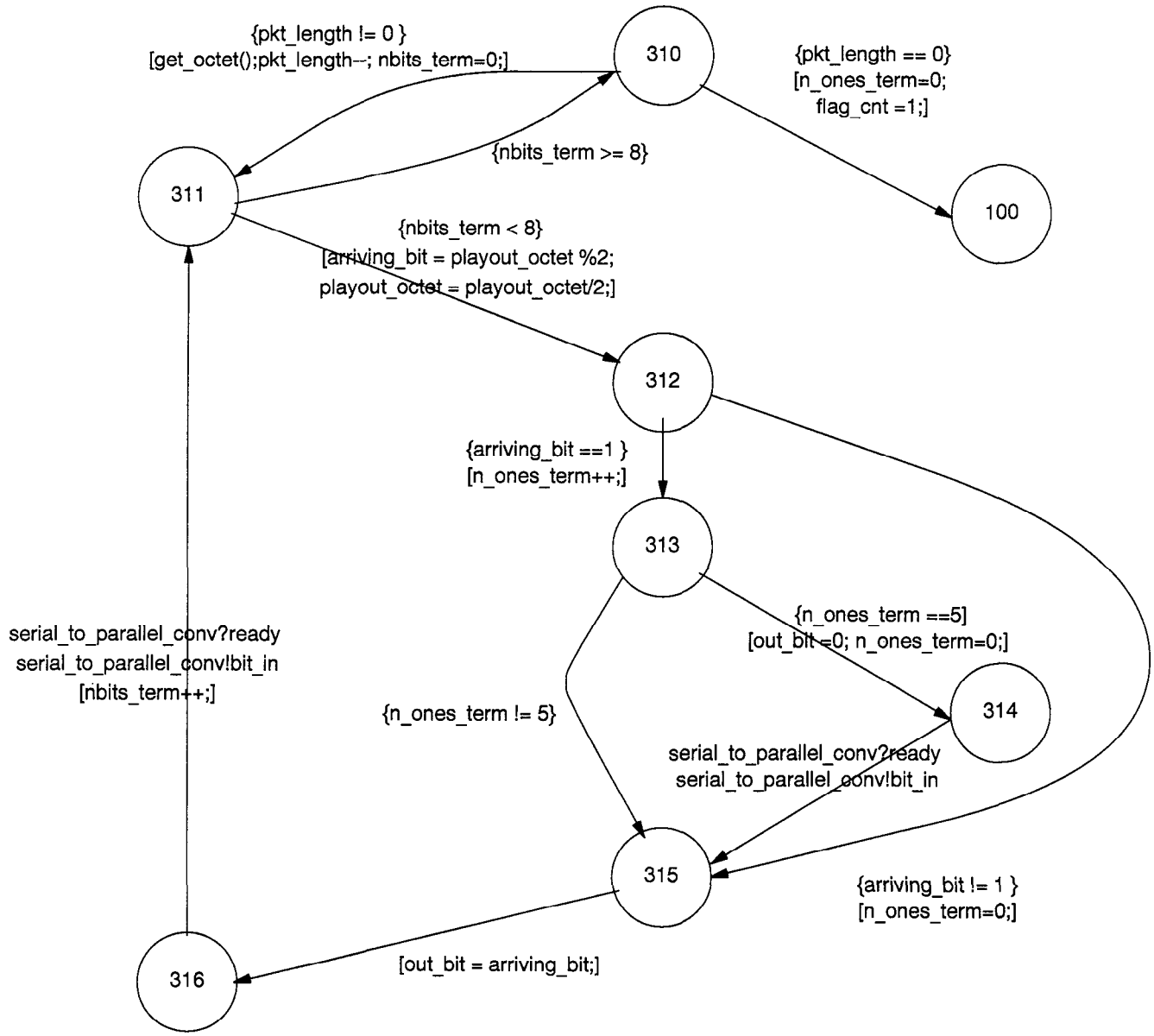


Figure D.18 (continued) – hi_layer_term
Part e

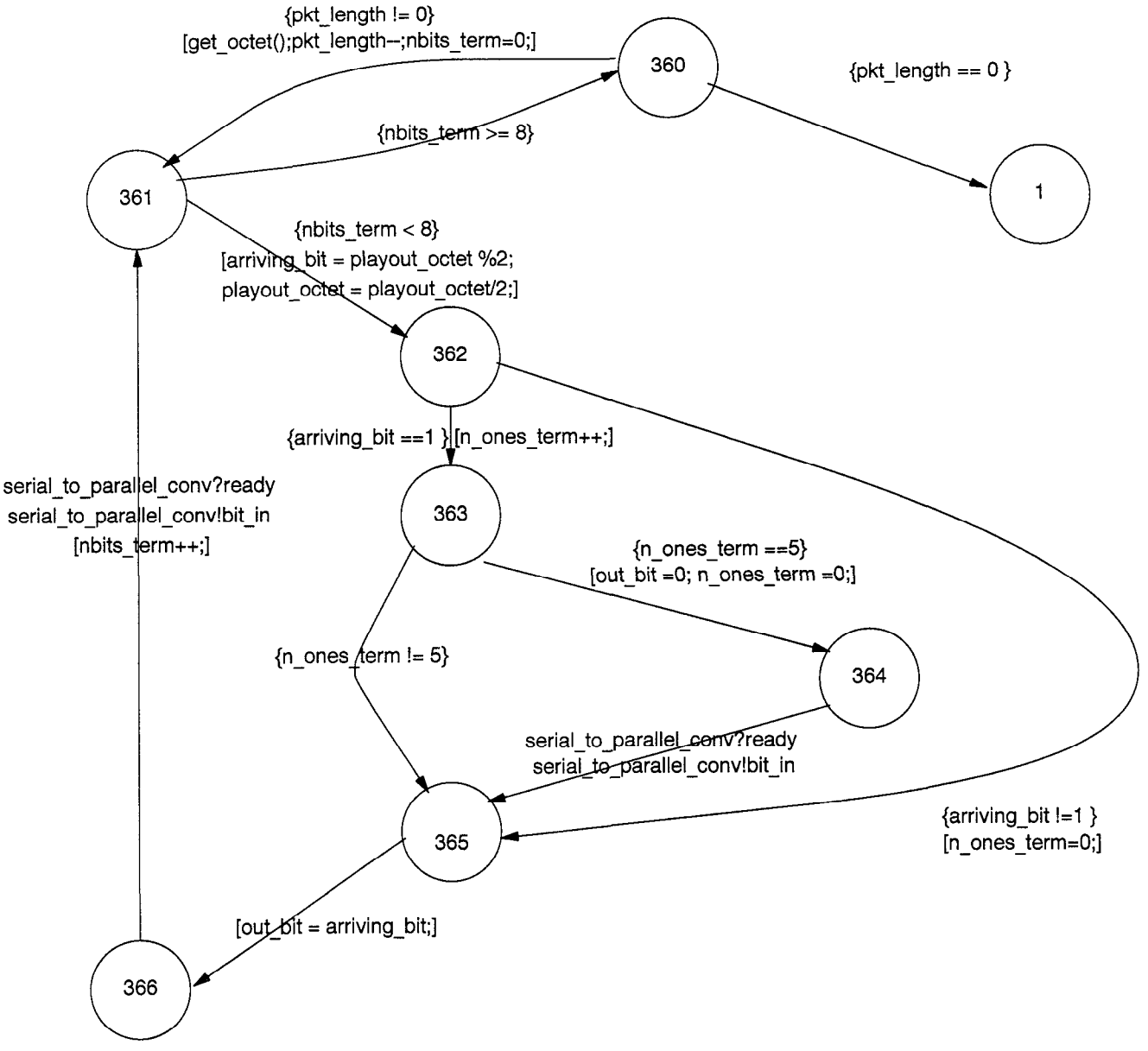


Figure D.18 (concluded) – hi_layer_term
Part f

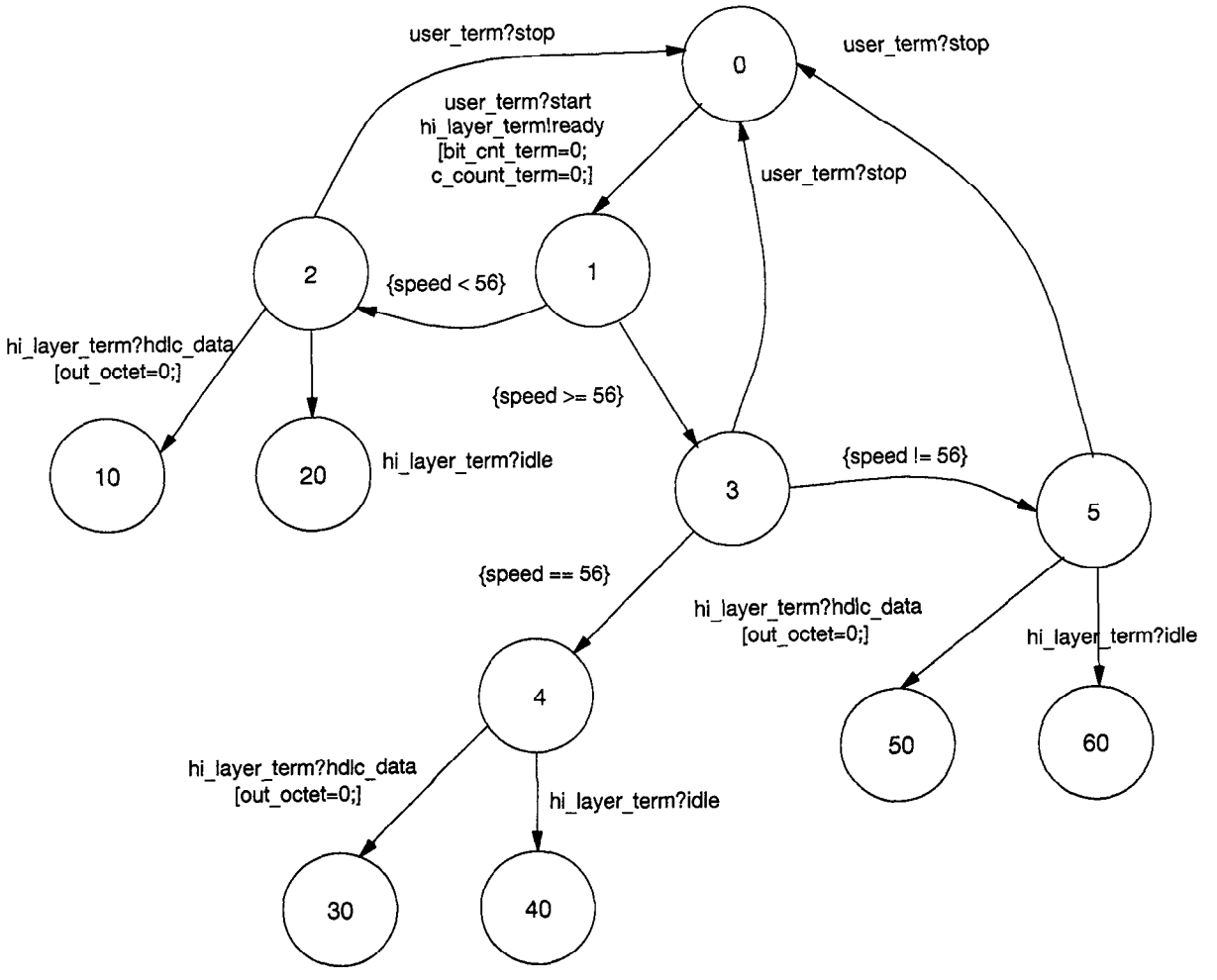


Figure D.19 – serial_to_parallel_conv
Part a

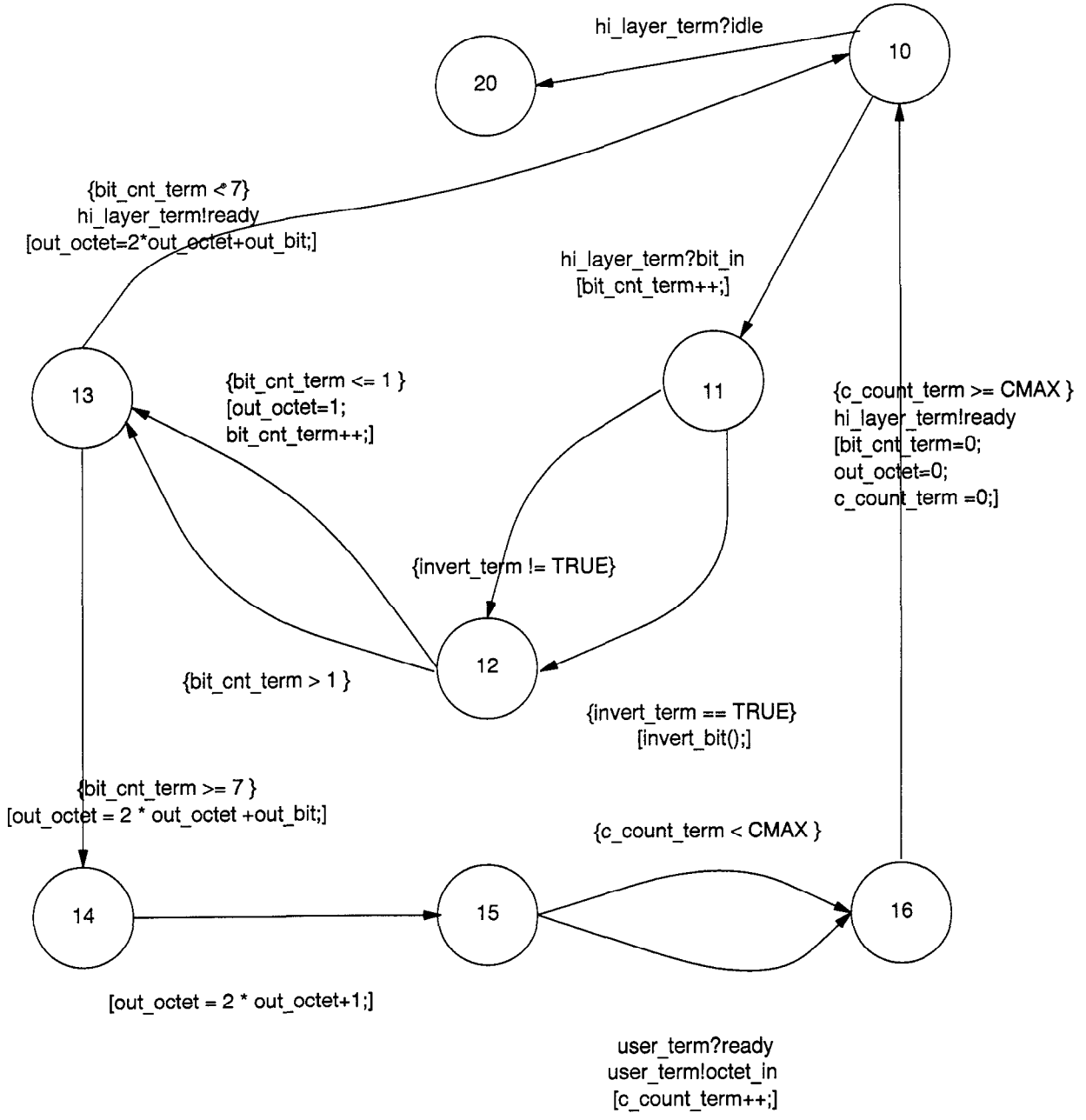


Figure D.19 (continued) – serial_to_parallel_conv
Part b

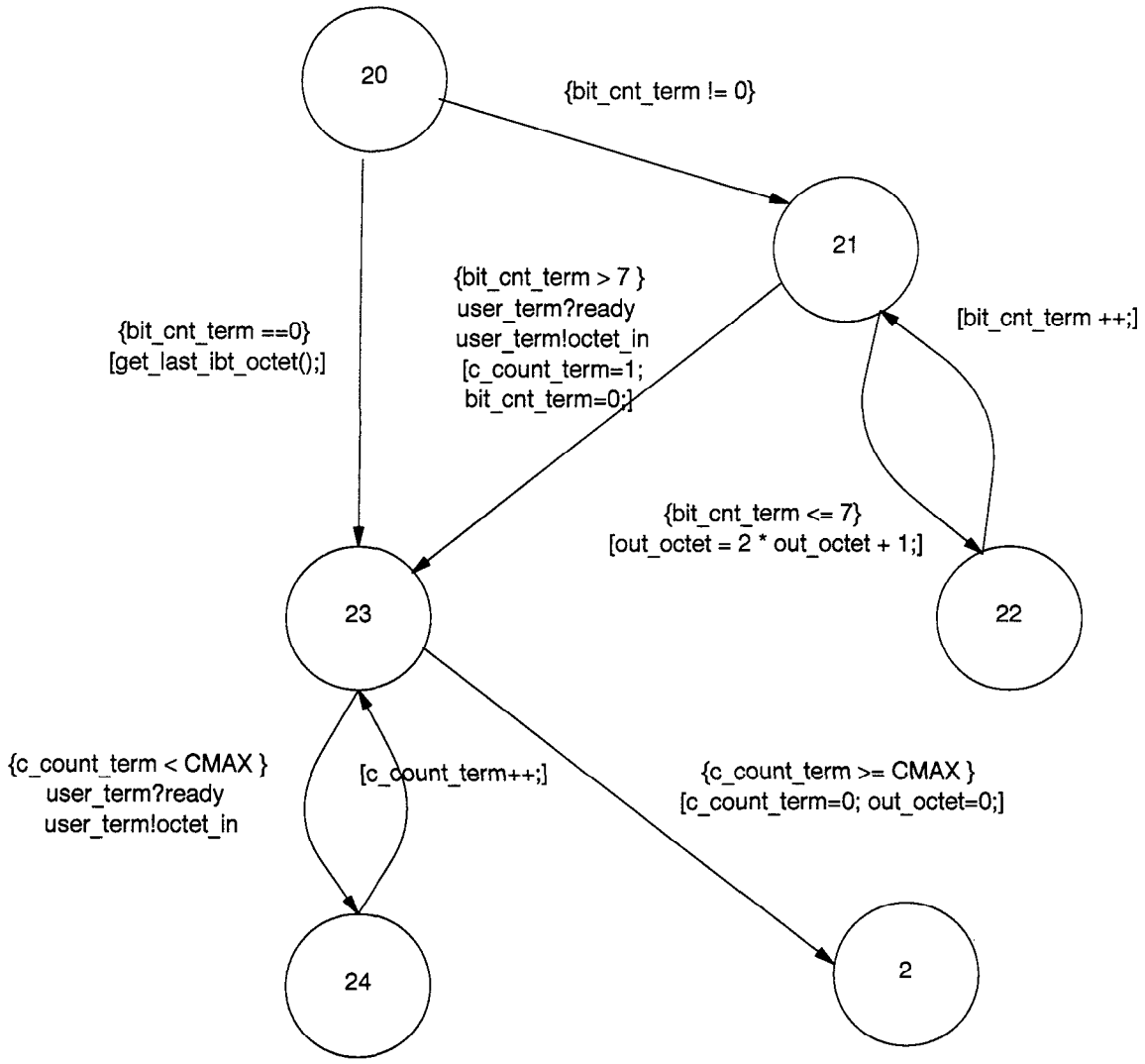


Figure D.19 (continued) – serial_to_parallel_conv
Part c

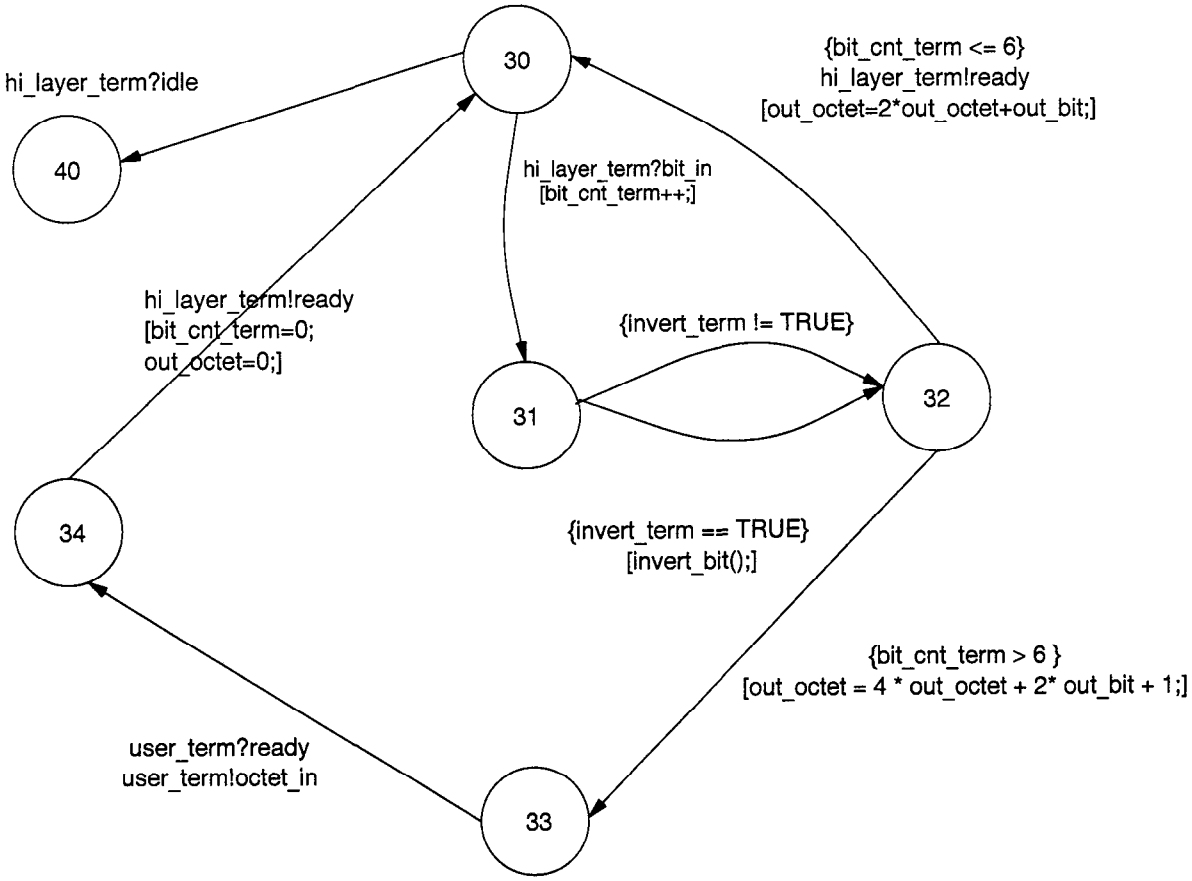


Figure D.19 (continued) – serial_to_parallel_conv
Part d

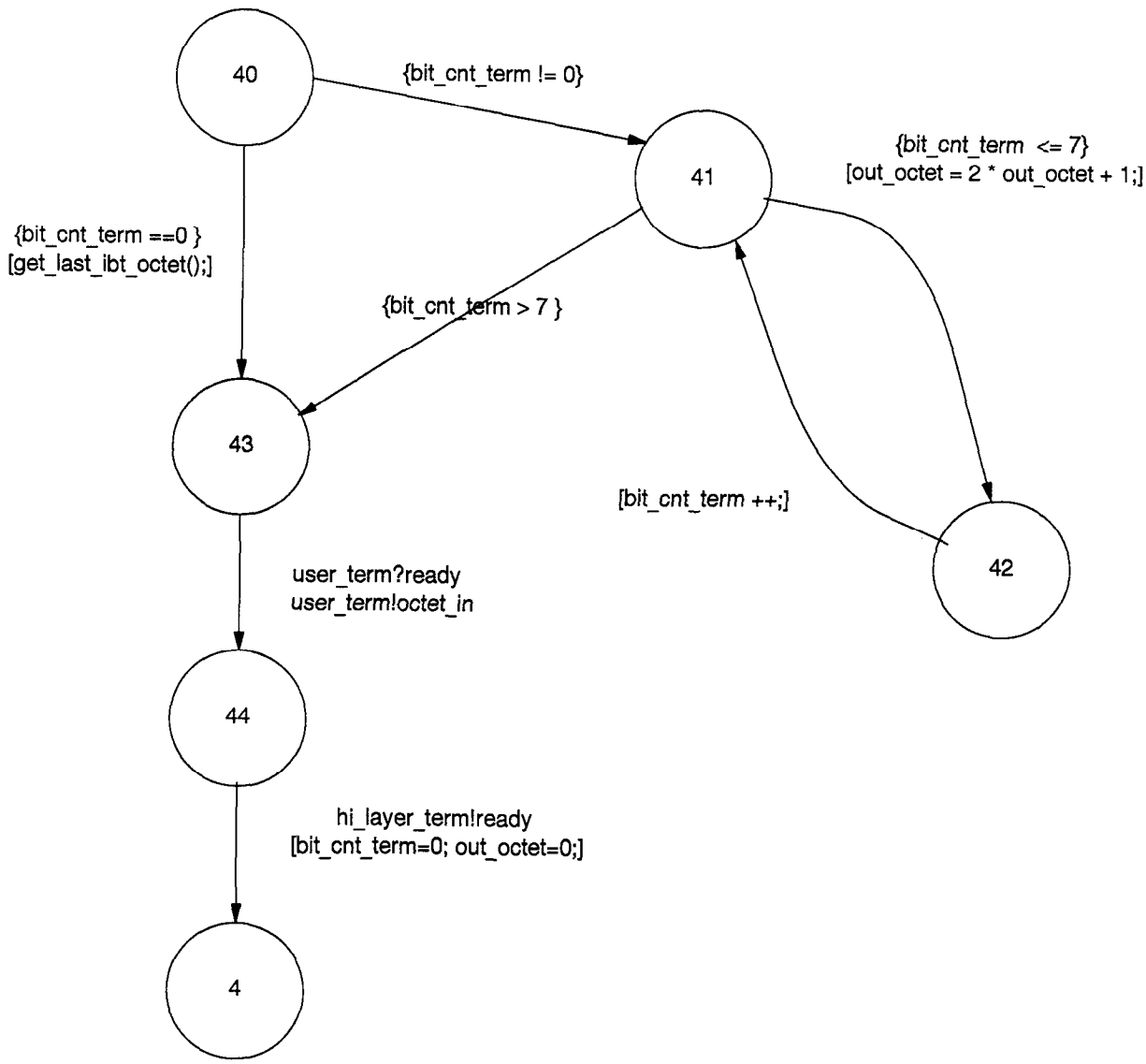


Figure D.19 (continued) – serial_to_parallel_conv
Part e

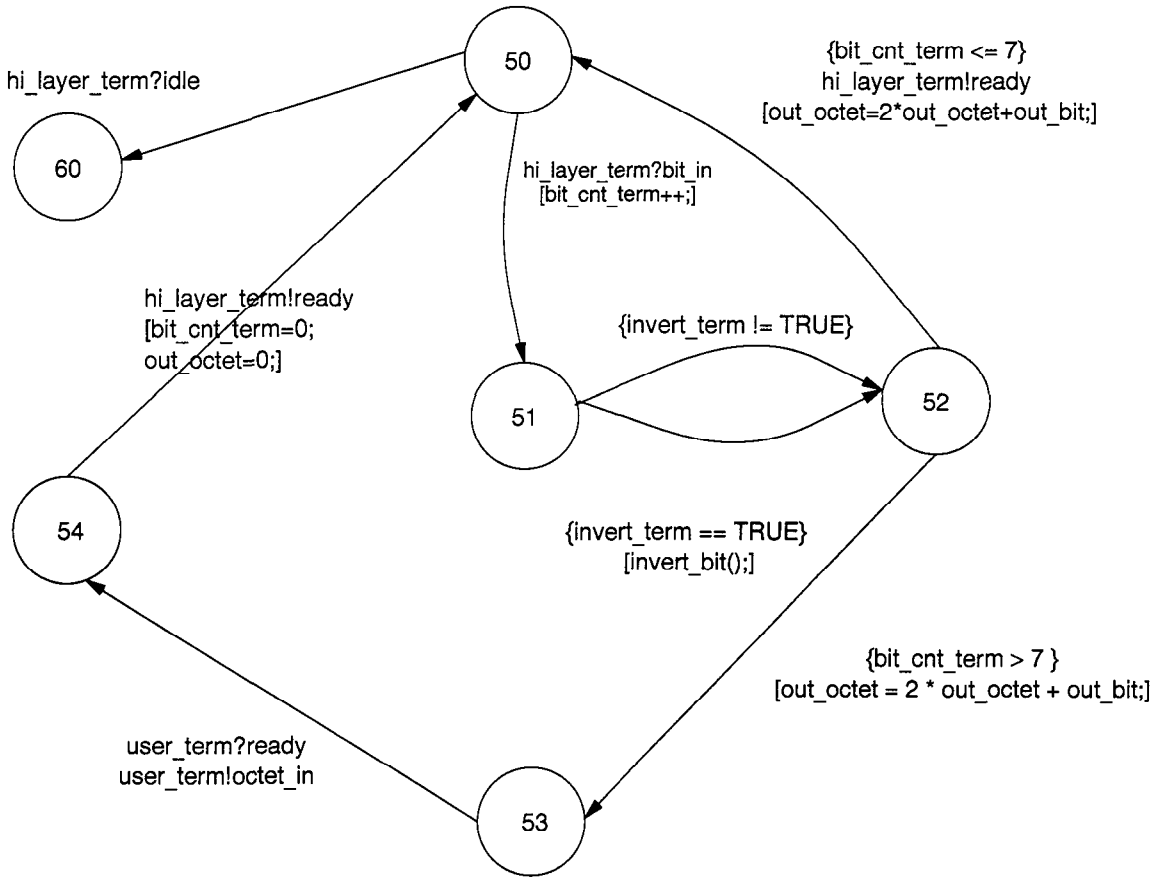


Figure D.19 (continued) – serial_to_parallel_conv
Part f

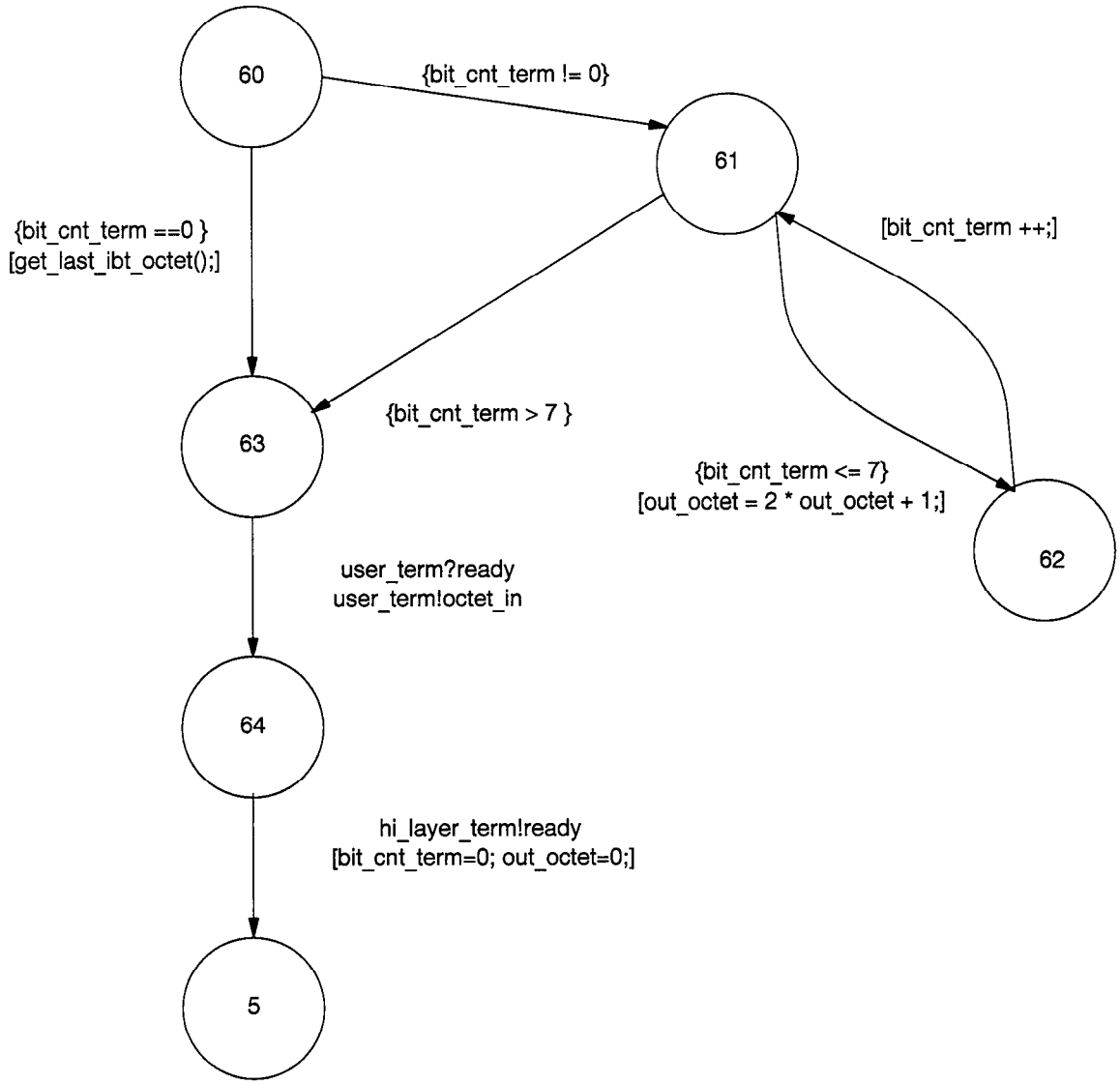


Figure D.19 (concluded) – serial_to_parallel_conv
Part g

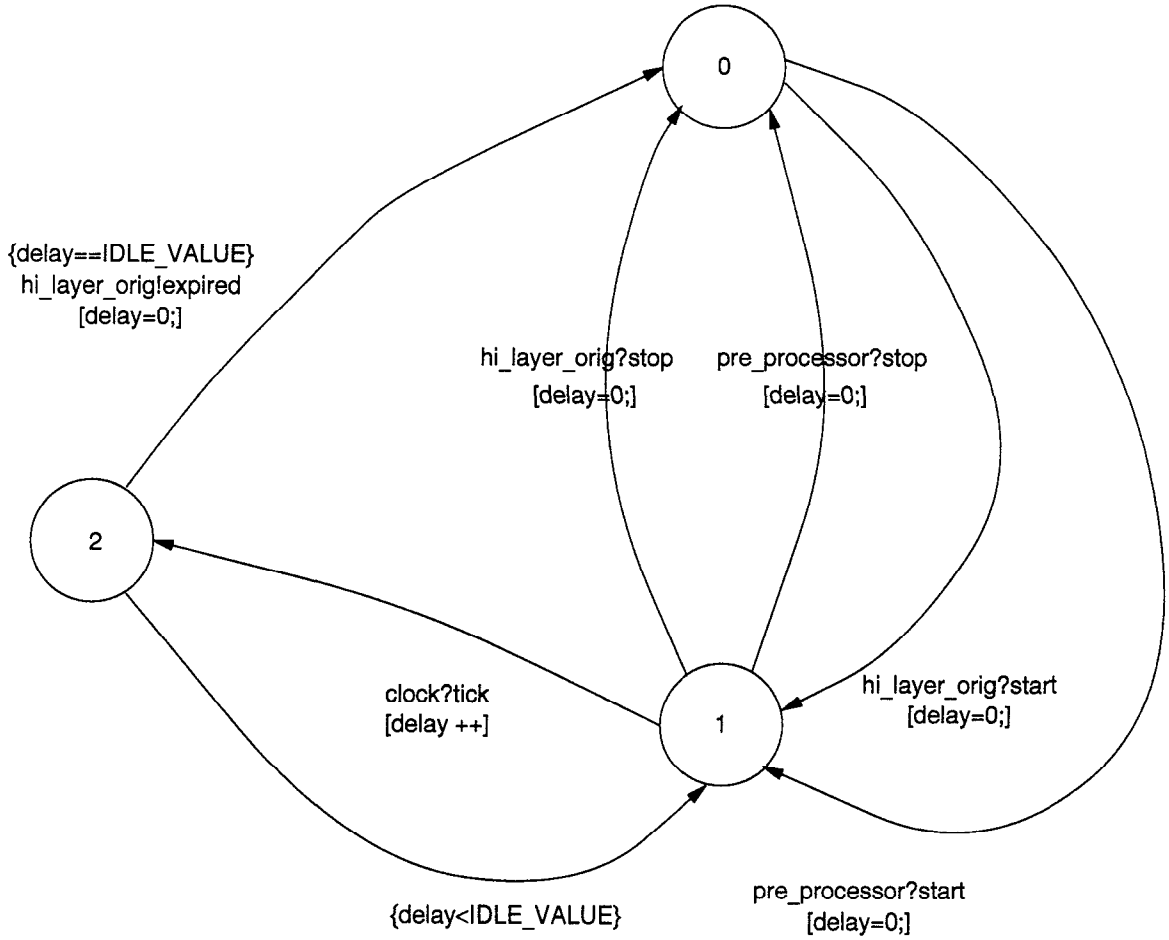


Figure D.20 – t_Idle

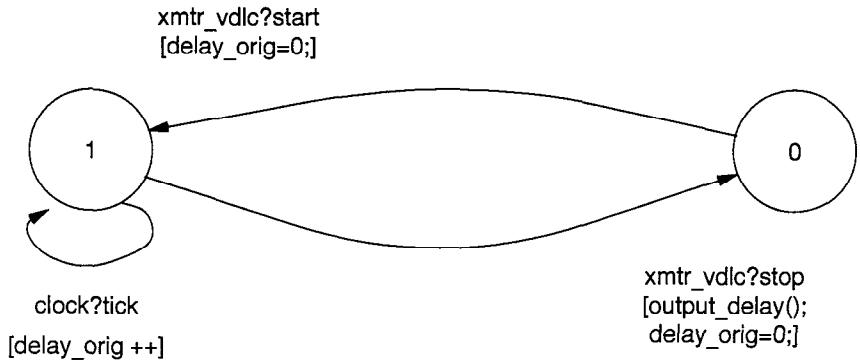


Figure D.21 – tvdelay_orig

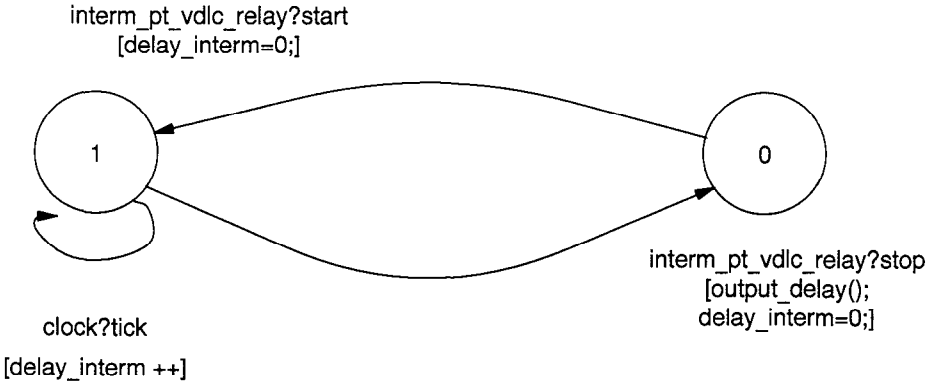


Figure D.22 – tvdelay_interm

Annex E (R2013)
(informative)

Formal Description of the Facsimile Demodulation and Compression Protocol (FADCOMP)

E.1 Processes in APSL Description

Table E.1 lists all the processes used in the Augmented Protocol Specification Language (APSL) specification of the FADCOMP Protocol. It includes a brief description of each:

Table E.1

Process Name	Description
call_layer	Call state layer entity
interm_pt_fax_relay	FADCOMP intermediate node
playout_fax	FADCOMP packet layer playout
fax_rcv_queue	FADCOMP packet layer queue at the terminating end
rcv_fax	FADCOMP packet layer receiver at the terminating end
xmtr_fax	FADCOMP packet layer transmitter at the originating end
bridge_timer	Timer for the BRIDGE state of the modulation layer of the terminating end
demux	Demultiplexer
ept_detector	Echo protection tone detector at the originating end
ept_timer	Echo protection tone timer at the terminating end
f_end_demod	Far end V.21 demodulator
idle_timer	Idle timer for the terminating end
n_end_demod	Near end V.21 demodulator
n_end_remod	Near end V.21 remodulator
tvdelay_interm	FADCOMP variable delay timer at the intermediate node
tvdelay_orig	FADCOMP variable delay timer at the originating end
wait_time	Wait timer at the originating end
clock	Clock
layer2_orig	Layer 2 at the originating end
layer2_interm	Layer 2 at the intermediate node
layer2_term	Layer 2 at the terminating end
t_accumulator	Timer of value 20 ms that controls the collection of data at the packet layer of the originating endpoint
timer_ra2	Timer associated with state r.a2
timer_rb2	Timer associated with state r b2
timer_rc2	Timer associated with state r.c2
timer_rd1	Timer associated with state r.d1
timer_re1	Timer associated with state r e1
timer_rg1	Timer associated with state r g1
timer_rg2	Timer associated with state r.g2
timer_rg3	Timer associated with state r.g3
timer_ta2	Timer associated with state t.a2
timer_tb2	Timer associated with state t.b2
timer_tc2	Timer associated with state t.c2
timer_td1	Timer associated with state t.d1
timer_te1	Timer associated with state t.e1
timer_tg1	Timer associated with state t.g1
timer_tg2	Timer associated with state t.g2
timer_tg3	Timer associated with state t.g3
user_orig	User process at the originating end
user_term	User process at the terminating end

The following are treated as EXTERNAL processes:

clock, demux, ept_detector, f_end_demod, idle_timer, n_end_demod,
n_end_remod,layer2_interm, layer2_orig, layer2_term,
mux, t_accumulator, timer_ra2, timer_rb2, timer_rc2,
timer_rd1, timer_re1, timer_rg1, timer_rg2,
timer_rg3, timer_ta2, timer_tb2, timer_tc2, timer_td1, timer_te1,
timer_tg1, timer_tg2, timer_tg3, training_detector, train_gen_orig, train_gen_term,
user_orig, user_term.

Table E.2 lists messages that apply for any timer:

Table E.2

Message name	Description
stop	Stop the timer and reset all variables to zero
start	Start the timer.
timer!expired	The timer has expired

E.2 Functions in the APSL Description of FADCOMP

Table E.3 alphabetically lists the various functions used in the APSL specification of FADCOMP

Table E.3

Function	Description
allocate_modem()	Attempts to allocate modem resources. For the originating endpoint, it sets the variable 'allocate_orig_ok' to TRUE if allocation is successful; otherwise, it sets it to FALSE. Similarly, it sets the variable 'allocate_term_ok' at the terminating endpoint.
begin_training()	Generates PCM samples that correspond to a training sequence for modem ML_TERM_TYPE at speed ML_TERM_SPEED.
check_compat()	Checks whether the modulation speed and type in the T.30 V.21 message just received are compatible with allocated modems. Sets 'fax_ok' to TRUE if the facsimile traffic can be demodulated/remodulated on both ends; otherwise, it sets the variable fax_ok to FALSE.
check_ept()	Checks whether an EPT tone is present and sets the variable 'etp_type_orig' to 1700 or 1800, depending on the frequency of the tone.
deallocate_resources()	Frees allocated mod/demod resources
drop_pkt()	Drops the current packet.

(continued)

Table E.3 (concluded)

Function	Description
generate_mod_bits()	Generates a modulated signal corresponding to the encoding of the all zeros symbol at the terminating endpoint for modem type ML_TERM_TYPE and speed ML_TERM_SPEED while awaiting data to arrive.
generate_ept_tone()	Generates echo protection tone at terminating endpoint as per CCITT Rec V.2.
get_bits()	Retrieves data bits from playout buffer.
mod_symbol()	Modulates a symbol arriving from the packet layer. For V.29, refer to Figure 11 for the arrangements of the symbols.
pop_pkt()	Takes a packet off the playout queue for playout and sets the terminating end variables 'bilo_term' and 'seq_term' accordingly.
push_pkt()	Queues a received packet and sets the corresponding playout_time variable.
rseq_increment()	Increments the receive sequence number (RSEQ) (1 to 15 with rollover to 1).
seq_increment()	Increments the sequence number (SEQ) (1 to 15 with rollover to 1).
set_modem_param()	Sets the system parameters (ML_ORIG_TYPE, ML_ORIG_SPEED) at the originating endpoint and (ML_TERM_TYPE, ML_TERM_SPEED) at the terminating endpoint, respectively, as extracted from the T.30 DCS or NSS commands.
set_playout_time()	Sets the variable 'playout_time' to BUILDOUT - time stamp + current_time.
set_pkt_length()	Calculates the packet length in octets and stores it in the variable 'pkt_length.'
stop_ept_tone()	Stops the echo protection tone generation at the terminating endpoint.
stop_training()	Stops the generation of PCM samples that correspond to a training sequence for modem ML_TERM_TYPE at speed ML_TERM_SPEED.
store_cmd_type()	Stores the type of the V.21 command from the near-end demodulator in the variable 'near_cmd_type' and from the far-end demodulator in 'far_cmd_type.'
ts_update()	Updates the time stamp by the value of the delay measured by the 'tvdelay_orig' or 'tvdelay_interm' timers.

E.3 Primitives

Table E.4 describes the various primitives used in the APSL description:

Table E.4

Primitive	APSL representation
DL_L1_READY_INDICATION	dl_l1_ready_indication
DL_PVP_DATA_REQUEST	dl_pvp_data_request
DL_PVP_H_DATA_REQUEST	dl_pvp_h_data_request
DL_UNIT_DATA_INDICATION	dl_unit_data_indication
DL_UNIT_H_DATA_INDICATION	dl_unit_h_data_indication
DL_UNIT_DATA_REQUEST	dl_unit_data_request
DL_UNIT_H_DATA_REQUEST	dl_unit_h_data_request
ML_FAX_ABORT_CONFIRMATION	ml_fax_abort_confirmation
ML_FAX_ABORT_INDICATION	ml_fax_abort_indication
ML_FAX_START_REQUEST	ml_fax_start_request
ML_FAX_START_RESPONSE	ml_fax_start_response
ML_FAX_STOP_CONFIRMATION	ml_fax_stop_confirmation
ML_FAX_STOP_REQUEST	ml_fax_stop_request
ML_FAX_STOP_RESPONSE	ml_fax_stop_response
ML_FAX_STOP_INDICATION	ml_fax_stop_indication
PL_FAX_DATA_INDICATION	pl_fax_data_indication
PL_FAX_DATA_START_REQUEST	pl_fax_data_start_request
PL_FAX_DATA_STOP_INDICATION	pl_fax_data_stop_indication
PL_FAX_DATA_STOP_REQUEST	pl_fax_data_stop_request
PL_FAX_SPURT_HEADER(ACTION)	pl_fax_spurt_header_abort
	pl_fax_spurt_header_ept_start
	pl_fax_spurt_header_ept_stop
	pl_fax_spurt_header_train

Note: The capability indication primitives are not used in this description

E.4 Counters in APSL Description

Table E.5 lists all the counters used in the APSL specification of the FADCOMP Protocol. It includes a brief description of each:

Table E.5

Counter name	What it counts
no_bits_orig	Bits at the originating endpoint
part_page_req_cnt	PPR commands received

E.5 Internal Variables

Table E.6 lists internal variables used in the APSL description:

Table E.6

Variable	Description
action_orig	Action field at the originating end
action_term	Action field at the terminating end
bilo_orig	Bit in the last octet at the originating end
current_time	Current time as determined by a continuous clock
delay_interm	Delay in the intermediate node
delay_orig	Delay in the originating end
ept_freq_term	Frequency of the echo protection tone at the terminating end
ept_type_orig	Type field for the echo protection tone
far_eor_type	Type of T.30 End of Retransmission message from the far-end demodulator
fax_ok	Variable set by check_compat() to indicate that the modulation speed and type from the incoming V.21 message are compatible with the allocated modem
have_room	1 when a packet can be stored, 0 otherwise
mbit_orig	Value of the M-bit field at the originating end
mbit_term	Value of the M-bit field of the received packet at the terminating end
near_eor_type	Type of T.30 End of Retransmission message from the near-end demodulator
pd	Protocol Discriminator
pkt_length	Length in octets of the facsimile information field in the received frame
playout_time	Time scheduled for playout of a received packet as calculated by set_playout_time()
q_empty	1 when playout queue is empty, 0 otherwise
rseq_term	RSEQ variable at the terminating end
seq_orig	SEQ variable at the originating end
seq_term	SEQ variable in the received packet
service_class	Value of the Service Class field in the packet header
ts	Time Stamp (updated by the ts_update function)
type_orig	Type field at the originating end
type_term	Type field at the terminating end
uih	1 when the frames used are UIH, 0 when the frames used are UI

E.6 Coordination Messages

Table E.7 lists messages used in the APSL description to coordinate between various originating end processes:

Table E.7

411.if 712<36 .nr 50 36
406.if 712<36 .nr 50 36

Origin	Message	Destination
bridge_timer	expired	mod_layer_term
call_layer	fadcomp pvp start start	mux, demux " timer_rxn timer_tyn
ept_detector	ept_tone	mod_layer_orig
ept_timer_orig	expired	mod_layer_orig
ept_timer_term	expired	mod_layer_term
fax_rcv_queue	queue_empty pkt_in	playout_fax "
f_end_demod	T.30 messages V.21	call_state_layer "
idle_timer	expired	mod_layer_term
mod_layer_orig	start stop pl_fax_spurt_header_abort pl_fax_data_start_request pl_fax_data_stop_request pl_fax_spurt_header_ept_start pl_fax_spurt_header_ept_stop pl_fax_spurt_header_train start stopt	ept_timer_orig " xmtr_fax " " " " " wait_time wait_time
mod_layer_term	start stop ml_fax_abort_indication ml_fax_start_response ml_fax_stop_indication ml_fax_stop_response start stop start stop	bridge_timer " call_layer_term " " " ept_timer_term " idle_timer "
n_end_demod	T.30 messages V.21 demod_error end_of_data	call_state_layer " mod_layer_orig "

(continued)

Table E.7 (concluded)

Origin	Message	Destination
interm_pt_vdlc_relay	dl_pvp_data_request dl_pvp_h_data_request start stop	layer2_interm " tv_delay_interm "
playout_fax	pl_fax_data_start_indication pl_fax_data_stop_indication pl_fax_idle_indication pl_fax_spurt_header_indication read_pkt scan	mod_layer_term " " " fax_rcv_queue "
rcv_fax	write_pkt	fax_rcv_queue
train_detector	train	mod_layer_orig
train_gen_orig	end_of_sequence	mod_layer_orig
train_gen_term	end_of_sequence	mod_layer_term
user_orig	input_in fax_data_request no_fax_data	call_layer_orig start stop mod_layer_origin
user_term	fax_data_indication start stop start stop	mod_layer_term mod_layer_term " playout_fax "
xmtr_fax	dl_unit_data_request dl_unit_h_data_request start stop	layer2_orig " tv_delay_orig "
wait_time	expired	mod_layer_orig

* Timer_rxn is a shorthand notation for timer_ra2, timer_rb2, timer_rc2, timer_rd1, timer_re1, timer_rg1, timer_rg2, timer_rg3 Similarly, timer_tyn is a shorthand notation for timer_ta2, timer_tb2, timer_tc2, timer_td1, timer_te1, timer_tg1, timer_tg2, timer_tg3

** The messages are the final T 30 messages (unless indicated otherwise) CTC, CFR, CRP, CTR, DCN, DCS, DIS, DTC, EOM, EOP, EOR EOM, EOR EOP, EOR MPS, EOR Null, EOR PRI-EOM, EOR PRI-EOP, EOR PRI-MPS, ERR, FTT, MCF, MPS, NSC, NSS (non-final, represented as NSS_FO), NSS, NSS (final or non-final, represented as NSS_FX), PIN, PIP, PPR, PPS-EOM, PPS-EOP, PPS-MPS, PPS-Null, PPS PRI-EOP, PPS PRI-EOM, PPS PRI-MPS, PRI-EOM, PRI-EOP, PRI-MPS, RTN, and RTP

```
/*
        various definitions
                        */

#define ABORT 2
#define BRIDGE_TIME_VALUE 500
#define BUILDOUT      100
#define EOM    1
#define EOP    2
#define MPS    3
#define PPSEOM  4
#define PPSEOP  5
#define PPSMPS  6
#define PPSNull 7
#define PPS_PRIEOM  8
#define PPS_PRIEOP  9
#define PPS_PRIMPS  10
#define PRIEOM      11
#define PRIEOP      12
#define PRIMPS      13
#define Null     0
#define EPT_1700       3
#define EPT_1800       4
#define EPT_STOP       5
#define EPT_TIME_VALUE              500
#define FALSE          0
#define PVP            0
#define STOP_EPT       5
#define TRUE           1
#define TRAINING       1
#define WAIT_FOR_SIGNAL_VALUE           100

/*
                allocate_orig_ok        = 1 modem allocation is possible at the originating endpoint,
                                = 0 otherwise;
                allocate_term_ok        = 1 modem allocation is possible at the terminating endpoint,
                                = 0 otherwise;
                action_orig     = action field at the originating end
                action_term     = action field at the terminating end
                bilo_orig       = bit in the last octet at the originating end
                current_time    = current time as determined by a continuous clock
                delay_interm    = delay in the intermediate node,
                delay_orig      = delay in the originating end,
                ept_freq_term   = frequency of the echo protection tone at the terminating end
                ept_type_orig   = type field for the echo protection tone
                fax_ok          = 1 when the modulation speed and type in V.21 message
                                are compatible with the allocated modem,
                                = 0 otherwise;
                far_cmd_type    = type of T.30 V.21 command from the far end demodulator
                far_eor_type    = type of T.30 End of Retransmission message from the far end demodulator
                have_room       =  1 when a packet can be stored
                                = 0 otherwise;
                mbit_orig       = value of the M Bit field at the originating end
                mbit_term       = value of the M Bit field of the received packet at the terminating end
                near_cmd_type = type of T.30 V.21 command from the near end demodulator
```

```
                near_eor_type   = type of T.30 End of Retransmission message from the near end demodulator
                no_bits_orig    = number of bits at the originating endpoint
                part_page_req_cnt = counter for the number of PPR received
                playout_time    = time scheduled for playout of a received packet,
                pd              Protocol Discriminator
                q_empty                 = 1 when playout queue is empty,
                                = 0 otherwise;
                pkt_length      = length in octets of the facsimile information field in the received frame,
                                = 0 otherwise;
                rseq_term       RSEQ variable at the terminating end
                seq_orig        SEQ variable at the originating end
                seq_term        SEQ variable in the received packet
                service_class   value of the Service Class field in the packet header
                ts              Time Stamp (updated by the ts_update function)
                type_orig       = type field at the originating endpoint,
                type_term       = type field at the terminating endpoint,
                uih             = 1 when the frames used are UIH
                                = 0 when the frames used are UI
                                                                */

/*
                External Processes
*/
EXTERNAL clock, demux, ept_detector, f_end_demod, idle_timer, n_end_demod, n_end_remod,
        layer2_interm, layer2_orig, layer2_term,
        mux, t_accumulator, timer_ra2, timer_rb2, timer_rc2,
        timer_rd1, timer_re1, timer_rg1, timer_rg2,
        timer_rg3, timer_ta2, timer_tb2, timer_tc2, timer_td1, timer_te1,
        timer_tg1, timer_tg2, timer_tg3,  training_detector, train_gen_orig, train_gen_term,
        user_orig,  user_term;
/*
                Call State Layer
*/
PROCESS         call_state;
CONTEXT         allocate_orig_ok, allocate_term_ok, fax_ok, far_cmd_type, far_eor_type,
        near_cmd_type, near_eor_type, part_page_req_cnt;
STATES          0-3,10-14,20-22,30-33,35-50,60-64,70-72,80-83,90-93, 100,
    111-114,120-122,130-133,135-150,160-164,170-172,180-183,190-193;
REND    demux!fadcomp, demux!pvp, demux!off,
        f_end_demod?CTC, f_end_demod?CFR, f_end_demod?CRP, f_end_demod?CTR,
        f_end_demod?DCN, f_end_demod?DCS, f_end_demod?DIS, f_end_demod?DTC,
        f_end_demod?EOR_EOM, f_end_demod?EOR_EOP,
        f_end_demod?EOR_MPS, f_end_demod?EOR_Null, f_end_demod?EOR_PRIEOM,
        f_end_demod?EOR_PRIEOP, f_end_demod?EOR_PRIMPS, f_end_demod?ERR,
        f_end_demod?FTT, f_end_demod?MCF,  f_end_demod?NSC, f_end_demod?NSS_F0 ,
        f_end_demod?NSS , f_end_demod?NSS_FX, f_end_demod?PIN, f_end_demod?PIP,
        f_end_demod?PPR,  f_end_demod?PPS_PRIEOP, f_end_demod?PPS_PRIEOM, f_end_demod?
                PPS_PRIMPS,
        f_end_demod?PRIEOM, f_end_demod?PRIEOP, f_end_demod?PRIMPS, f_end_demod?RTN,
        f_end_demod?RTP, f_end_demod?v21,
        mod_layer_orig!ml_fax_start_request, mod_layer_orig!ml_fax_stop_request,
        mod_layer_orig?ml_fax_abort_confirmation, mod_layer_orig?ml_fax_stop_confirmation,
        mod_layer_term?ml_fax_abort_indication, mod_layer_term?ml_fax_stop_indication,
        mod_layer_term!ml_fax_start_response, mod_layer_term!ml_fax_stop_response,
        mux!fadcomp, mux!pvp, n_end_demod?CFR, n_end_demod?CRP, n_end_demod?CTC,
```

```
        n_end_demod?CTR, n_end_demod?DCN, n_end_demod?DCS, n_end_demod?DIS,
        n_end_demod?DTC, n_end_demod?EOR_EOM ,
        n_end_demod?EOR_EOP , n_end_demod?EOR_MPS , n_end_demod?EOR_Null , n_end_demod?
                EOR_PRIEOM ,
        n_end_demod?EOR_PRIEOP, n_end_demod?EOR_PRIMPS , n_end_demod?ERR, n_end_demod?FT
        n_end_demod?MCF, n_end_demod?NSC, n_end_demod?NSS, n_end_demod?NSS_F0,
        n_end_demod?NSS_FX, n_end_demod?PIN, n_end_demod?PIP, n_end_demod?PPR,
        n_end_demod?PPS_PRIEOM, n_end_demod?PPS_PRIEOP, n_end_demod?PPS_PRIMPS,
        n_end_demod?PRIEOM, n_end_demod?PRIEOP, n_end_demod?PRIMPS, n_end_demod?RTN ,
        n_end_demod?RTP , n_end_demod?v21,
        timer_ra2!start , timer_ra2?expired , timer_rb2!start ,
        timer_rb2?expired, timer_rc2!start, timer_rc2?expired, timer_rd1!start, timer_rd1?expired,
        timer_re1!start, timer_re1?expired, timer_rg1!start , timer_rg1?expired, timer_rg2!start ,
        timer_rg2?expired, timer_rg3!start, timer_rg3?expired, timer_ta2!start ,
        timer_ta2?expired, timer_tb2!start, timer_tb2?expired, timer_tc2!start, timer_tc2?expired,
        timer_td1!start, timer_td1?expired, timer_te1!start, timer_te1?expired, timer_tg1!start,
        timer_tg1?expired, timer_tg2!start, timer_tg2?expired, timer_tg3!start,
        timer_tg3?expired, user_orig?start, user_orig?stop;

INITIAL STATE 0;
TRANSITIONS

        /* Figure E.1 */

0:1     WHEN user_orig?start,
1:0     WHEN user_orig?stop,

    /* State 1 is the WAIT_FOR_FAX (a1)  State */

1:2     WHEN n_end_demod?DIS [allocate_modem();],
2:1     WHEN {allocate_term_ok != TRUE},
2:100   WHEN {allocate_term_ok == TRUE} timer_ra2!start, /* role = REMOD */
1:3     WHEN f_end_demod?DIS [allocate_modem();],
3:1     WHEN {allocate_orig_ok != TRUE},
3:10    WHEN {allocate_orig_ok == TRUE} timer_ta2!start, /* role = DEMOD */

        /* Figure E.2 */

    /* State 10 is the WAIT_FOR_DCS_FROM_NEAR_END (t.a2) State */

10:11   WHEN n_end_demod?DCS [check_compat();],
10:11   WHEN n_end_demod?NSS [check_compat();],
10:14   WHEN n_end_demod?DTC,
10:14   WHEN n_end_demod?NSC,
11:13   WHEN {fax_ok != TRUE},
11:12   WHEN {fax_ok == TRUE} demux!fadcomp [set_modem_param(); part_page_req_cnt=0;],
12:20   WHEN mod_layer_orig!ml_fax_start_request,
14:100  WHEN timer_ra2!start, /* role = REMOD */
10:13   WHEN timer_ta2?expired,
10:13   WHEN f_end_demod?DCN,
10:13   WHEN f_end_demod?DIS,
10:13   WHEN n_end_demod?DCN,
10:13   WHEN n_end_demod?DIS,
13:1    WHEN [deallocate_resources();],
```

```
        /* Figure E.3 */

    /* State 20 is the WAIT_FOR_TCF_TO_END (t.b1) State */

20:21   WHEN mod_layer_orig?ml_fax_abort_confirmation,
20:22   WHEN f_end_demod?v21,
20:22   WHEN n_end_demod?v21 ,
22:21   WHEN mod_layer_orig!ml_fax_stop_request,
21:1    WHEN mux!pvp * demux!pvp [deallocate_resources();],
20:30   WHEN mod_layer_orig?ml_fax_stop_confirmation * mux!pvp * demux!pvp * timer_tb2!start,

        /* Figure E.4 */

    /* State 30 is the WAIT_FOR_CFR (t.b2) State */

30:31   WHEN f_end_demod?CRP,
30:31   WHEN f_end_demod?FTT,
30:31   WHEN f_end_demod?NSC,
30:31   WHEN f_end_demod?DTC,
31:10   WHEN timer_ta2!start,
30:32   WHEN n_end_demod?DCS ,
30:32   WHEN n_end_demod?NSS_FX,
32:33   WHEN mux!pvp * demux!pvp,
30:33   WHEN timer_tb2?expired,
30:33   WHEN f_end_demod?DCN,
30:33   WHEN f_end_demod?DIS,
30:33   WHEN n_end_demod?DCN,
30:33   WHEN n_end_demod?DIS,
33:1    WHEN [deallocate_resources();],
30:35   WHEN f_end_demod?CFR * demux!fadcomp * mod_layer_orig!ml_fax_start_request,

        /* Figure E.5 */

    /* State 35 is the WAIT_FOR_END_OF_PAGE (t.c1) State */

35:36   WHEN f_end_demod?v21,
35:36   WHEN n_end_demod?v21,
35:37   WHEN mod_layer_orig?ml_fax_abort_confirmation,
36:37   WHEN mod_layer_orig!ml_fax_stop_request,
37:1    WHEN mux!pvp * demux!pvp [deallocate_resources();],
35:38   WHEN mod_layer_orig?ml_fax_stop_confirmation * mux!pvp * demux!pvp * timer_tc2!start,

        /* Figure E.6 */

    /* State 38 is the WAIT_FOR_POST_PAGE_CMD (t.c2) State */

38:39   WHEN timer_tc2?expired,
38:39   WHEN f_end_demod?DCN,
38:39   WHEN f_end_demod?DIS,
38:39   WHEN n_end_demod?DCN,
38:39   WHEN n_end_demod?DIS,
38:40   WHEN n_end_demod?v21 * timer_td1!start [store_cmd_type();],
39:1    WHEN [deallocate_resources();],

        /* Figure E.7 */
```

```
    /* State 40 is the WAIT_FOR_POST_PAGE_RESPONSE (t.d1) State */

    /* END OF MESSAGE */

40:41  WHEN {(near_cmd_type == EOP) || (near_cmd_type == PRIEOP) },
41:42  WHEN n_end_demod?MCF,
41:42  WHEN f_end_demod?DCN,
41:42  WHEN f_end_demod?DIS,
41:42  WHEN n_end_demod?DCN,
41:42  WHEN n_end_demod?DIS,
41:42  WHEN timer_td1?expired,
42:1   WHEN [deallocate_resources();],
40:47  WHEN { (near_cmd_type == PPSEOP)|| (near_cmd_type == PPS_PRIEOP)},
47:42  WHEN f_end_demod?MCF,
47:42  WHEN f_end_demod?DCN,
47:42  WHEN f_end_demod?DIS,
47:42  WHEN n_end_demod?DCN,
47:42  WHEN n_end_demod?DIS,
47:42  WHEN timer_td1?expired,
40:42  WHEN f_end_demod?DCN,
40:42  WHEN f_end_demod?DIS,
40:42  WHEN n_end_demod?DCN,
40:42  WHEN n_end_demod?DIS,
40:42  WHEN timer_td1?expired,
40:43  WHEN {(near_cmd_type == EOM) || (near_cmd_type == PRIEOM) },
43:1   WHEN f_end_demod?RTN * timer_ta2!start,
43:1   WHEN f_end_demod?RTP * timer_ta2!start,
43:1   WHEN n_end_demod?MCF * timer_ta2!start,
43:42  WHEN f_end_demod?DCN,
43:42  WHEN f_end_demod?DIS,
43:42  WHEN n_end_demod?DCN,
43:42  WHEN n_end_demod?DIS,
43:42  WHEN timer_td1?expired,
40:50  WHEN {(near_cmd_type == PPSEOM) || (near_cmd_type == PPS_PRIEOM) },
50:1   WHEN n_end_demod?MCF * timer_ta2!start,
50:42  WHEN f_end_demod?DCN,
50:42  WHEN f_end_demod?DIS,
50:42  WHEN n_end_demod?DCN,
50:42  WHEN n_end_demod?DIS,
50:42  WHEN timer_td1?expired,


    /*    MULTIPLE PAGES */

40:49  WHEN {(near_cmd_type == MPS) || (near_cmd_type == PRIMPS) },
49:45  WHEN f_end_demod?MCF * demux!fadcomp [part_page_req_cnt=0;],
49:10  WHEN f_end_demod?RTN * timer_ta2!start,
49:10  WHEN f_end_demod?RTP * timer_ta2!start,
40:44  WHEN {(near_cmd_type == PPSNull) || (near_cmd_type == PPSMPS) ||
         (near_cmd_type == PPS_PRIMPS)},
44:42  WHEN timer_td1?expired,
44:42  WHEN f_end_demod?DCN,
44:42  WHEN f_end_demod?DIS,
44:42  WHEN n_end_demod?DCN,
44:42  WHEN n_end_demod?DIS,
```

```
44:45   WHEN f_end_demod?MCF * demux!fadcomp [part_page_req_cnt=0;],
45:35   WHEN mod_layer_orig!ml_fax_start_request,
40:46   WHEN f_end_demod?PPR,
46:45   WHEN {part_page_req_cnt < 3}  demux!fadcomp [part_page_req_cnt++;],
46:70   WHEN {part_page_req_cnt == 3} timer_tg1!start [part_page_req_cnt=0;],

    /* PROCEDURE TO RETRAIN */

41:10   WHEN f_end_demod?RTN * timer_ta2!start,
41:10   WHEN f_end_demod?RTP * timer_ta2!start,



    /* PROCEDURE INTERRUPT */

40:48   WHEN f_end_demod?PIP,
40:48   WHEN f_end_demod?PIN,
48:60   WHEN timer_te1!start,

        /* Figure E.8  */

   /*  State 60 is the WAIT_FOR_CMD (t.e1) State */

60:10   WHEN n_end_demod?NSS_F0 * timer_ta2!start,
60:61   WHEN n_end_demod?NSC,
60:61   WHEN n_end_demod?DTC,
61:100  WHEN timer_ra2!start, /* role = REMOD */
60:62   WHEN n_end_demod?DCS [check_compat();],
60:62   WHEN n_end_demod?NSS [check_compat();],
62:64   WHEN {fax_ok != TRUE},
62:63   WHEN {fax_ok == TRUE} demux!fadcomp [set_modem_param(); part_page_req_cnt=0;],
63:20   WHEN mod_layer_orig!ml_fax_start_request,
60:64   WHEN timer_te1?expired,
60:64   WHEN f_end_demod?DCN,
60:64   WHEN f_end_demod?DIS,
60:64   WHEN n_end_demod?DCN,
60:64   WHEN n_end_demod?DIS,
60:64   WHEN n_end_demod?EOR_PRIEOM,
60:64   WHEN n_end_demod?EOR_PRIEOP,
60:64   WHEN n_end_demod?EOR_PRIMPS,
60:64   WHEN n_end_demod?PPS_PRIEOM,
60:64   WHEN n_end_demod?PPS_PRIEOP,
60:64   WHEN n_end_demod?PPS_PRIMPS,
60:64   WHEN n_end_demod?PRIEOM,
60:64   WHEN n_end_demod?PRIEOP,
60:64   WHEN n_end_demod?PRIMPS,
64:1    WHEN [deallocate_resources();],

        /* Figure E.9 */

   /*  State 70 is the WAIT_FOR_CTC_OR_EOR.X State (t.g1) */

70:71   WHEN timer_tg1?expired,
70:71   WHEN f_end_demod?DCN,
70:71   WHEN f_end_demod?DIS,
```

```
70:71   WHEN n_end_demod?DCN,
70:71   WHEN n_end_demod?DIS,
71:1    WHEN [deallocate_resources();],
70:80   WHEN n_end_demod?CTC * timer_tg2!start [set_modem_param();],
70:72   WHEN n_end_demod?EOR_PRIEOM [near_eor_type=PRIEOM;],
70:72   WHEN n_end_demod?EOR_PRIEOP [near_eor_type=PRIEOP;],
70:72   WHEN n_end_demod?EOR_PRIMPS [near_eor_type=PRIMPS;],
70:72   WHEN n_end_demod?EOR_EOM [near_eor_type=EOM;],
70:72   WHEN n_end_demod?EOR_EOP [near_eor_type=EOP;],
70:72   WHEN n_end_demod?EOR_Null [near_eor_type=Null;],
70:72   WHEN n_end_demod?EOR_MPS [near_eor_type=MPS;],
72:90   WHEN timer_tg3!start,

        /* Figure E.10 */

    /*  State 80 is the WAIT_FOR_CTR (t.g2) State */

80:83   WHEN timer_tg2?expired,
80:83   WHEN f_end_demod?DCN,
80:83   WHEN f_end_demod?DIS,
80:83   WHEN n_end_demod?DCN,
80:83   WHEN n_end_demod?DIS,
80:81   WHEN f_end_demod?CTR [check_compat();],
81:82   WHEN {fax_ok == TRUE} demux!fadcomp [set_modem_param(); part_page_req_cnt=0;],
81:83   WHEN {fax_ok != TRUE},
82:35   WHEN mod_layer_orig!ml_fax_start_request,
83:1    WHEN [deallocate_resources();],

        /* Figure E.11 */

    /*  State 90 is the WAIT_FOR_ERR (t.g3) State */

90:50   WHEN f_end_demod?PIN * timer_te1!start,
90:91   WHEN timer_tg3?expired,
90:91   WHEN f_end_demod?DCN,
90:91   WHEN f_end_demod?DIS,
90:91   WHEN n_end_demod?DCN,
90:91   WHEN n_end_demod?DIS,
90:92   WHEN f_end_demod?ERR,
92:10   WHEN {(near_eor_type == EOM)||(near_eor_type == PRIEOM)} timer_ta2!start,
92:93   WHEN {(near_eor_type==Null)||(near_eor_type ==MPS)||(near_eor_type==PRIMPS)}
            demux!fadcomp [set_modem_param();],
93:35   WHEN mod_layer_orig!ml_fax_start_request,
92:1    WHEN {(near_eor_type == EOP)||(near_eor_type == PRIEOP)}  [deallocate_resources();],
91:1    WHEN [deallocate_resources();],


        /* Figure E.12 */

    /* State 100 is the WAIT_FOR_DCS_FROM_FAR_END (r.a2)  State */

100:111         WHEN f_end_demod?DCS [check_compat();],
100:111         WHEN f_end_demod?NSS [check_compat();],
111:112         WHEN {fax_ok == TRUE} demux!off * mux!fadcomp [set_modem_param(); part_page_req_cnt=(
112:120         WHEN mod_layer_term!ml_fax_start_response,
```

```
111:113         WHEN {fax_ok != TRUE},
100:114         WHEN f_end_demod?DTC,
100:114         WHEN f_end_demod?NSC,
114:10  WHEN timer_ta2!start, /* role=DEMOD */
100:113         WHEN timer_ra2?expired,
100:113         WHEN f_end_demod?DCN,
100:113         WHEN f_end_demod?DIS,
100:113         WHEN n_end_demod?DCN,
100:113         WHEN n_end_demod?DIS,
113:1   WHEN [deallocate_resources();],


        /* Figure E.13 */

    /* State 120 is the WAIT_FOR_TCF_TO_END (r.b1) State */

120:121         WHEN mod_layer_term?ml_fax_abort_indication,
120:122         WHEN f_end_demod?v21,
120:122         WHEN n_end_demod?v21 ,
121:1   WHEN mux!pvp * demux!pvp  [deallocate_resources();],
122:121         WHEN mod_layer_term!ml_fax_stop_response,
120:130         WHEN mod_layer_term?ml_fax_stop_indication * mux!pvp * demux!pvp * timer_rb2!start
        [deallocate_resources();],

        /* Figure E.14 */

    /* State 130 is the WAIT_FOR_CFR (r.b2) State */

130:131         WHEN n_end_demod?CRP,
130:131         WHEN n_end_demod?FTT,
130:131         WHEN n_end_demod?NSC,
130:131         WHEN n_end_demod?DTC,
131:100         WHEN timer_ra2!start,
130:132         WHEN f_end_demod?DCS ,
130:132         WHEN f_end_demod?NSS_FX,
132:133         WHEN mux!pvp * demux!pvp,
130:133         WHEN timer_rb2?expired,
130:133         WHEN n_end_demod?DCN,
130:133         WHEN n_end_demod?DIS,
130:133         WHEN f_end_demod?DCN,
130:133         WHEN f_end_demod?DIS,
133:1   WHEN [deallocate_resources();],
130:135         WHEN n_end_demod?CFR * mux!fadcomp * demux!off* mod_layer_term!ml_fax_start_response,

        /* Figure E.15 */

    /* State 135 is the WAIT_FOR_END_OF_PAGE (r.c1) State */

135:136         WHEN f_end_demod?v21,
135:136         WHEN n_end_demod?v21,
135:137         WHEN mod_layer_term?ml_fax_abort_indication,
136:137         WHEN mod_layer_term!ml_fax_stop_response,
137:1   WHEN mux!pvp * demux!pvp [deallocate_resources();],
135:138         WHEN mod_layer_term?ml_fax_stop_indication * mux!pvp * demux!pvp * timer_rc2!start,
```

```
        /* Figure E.16 */

    /* State 138 is the WAIT_FOR_POST_PAGE_CMD (r.c2) State */

138:139         WHEN timer_rc2?expired,
138:139         WHEN f_end_demod?DCN,
138:139         WHEN f_end_demod?DIS,
138:139         WHEN n_end_demod?DCN,
138:139         WHEN n_end_demod?DIS,
138:140         WHEN f_end_demod?v21 * timer_rd1!start [store_cmd_type();],
139:1   WHEN [deallocate_resources();],

        /* Figure E.17 */

    /* State 140 is the WAIT_FOR_POST_PAGE_RESPONSE (r.d1) State */

     /* END OF MESSAGE */

140:141         WHEN {(far_cmd_type == EOP) || (far_cmd_type == PRIEOP) },
141:142         WHEN n_end_demod?MCF,
141:142         WHEN f_end_demod?DCN,
141:142         WHEN f_end_demod?DIS,
141:142         WHEN n_end_demod?DCN,
141:142         WHEN n_end_demod?DIS,
141:142         WHEN timer_rd1?expired,
142:1   WHEN [deallocate_resources();],
140:147         WHEN { (far_cmd_type == PPSEOP)|| (far_cmd_type == PPS_PRIEOP)},
147:142         WHEN n_end_demod?MCF,
147:142         WHEN f_end_demod?DCN,
147:142         WHEN f_end_demod?DIS,
147:142         WHEN n_end_demod?DCN,
147:142         WHEN n_end_demod?DIS,
147:142         WHEN timer_rd1?expired,
140:142         WHEN f_end_demod?DCN,
140:142         WHEN f_end_demod?DIS,
140:142         WHEN n_end_demod?DCN,
140:142         WHEN n_end_demod?DIS,
140:142         WHEN timer_rd1?expired,
140:143         WHEN {(far_cmd_type == EOM) || (far_cmd_type == PRIEOM) },
143:100         WHEN n_end_demod?RTN * timer_ra2!start,
143:100         WHEN n_end_demod?RTP * timer_ra2!start,
143:100         WHEN n_end_demod?MCF * timer_ra2!start,
143:142         WHEN f_end_demod?DCN,
143:142         WHEN f_end_demod?DIS,
143:142         WHEN n_end_demod?DCN,
143:142         WHEN n_end_demod?DIS,
143:142         WHEN timer_rd1?expired,
140:150         WHEN {(far_cmd_type == PPSEOM) || (far_cmd_type == PPS_PRIEOM) },
150:100         WHEN n_end_demod?MCF * timer_ra2!start,
150:142         WHEN f_end_demod?DCN,
150:142         WHEN f_end_demod?DIS,
150:142         WHEN n_end_demod?DCN,
150:142         WHEN n_end_demod?DIS,
150:142         WHEN timer_rd1?expired,
```

```
    /*      MULTIPLE PAGES */

140:149         WHEN {(far_cmd_type == MPS) || (far_cmd_type == PRIMPS) },
149:145         WHEN n_end_demod?MCF * mux!fadcomp * demux!off [part_page_req_cnt=0;],
149:100         WHEN n_end_demod?RTN * timer_ra2!start,
149:100         WHEN n_end_demod?RTP * timer_ra2!start,
140:144         WHEN {(far_cmd_type == PPSNull) || (far_cmd_type == PPSMPS) ||
          (far_cmd_type == PPS_PRIMPS)},
144:142         WHEN timer_rd1?expired,
144:142         WHEN f_end_demod?DCN,
144:142         WHEN f_end_demod?DIS,
144:142         WHEN n_end_demod?DCN,
144:142         WHEN n_end_demod?DIS,
144:145         WHEN n_end_demod?MCF * mux!fadcomp * demux!off [part_page_req_cnt=0;],
145:135         WHEN mod_layer_orig!ml_fax_start_request,
140:146         WHEN n_end_demod?PPR,
146:145         WHEN {part_page_req_cnt < 3}  demux!fadcomp [part_page_req_cnt++;],
146:170         WHEN {part_page_req_cnt == 3} timer_rg1!start [part_page_req_cnt=0;],

    /* PROCEDURE TO RETRAIN */

141:100         WHEN n_end_demod?RTN * timer_ra2!start,
141:100         WHEN n_end_demod?RTP * timer_ra2!start,

    /* PROCEDURE INTERRUPT */

140:148         WHEN n_end_demod?PIP,
140:148         WHEN n_end_demod?PIN,
148:160         WHEN timer_re1!start,

        /* Figure E.18 */

    /*  State 160 is the WAIT_FOR_CMD (r.e1) State */

160:100         WHEN f_end_demod?NSS_F0 * timer_ra2!start,
160:161         WHEN f_end_demod?NSC,
160:161         WHEN f_end_demod?DTC,
161:10  WHEN timer_ta2!start, /* role = DEMOD */
160:162         WHEN f_end_demod?DCS [check_compat();],
160:162         WHEN f_end_demod?NSS [check_compat();],
162:164         WHEN {fax_ok != TRUE},
162:163         WHEN {fax_ok == TRUE} mux!fadcomp * demux!off [set_modem_param(); part_page_req_cnt=0;],
163:120         WHEN mod_layer_term!ml_fax_start_response,
160:164         WHEN timer_re1?expired,
160:164         WHEN f_end_demod?DCN,
160:164         WHEN f_end_demod?DIS,
160:164         WHEN n_end_demod?DCN,
160:164         WHEN n_end_demod?DIS,
160:164         WHEN f_end_demod?EOR_PRIEOM,
160:164         WHEN f_end_demod?EOR_PRIEOP,
160:164         WHEN f_end_demod?EOR_PRIMPS,
160:164         WHEN f_end_demod?PPS_PRIEOM,
160:164         WHEN f_end_demod?PPS_PRIEOP,
160:164         WHEN f_end_demod?PPS_PRIMPS,
160:164         WHEN f_end_demod?PRIEOM,
```

```
160:164         WHEN f_end_demod?PRIEOP,
160:164         WHEN f_end_demod?PRIMPS,
164:1   WHEN [deallocate_resources();],

        /* Figure E.19 */

    /*  State 170 is the WAIT_FOR_CTC_OR_EOR.X (r.g1) State */

170:171         WHEN timer_rg1?expired,
170:171         WHEN f_end_demod?DCN,
170:171         WHEN f_end_demod?DIS,
170:171         WHEN n_end_demod?DCN,
170:171         WHEN n_end_demod?DIS,
171:1   WHEN [deallocate_resources();],
170:180         WHEN f_end_demod?CTC * timer_rg2!start [set_modem_param();],
170:172         WHEN f_end_demod?EOR_PRIEOM [far_eor_type= PRIEOM;],
170:172         WHEN f_end_demod?EOR_PRIEOP [far_eor_type=PRIEOP;],
170:172         WHEN f_end_demod?EOR_PRIMPS [far_eor_type=PRIMPS;],
170:172         WHEN f_end_demod?EOR_EOM [far_eor_type=EOM;],
170:172         WHEN f_end_demod?EOR_EOP [far_eor_type=EOP;],
170:172         WHEN f_end_demod?EOR_Null [far_eor_type=Null;],
170:172         WHEN f_end_demod?EOR_MPS [far_eor_type=MPS;],
172:190         WHEN timer_rg3!start,

        /* Figure E.20 */

    /*  State 180 is the WAIT_FOR_CTR (r.g2) State */

180:183         WHEN timer_rg2?expired,
180:183         WHEN f_end_demod?DCN,
180:183         WHEN f_end_demod?DIS,
180:183         WHEN n_end_demod?DCN,
180:183         WHEN n_end_demod?DIS,
180:181         WHEN n_end_demod?CTR [check_compat();],
181:182         WHEN {fax_ok == TRUE} mux!fadcomp * demux!off [set_modem_param(); part_page_req_cnt=(
181:183         WHEN {fax_ok != TRUE},
182:135         WHEN mod_layer_term!ml_fax_start_response,
183:1   WHEN [deallocate_resources();],

        /* Figure E.21 */

    /*  State 190 is the WAIT_FOR_ERR (r.g3) State */

190:150         WHEN n_end_demod?PIN * timer_re1!start,
190:191         WHEN timer_rg3?expired,
190:191         WHEN f_end_demod?DCN,
190:191         WHEN f_end_demod?DIS,
190:191         WHEN n_end_demod?DCN,
190:191         WHEN n_end_demod?DIS,
190:192         WHEN n_end_demod?ERR,
192:100         WHEN {(far_eor_type==EOM)||(far_eor_type ==PRIEOM)} timer_ra2!start,
192:193         WHEN {(far_eor_type==Null)||(far_eor_type == MPS)||(far_eor_type == PRIMPS)}
                mux!fadcomp * demux!off [set_modem_param();],
193:135         WHEN mod_layer_term!ml_fax_start_response,
192:1   WHEN {(far_eor_type==EOP)||(far_eor_type == PRIEOP)}  [deallocate_resources();],
```

```
191:1   WHEN [deallocate_resources();].
/*
                Modulation Layer at the Originating End
*/

PROCESS         mod_layer_orig;
STATES          0-9;
REND    call_state!ml_fax_abort_confirmation,
        call_state!ml_fax_stop_confirmation,
        call_state?ml_fax_start_request,
        call_state?ml_fax_stop_request,
        ept_detector?ept_tone,
        ept_timer_orig!start, ept_timer_orig!stop, ept_timer_orig?expired,
        n_end_demod?demod_error, n_end_demod?end_of_data, n_end_demod!start,
        training_detector?train,
        training_detector?training_successful,
        training_detector?training_failed,
        xmtr_fax!pl_fax_spurt_header_abort,
        xmtr_fax!pl_fax_data_start_request,
        xmtr_fax!pl_fax_data_stop_request,
        xmtr_fax!pl_fax_spurt_header_ept_start,
        xmtr_fax!pl_fax_spurt_header_ept_stop,
        xmtr_fax!pl_fax_spurt_header_train,
        wait_time!start, wait_time!stop, wait_time?expired;

INITIAL STATE 0;
TRANSITIONS

                /* Figure E.22 (a) */

0:1     WHEN call_state?ml_fax_start_request * wait_time!start,
1:2     WHEN wait_time?expired * call_state!ml_fax_abort_confirmation,
2:0     WHEN xmtr_fax!pl_fax_spurt_header_abort,
1:0     WHEN call_state?ml_fax_stop_request * wait_time!stop,
1:3     WHEN ept_detector?ept_tone * ept_timer_orig!start * wait_time!stop [check_ept();],
3:4     WHEN xmtr_fax!pl_fax_spurt_header_ept_start,
4:0     WHEN call_state?ml_fax_stop_request * xmtr_fax!pl_fax_spurt_header_ept_stop * ept_timer_orig!stop,
4:2     WHEN ept_timer_orig?expired * call_state!ml_fax_abort_confirmation,
1:5     WHEN training_detector?train * xmtr_fax!pl_fax_spurt_header_train * wait_time!stop,
5:0     WHEN call_state?ml_fax_stop_request,
5:6     WHEN training_detector?training_successful,
5:7     WHEN training_detector?training_failed * xmtr_fax!pl_fax_spurt_header_abort,

                /* Figure E.22 (b) */

7:0     WHEN call_state!ml_fax_abort_confirmation,
6:8     WHEN xmtr_fax!pl_fax_data_start_request * n_end_demod!start,
8:0     WHEN call_state?ml_fax_stop_request,
8:7     WHEN n_end_demod?demod_error * xmtr_fax!pl_fax_spurt_header_abort,
8:9     WHEN n_end_demod?end_of_data * xmtr_fax!pl_fax_data_stop_request,
9:0     WHEN call_state!ml_fax_stop_confirmation.
/*
        Originating End  XMTR – Figure E.23 (a)
                        */
```

```
PROCESS      xmtr_fax;
CONTEXT action_orig,ept_type_orig,mbit_orig, no_bits_orig,
    seq_orig,type_orig,uih,bilo_orig;
STATES       0-12;
REND  mod_layer_orig?pl_fax_spurt_header_abort,
      mod_layer_orig?pl_fax_data_start_request,
      mod_layer_orig?pl_fax_data_stop_request,
      mod_layer_orig?pl_fax_spurt_header_ept_start,
      mod_layer_orig?pl_fax_spurt_header_ept_stop,
      mod_layer_orig?pl_fax_spurt_header_train,
      t_accumulator!start, t_accumulator?expired,
      tvdelay_orig!start, tvdelay_orig!stop,
      layer2_orig?dl_l1_ready_indication,
      layer2_orig!dl_unit_data_request,
      layer2_orig!dl_unit_h_data_request;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  mod_layer_orig?pl_fax_spurt_header_ept_start [type_orig=0;seq_orig=0; bilo_orig=0;],
1:2     WHEN  {ept_type_orig ==1700} [action_orig=EPT_1700;],
1:2     WHEN  {ept_type_orig ==1800} [action_orig=EPT_1800;],
0:2     WHEN  mod_layer_orig?pl_fax_spurt_header_ept_stop [type_orig=0;seq_orig=0;
              bilo_orig=0;action_orig=STOP_EPT;],
0:2     WHEN  mod_layer_orig?pl_fax_spurt_header_train [type_orig=0;seq_orig=0;
              bilo_orig=0;action_orig=TRAINING;],
0:2     WHEN  mod_layer_orig?pl_fax_spurt_header_abort [type_orig=0;seq_orig=0;
              bilo_orig=0;action_orig=ABORT;],
2:3     WHEN  tvdelay_orig!start,
3:4     WHEN  layer2_orig?dl_l1_ready_indication * tvdelay_orig!stop,
4:5     WHEN  [ts_update();] ,
5:0     WHEN  {uih == FALSE}layer2_orig!dl_unit_data_request,
5:0     WHEN  {uih == TRUE}layer2_orig!dl_unit_h_data_request,

0:6     WHEN  mod_layer_orig?pl_fax_data_start_request * t_accumulator!start [type_orig=1;mbit_orig = 1;],
/*
        Figure E.23 (b)
                    */

6:7     WHEN  t_accumulator?expired,
7:8     WHEN  {no_bits_orig %8 == 0 } [bilo_orig=0;],
7:8     WHEN  {no_bits_orig %8 != 0 } [bilo_orig=8-no_bits_orig;],
6:7     WHEN  mod_layer_orig?pl_fax_data_stop_request [mbit_orig = 0;],
8:9     WHEN  tvdelay_orig!start,
9:10    WHEN  layer2_orig?dl_l1_ready_indication * tvdelay_orig!stop,
10:11   WHEN  [ts_update();] ,
11:12   WHEN  {uih == FALSE}layer2_orig!dl_unit_data_request [seq_increment();],
11:12   WHEN  {uih == TRUE}layer2_orig!dl_unit_h_data_request [seq_increment();],
12:6    WHEN  {mbit_orig== 1} t_accumulator!start,
12:0    WHEN  {mbit_orig== 0}.
/*
      Intermediate Point -- Figure E.24 --
                                          */

PROCESS      interm_pt_fax_relay;
```

```
CONTEXT pd, uih;
STATES      0-5;
REND  layer2_interm?dl_pvp_data_indication,
      layer2_interm?dl_pvp_h_data_indication,
      layer2_interm!dl_pvp_data_request,
      layer2_interm!dl_pvp_h_data_request,
      layer2_interm?dl_l1_ready_indication,
      tvdelay_interm!start, tvdelay_interm!stop;

INITIAL STATE 0;
TRANSITIONS

0:1   WHEN  layer2_interm?dl_pvp_h_data_indication [uih = TRUE;],
0:1   WHEN  layer2_interm?dl_pvp_data_indication [uih = FALSE;],
1:0   WHEN  {pd != PVP},
1:2   WHEN  {pd == PVP} tvdelay_interm!start,
2:3   WHEN  layer2_interm?dl_l1_ready_indication * tvdelay_interm!stop [ts_update();],
3:0   WHEN  {uih == FALSE} layer2_interm!dl_pvp_data_request,
3:0   WHEN  {uih == TRUE} layer2_interm!dl_pvp_h_data_request.


/*
      Receiver Terminating End      -- Figure E.25
                                    */
PROCESS     rcv_fax;
CONTEXT ts,pd,service_class;
STATES      0-3;
REND  layer2_term?dl_unit_h_data_indication,
      layer2_term?dl_unit_data_indication,
      fax_rcv_queue!write_pkt;

INITIAL STATE 0;
TRANSITIONS

0:1   WHEN  layer2_term?dl_unit_h_data_indication,
0:1   WHEN  layer2_term?dl_unit_data_indication,
1:0   WHEN  {pd != PVP},
1:2   WHEN  {pd == PVP},
2:3   WHEN  {service_class == 0x01 },
2:0   WHEN  {service_class != 0x01 },
3:0   WHEN  {ts > BUILDOUT},
3:0   WHEN  {ts <= BUILDOUT} fax_rcv_queue!write_pkt [set_play_out_time();].


/*
      FAX Receiver Queue    -- Figure E.26
                            */
PROCESS     fax_rcv_queue;
CONTEXT have_room,q_empty;
STATES      0-3;
REND  rcv_fax?write_pkt, playout_fax?scan,
      playout_fax!pkt_in, playout_fax!queue_empty,
      playout_fax?read_pkt;

INITIAL STATE 0;
```

```
TRANSITIONS

0:1     WHEN  rcv_fax?write_pkt,
1:0     WHEN  {have_room == TRUE} [push_pkt();],
1:0     WHEN  {have_room != TRUE} [drop_pkt();],
0:2     WHEN  playout_fax?scan,
2:0     WHEN  {q_empty == TRUE} playout_fax!queue_empty,
2:0     WHEN  {q_empty != TRUE} playout_fax!pkt_in,
0:3     WHEN  playout_fax?read_pkt,
3:0     WHEN  [set_pkt_length();pop_pkt();].



/*
        Playout FAX -- Figure E.27 (a)
                                */
PROCESS     playout_fax;
CONTEXT current_time, mbit_term, playout_time, pkt_length, rseq_term, seq_term,
        type_term;
STATES      0-3,10-14,20-25;
REND  clock?time,
        fax_rcv_queue!scan, fax_rcv_queue?pkt_in,fax_rcv_queue?queue_empty,
        fax_rcv_queue!read_pkt,
        mod_layer_term!pl_fax_data_indication,
        mod_layer_term!pl_fax_data_stop_indication,
        mod_layer_term!pl_fax_spurt_header_indication_action,
        user_term?start, user_term?stop;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  user_term?start [rseq_term=0;],
1:0     WHEN  user_term?stop,
1:2     WHEN  fax_rcv_queue!scan,
2:1     WHEN  fax_rcv_queue?queue_empty ,
2:3     WHEN  fax_rcv_queue?pkt_in * fax_rcv_queue!read_pkt,
3:10    WHEN  {seq_term == 0},
3:20    WHEN  {seq_term>0},

/*
        -- Figure E.27 (b) --

      Packet with sequence number of 0
                        */

10:11   WHEN  clock?time,
11:1    WHEN  {current_time >playout_time},
11:12   WHEN  {current_time <= playout_time},
12:11   WHEN  {current_time <playout_time} clock?time,
12:13   WHEN  {current_time == playout_time },
13:1    WHEN  {type_term == 0} mod_layer_term!pl_fax_spurt_header_indication_action [rseq_term=1;],
13:14   WHEN  {type_term > 0},
14:1    WHEN  {type_term > 1 },
14:1    WHEN  {type_term == 1} mod_layer_term!pl_fax_data_indication [rseq_term=1; get_bits();],
```

```
/*
        -- Figure E.27 (c) --

      Subsequent packets of page information
                              */
20:23   WHEN  {seq_term == rseq_term},
23:24   WHEN  {mbit_term == 1} mod_layer_term!pl_fax_data_indication [rseq_increment();get_bits();],
24:1    WHEN  {pkt_length == 0},
24:23   WHEN  {pkt_length > 0} [mod_symbol();pkt_length--;],
23:25   WHEN  {mbit_term == 0 } mod_layer_term!pl_fax_data_stop_indication [rseq_increment();get_bits();],
25:1    WHEN  {pkt_length ==0},
25:23   WHEN  {pkt_length > 0} [mod_symbol();pkt_length--;],
20:21   WHEN  {seq_term != rseq_term} clock?time,
21:1    WHEN  {current_time > playout_time },
21:22   WHEN  {current_time <= playout_time },
22:21   WHEN  {current_time < playout_time } clock?time,
22:23   WHEN  {current_time == playout_time }.
/*
        Modulation layer at the terminating end

                              */

PROCESS       mod_layer_term;
CONTEXT       action_term,ept_freq_term ;
STATES        0-5,10-12,20-25,30-31;
REND    bridge_timer!start, bridge_timer!stop, bridge_timer?expired,
        call_state!ml_fax_abort_indication,
        call_state!ml_fax_stop_indication,
        call_state?ml_fax_start_response,
        call_state?ml_fax_stop_response,
        ept_timer_term?expired, ept_timer_term!start, ept_timer_term!stop,
        idle_timer?expired, idle_timer!start,
        n_end_remod?done, n_end_remod!start,
        playout_fax?pl_fax_data_indication,
        playout_fax?pl_fax_data_stop_indication,
        playout_fax?pl_fax_spurt_header_indication_action,
        train_gen_term?end_of_sequence;

INITIAL STATE 0;
TRANSITIONS

/*
      OFF AND IDLE  STATES
        -- Figure E.28 --
*/

0:1     WHEN call_state?ml_fax_start_response * idle_timer!start [set_modem_param();],
1:0     WHEN idle_timer?expired * call_state!ml_fax_abort_indication,
1:0     WHEN call_state?ml_fax_stop_response,
1:2     WHEN playout_fax?pl_fax_spurt_header_indication_action,
2:3     WHEN {action_term == EPT_1700} [ept_freq_term=1700;],
2:3     WHEN {action_term == EPT_1800} [ept_freq_term = 1800;],
2:10    WHEN {action_term == TRAINING} [begin_training();],

2:0     WHEN {action_term == ABORT} call_state!ml_fax_abort_indication,
```

```
/*
      EPT  STATE
         -- Figure E.29 --
*/
3:4      WHEN ept_timer_term!start [generate_ept_tone();],
4:0      WHEN ept_timer_term?expired * call_state!ml_fax_abort_indication
            [stop_ept_tone();],
4:5      WHEN playout_fax?pl_fax_spurt_header_indication_action,
5:1      WHEN {action_term == EPT_STOP} ept_timer_term!stop [stop_ept_tone();],
5:0      WHEN {action_term == ABORT} call_state!ml_fax_abort_indication
            [stop_ept_tone();],
5:0      WHEN call_state?ml_fax_stop_response,

/*
      -- Figure E.30 --

         TRAIN STATE
*/

10:0     WHEN call_state?ml_fax_stop_response,
10:11    WHEN playout_fax?pl_fax_spurt_header_indication_action,
11:0     WHEN {action_term == ABORT} call_state!ml_fax_abort_indication
            [stop_training();],
11:10    WHEN {action_term != ABORT},
10:12    WHEN playout_fax?pl_fax_data_indication,
12:20    WHEN train_gen_term?end_of_sequence * n_end_remod!start,
12:0     WHEN call_state?ml_fax_stop_response,
10:30    WHEN train_gen_term?end_of_sequence * bridge_timer!start [generate_mod_bits();],

/*
      -- Figure E.31 --

         DATA STATE
*/

20:0     WHEN call_state?ml_fax_stop_response,
20:30    WHEN n_end_remod?done * bridge_timer!start [generate_mod_bits();],
20:23    WHEN playout_fax?pl_fax_data_stop_indication ,
23:0     WHEN n_end_remod?done * call_state!ml_fax_stop_indication,
20:23    WHEN playout_fax?pl_fax_spurt_header_indication_action,
23:0     WHEN call_state!ml_fax_abort_indication,

/*
      -- Figure E.32 --

      BRIDGE STATE
*/
30:0     WHEN call_state?ml_fax_stop_response * bridge_timer!stop,
30:0     WHEN bridge_timer?expired * call_state!ml_fax_abort_indication,
30:31    WHEN playout_fax?pl_fax_spurt_header_indication_action,
31:0     WHEN call_state!ml_fax_abort_indication,
30:20    WHEN playout_fax?pl_fax_data_indication * bridge_timer!stop *n_end_remod!start,
30:20    WHEN playout_fax?pl_fax_data_stop_indication * bridge_timer!stop *n_end_remod!start.
/*
         WAIT_TIME -- Figure E.33 --
```

```
                                        */
PROCESS      wait_time;
CONTEXT wait_for_signal_delay;
STATES       0-2;
REND  mod_layer_orig?start, mod_layer_orig!expired, mod_layer_orig?stop, clock?tick;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  mod_layer_orig?start [wait_for_signal_delay=0;],
1:2     WHEN  clock?tick  [wait_for_signal_delay++;],
2:1     WHEN  {wait_for_signal_delay < WAIT_FOR_SIGNAL_VALUE} ,
2:0     WHEN  mod_layer_orig?stop  [wait_for_signal_delay=0;],
2:0     WHEN  {wait_for_signal_delay == WAIT_FOR_SIGNAL_VALUE} mod_layer_orig!expired
              [wait_for_signal_delay=0;].

/*
        TVDELAY_ORIG -- Figure E.34 --
                                        */
PROCESS      tvdelay_orig;
CONTEXT delay_orig;
STATES       0-1;
REND  xmtr_fax?start, xmtr_fax?stop, clock?tick;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  xmtr_fax?start [delay_orig=0;],
1:1     WHEN  clock?tick [delay_orig++;],
1:0     WHEN  xmtr_fax?stop[output_delay();delay_orig=0;].

/*
        TVDELAY_INTERM -- Figure E.35 --
                                        */
PROCESS      tvdelay_interm;
CONTEXT delay_interm;
STATES       0-1;
REND  interm_pt_fax_relay?start, interm_pt_fax_relay?stop,
        clock?tick;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  interm_pt_fax_relay?start [delay_interm=0;],
1:1     WHEN  clock?tick [delay_interm++;],
1:0     WHEN  interm_pt_fax_relay?stop[output_delay();delay_interm=0;].
/*
        BRIDGE_TIME TIMER -- Figure E.36 --
                                        */
PROCESS      bridge_timer;
CONTEXT bridge_delay;
STATES       0-2;
REND  clock?tick, mod_layer_term?start, mod_layer_term?stop, mod_layer_term!expired;

INITIAL STATE 0;
```

```
TRANSITIONS

0:1     WHEN  mod_layer_term?start [bridge_delay=0;],
1:2     WHEN  clock?tick [bridge_delay++;],
2:1     WHEN  {bridge_delay < BRIDGE_TIME_VALUE} ,
2:0     WHEN  {bridge_delay == BRIDGE_TIME_VALUE} mod_layer_term!expired,
1:0     WHEN  mod_layer_term?stop [bridge_delay=0;].
/*
        EPT_TIMER_ORIG -- Figure E.37 --
                                        */
PROCESS     ept_timer_orig;
CONTEXT ept_delay;
STATES      0-2;
REND  mod_layer_orig?start, mod_layer_orig!expired, clock?tick;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  mod_layer_orig?start [ept_delay=0;],
1:2     WHEN  clock?tick  [ept_delay++;],
2:1     WHEN  {ept_delay < EPT_TIME_VALUE} ,
2:0     WHEN  {ept_delay == EPT_TIME_VALUE} mod_layer_orig!expired
              [ept_delay=0;].
/*
        EPT_TIMER_TERM -- Figure E.38 --
                                        */
PROCESS     ept_timer_term;
CONTEXT ept_delay;
STATES      0-2;
REND  mod_layer_term?start, mod_layer_term!expired, clock?tick;

INITIAL STATE 0;
TRANSITIONS

0:1     WHEN  mod_layer_term?start [ept_delay=0;],
1:2     WHEN  clock?tick  [ept_delay++;],
2:1     WHEN  {ept_delay < EPT_TIME_VALUE} ,
2:0     WHEN  {ept_delay == EPT_TIME_VALUE} mod_layer_term!expired
              [ept_delay=0;].
```

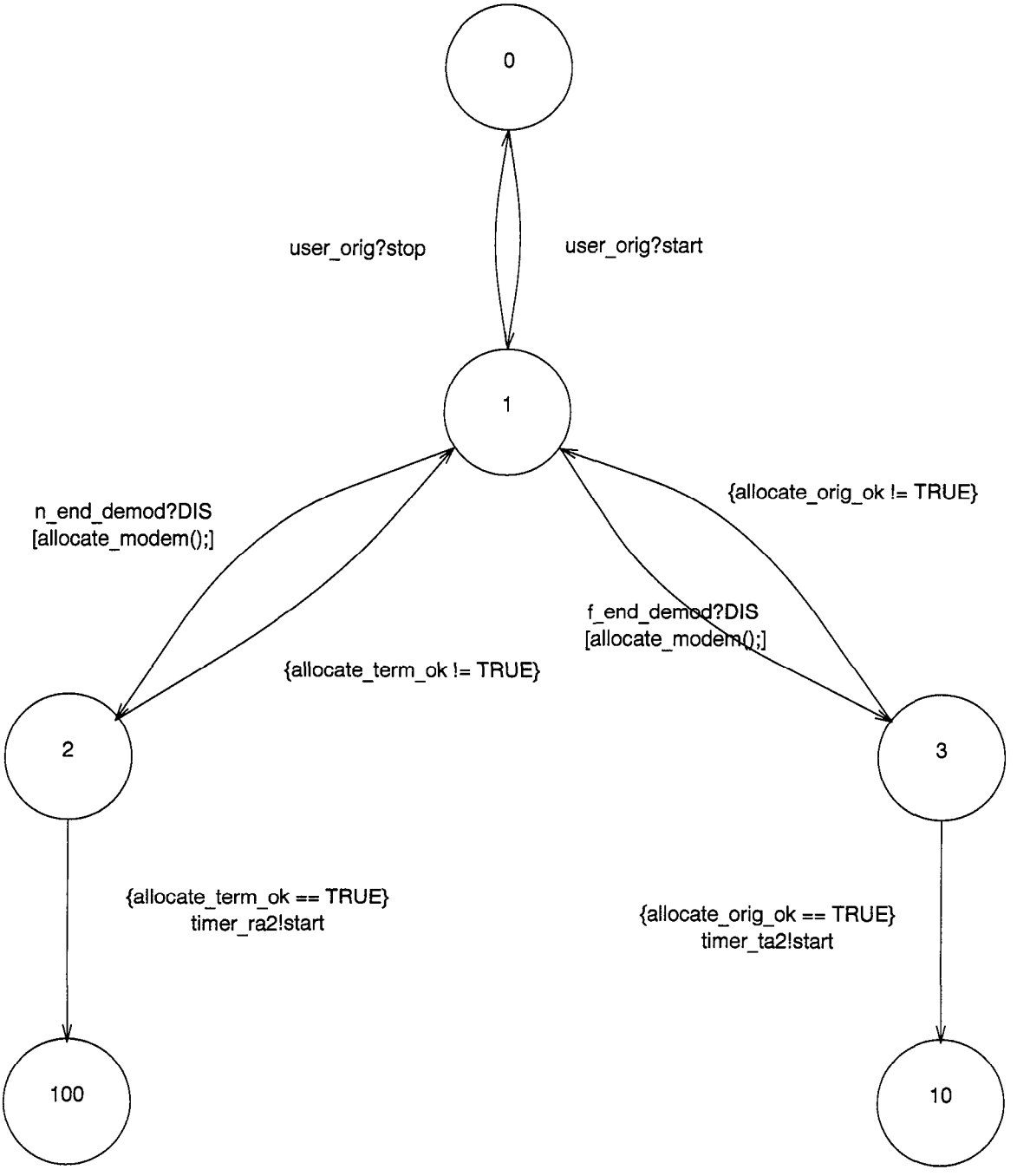


Figure E.1 – call_state: WAIT_FOR_FAX state (a1)

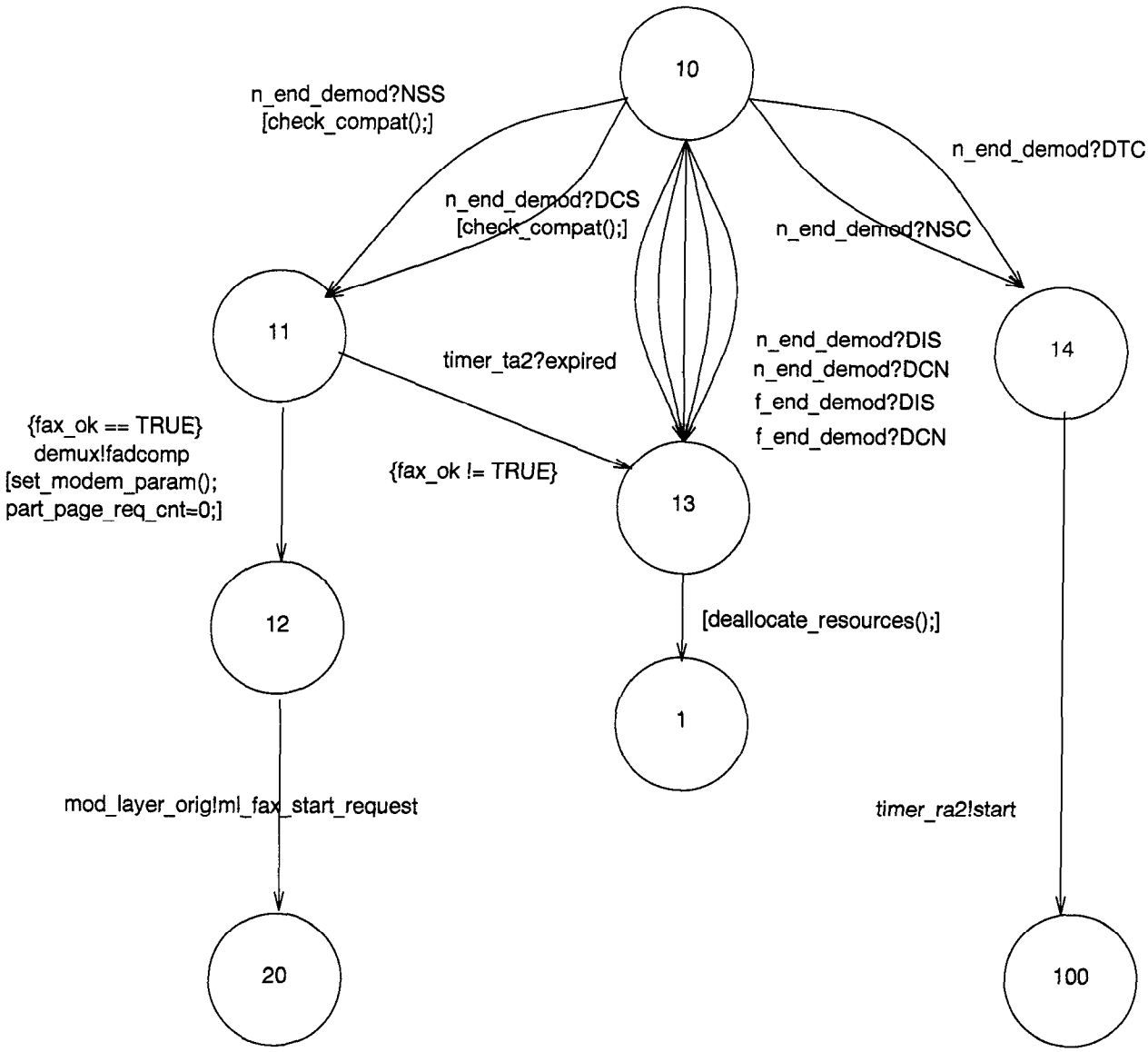


Figure E.2 – call_state: WAIT_FOR_DCS_FROM_NEAR_END state (t.a2)

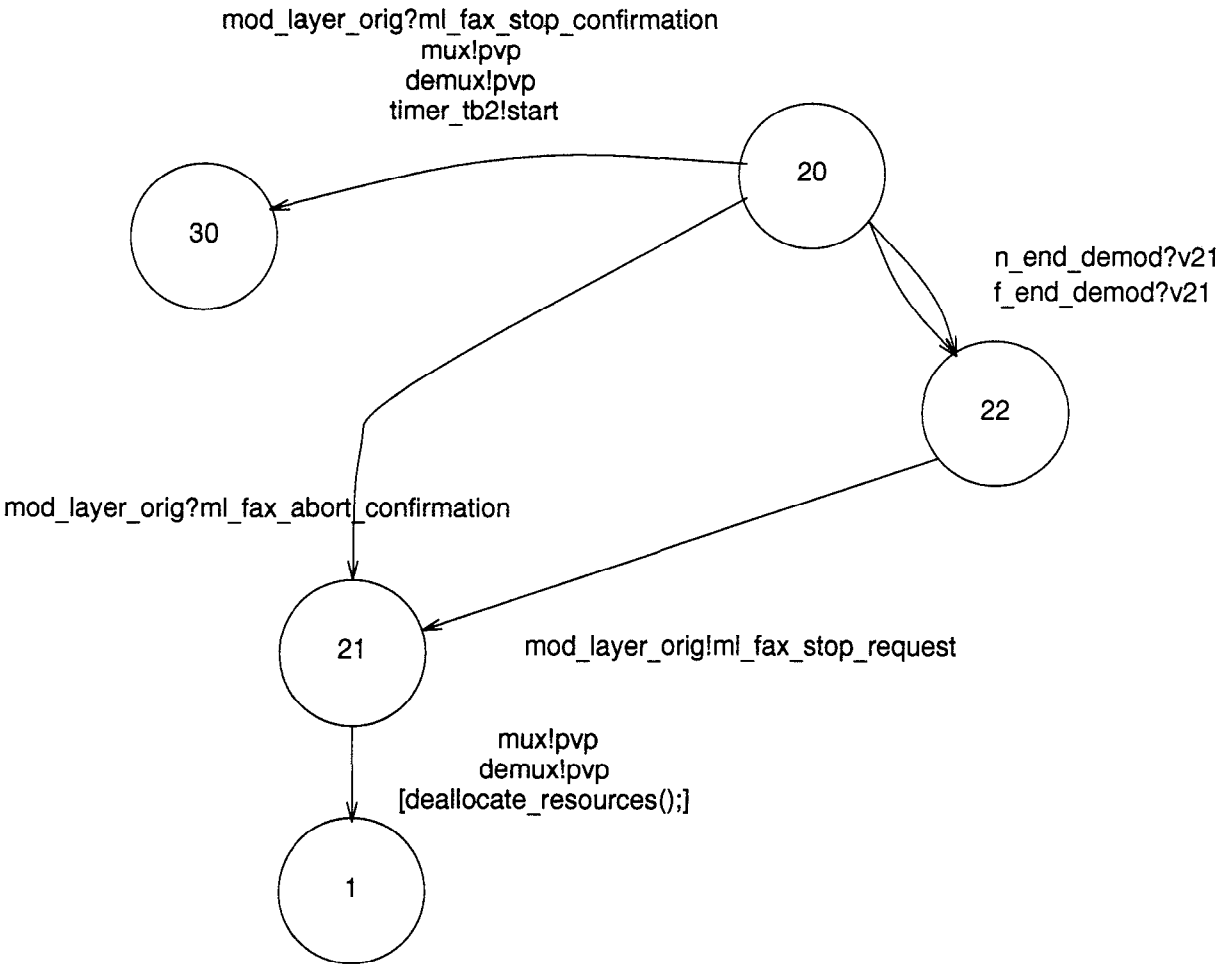


Figure E.3 – call_state: WAIT_FOR_TCF_TO_END state (t.b1)

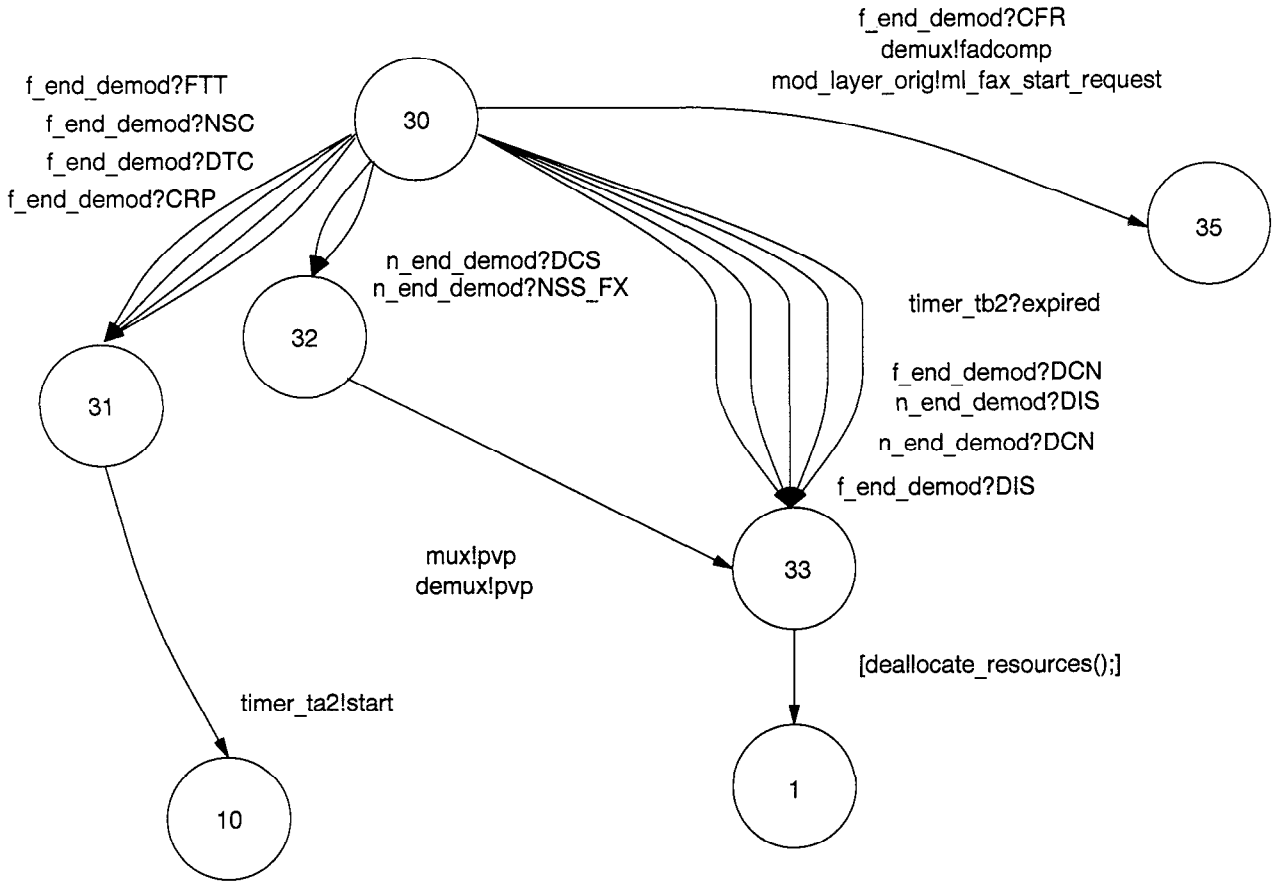


Figure E.4 – call_state: WAIT_FOR_CFR state (t.b2)

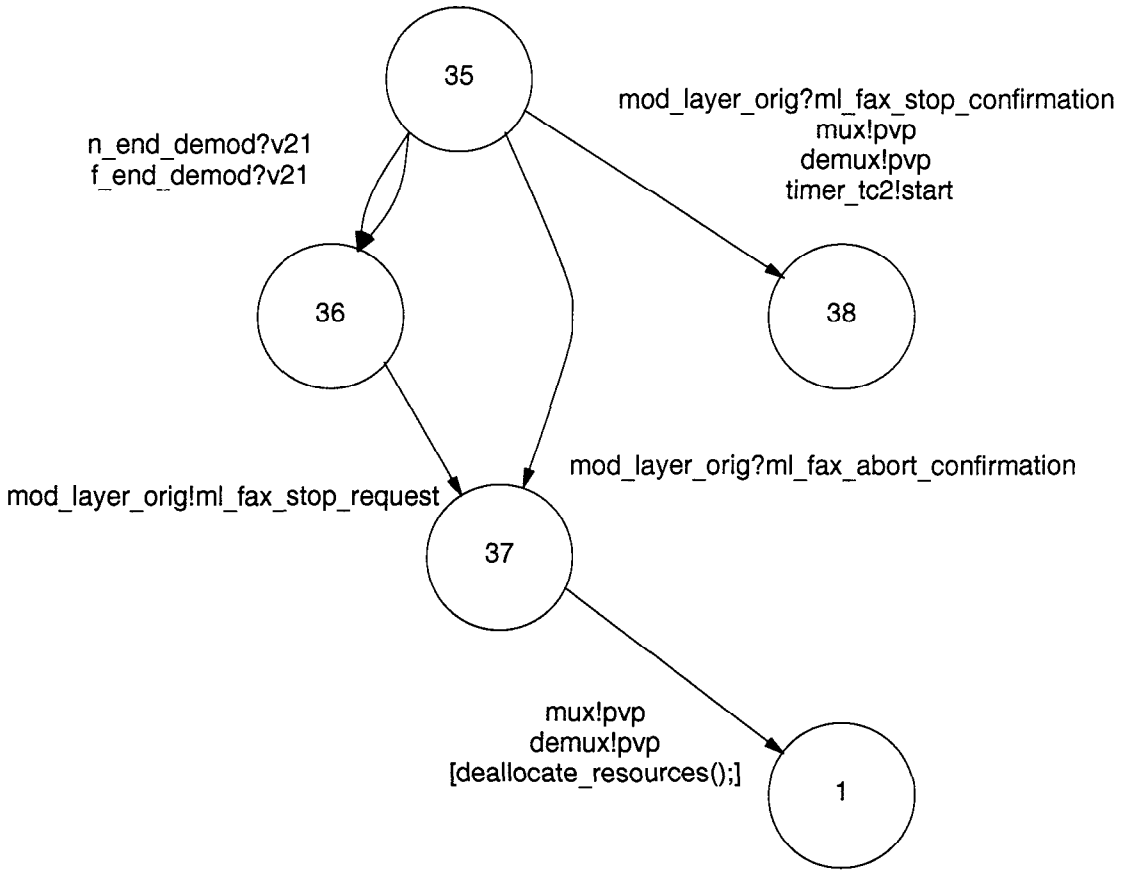


Figure E.5 – call_state: WAIT_FOR_END_OF_PAGE state (t.c1)

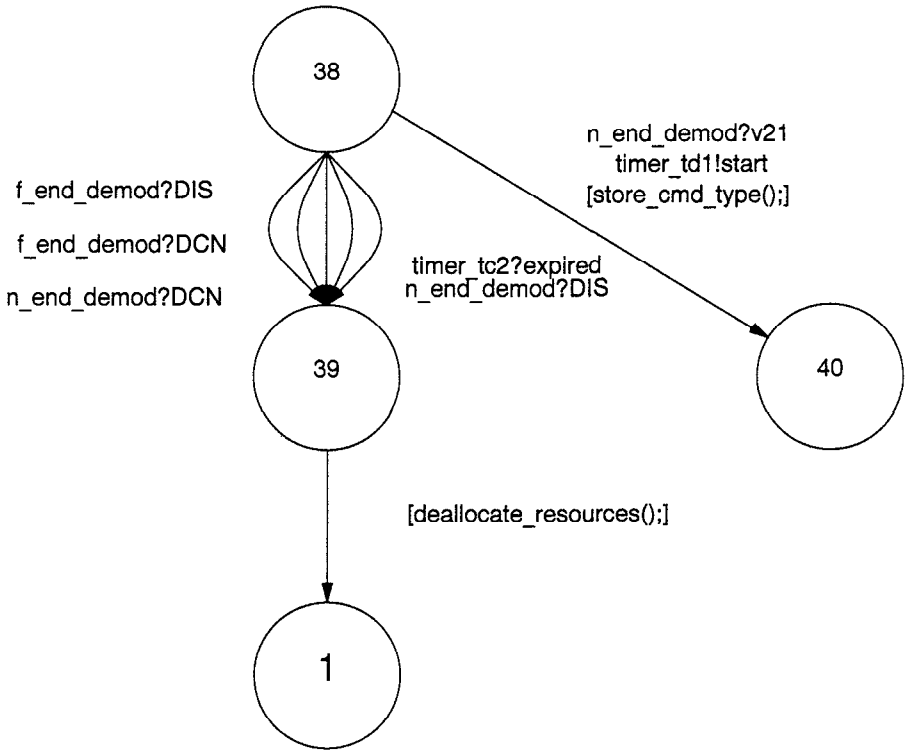


Figure E.6 – call_state: WAIT_FOR_POST_PAGE_CMD state (t.c2)

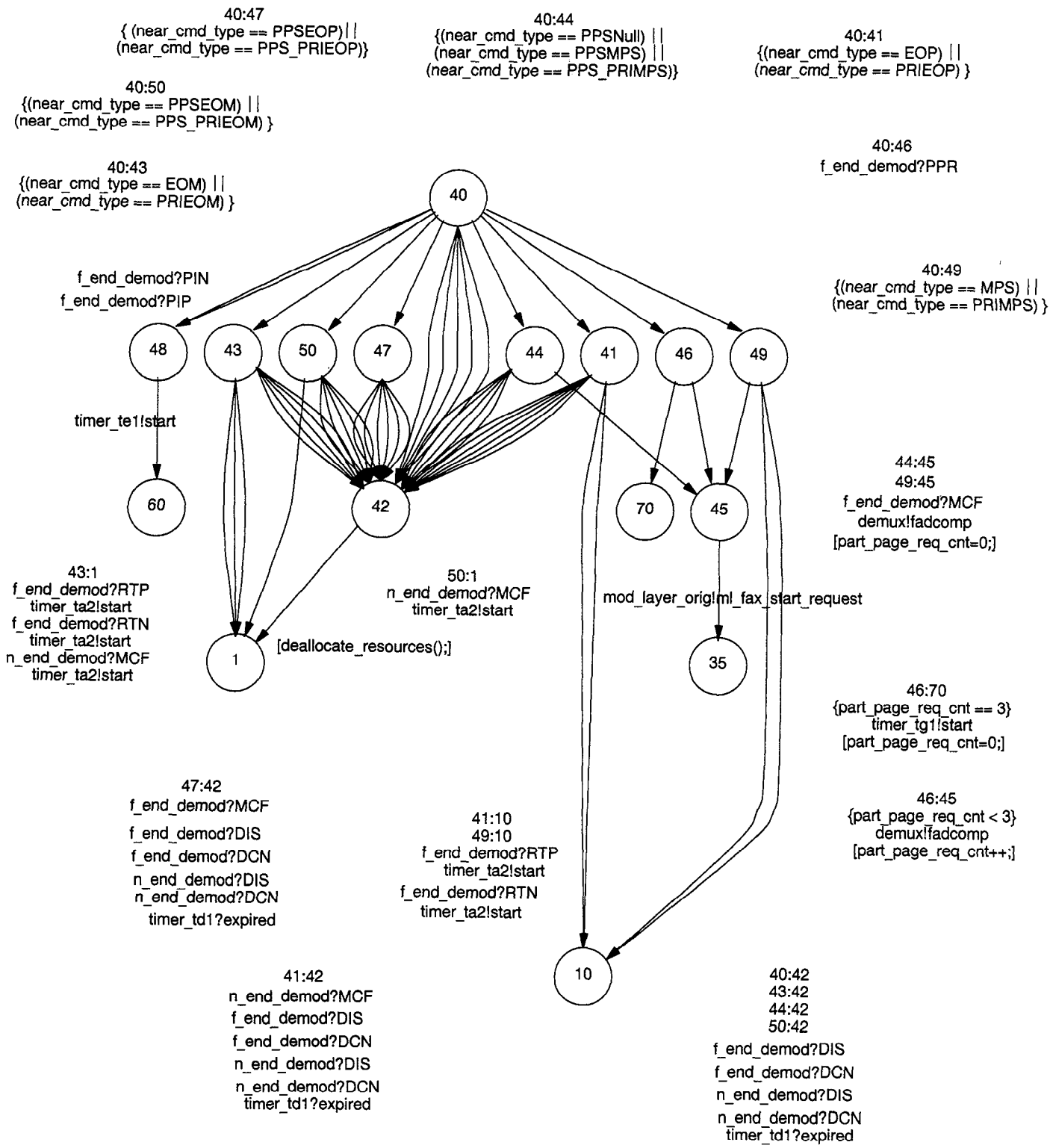


Figure E.7 – call_state: WAIT_FOR_PAGE_PAGE_RESPONSE state (t.d1)

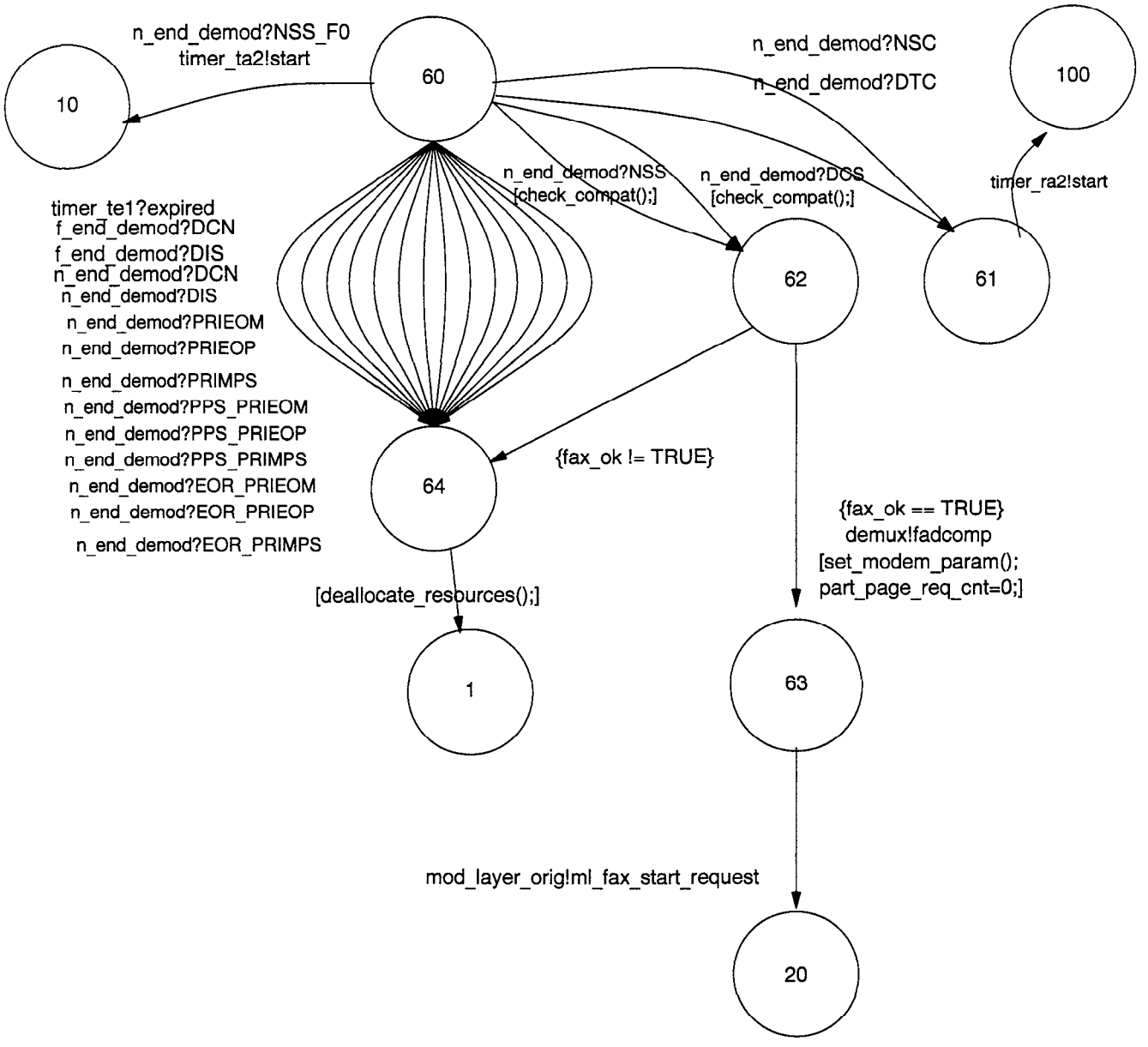


Figure E.8 – call_state: WAIT_FOR_CMD state (t.e1)

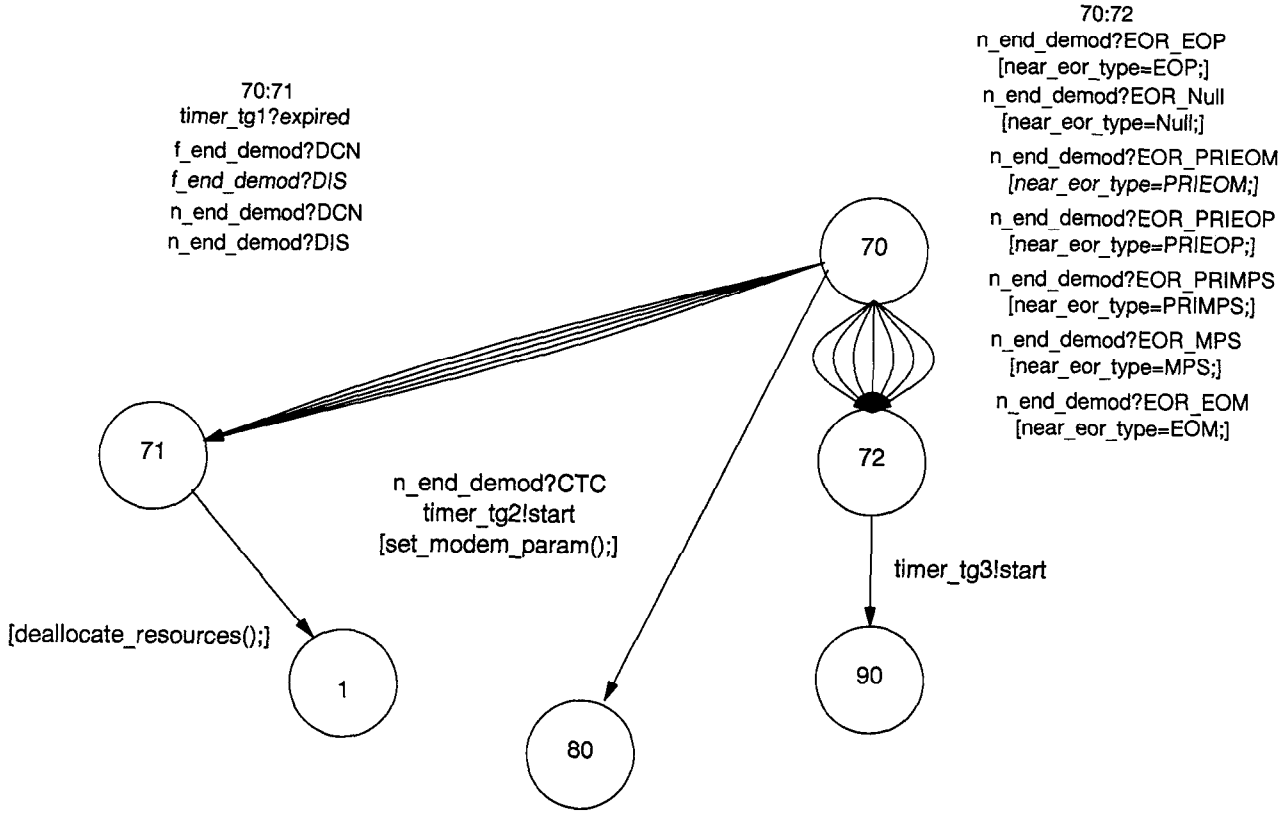


Figure E.9 – call_state: WAIT_FOR_CTC_OR_EOR.X state (t.g1)

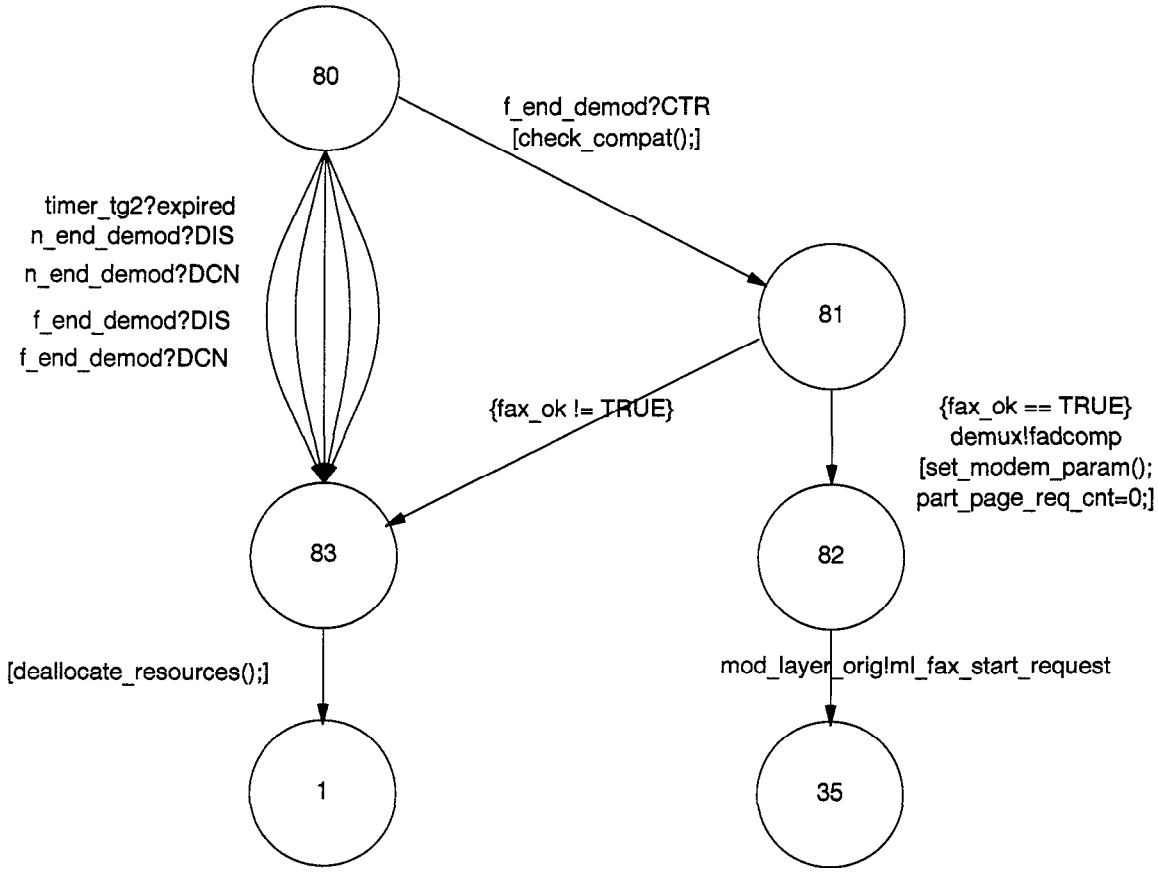


Figure E.10 – call_state: WAIT_FOR_CTR state (t.g2)

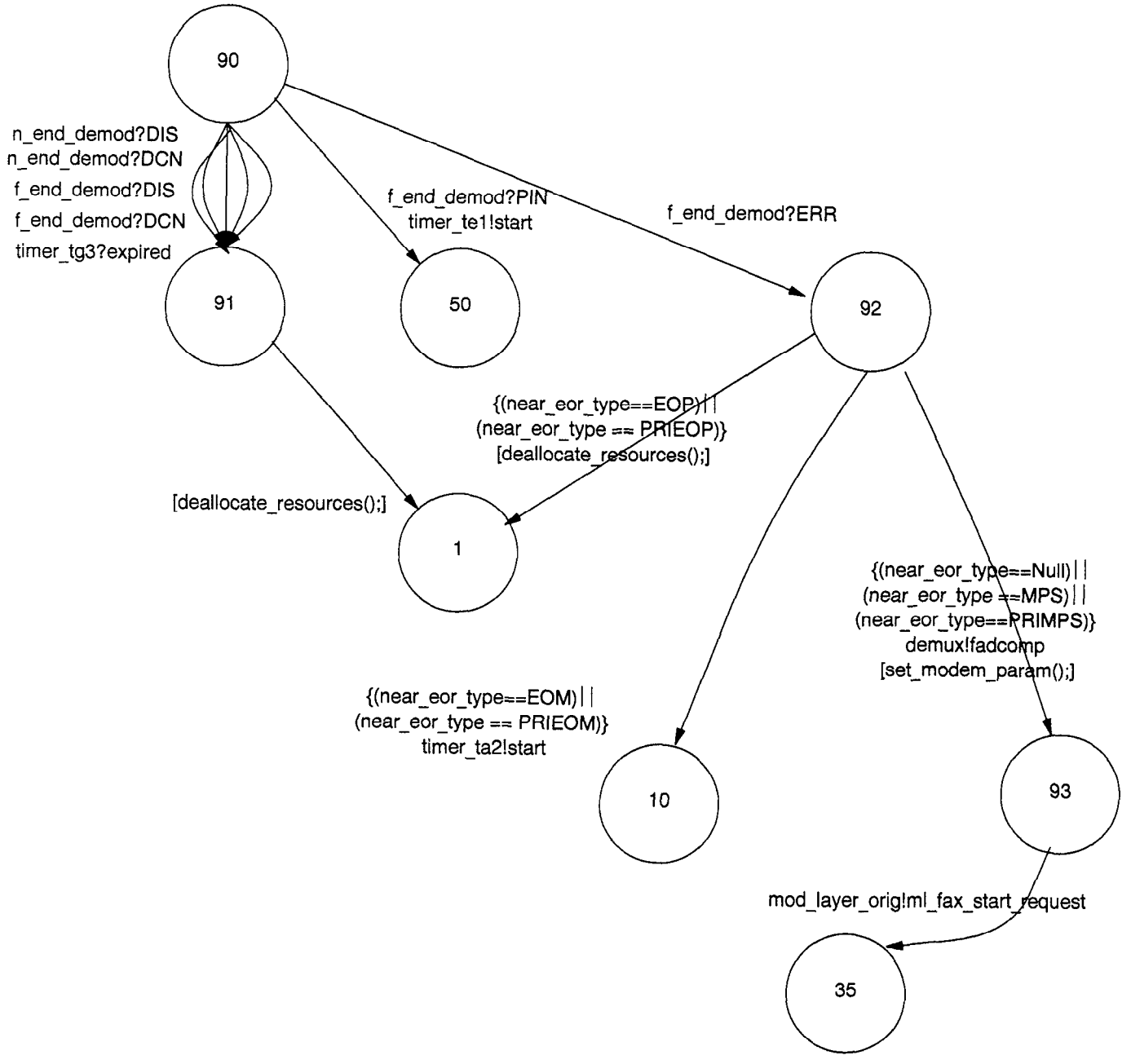


Figure E.11 – call_state: WAIT_FOR_ERR state (t.g3)

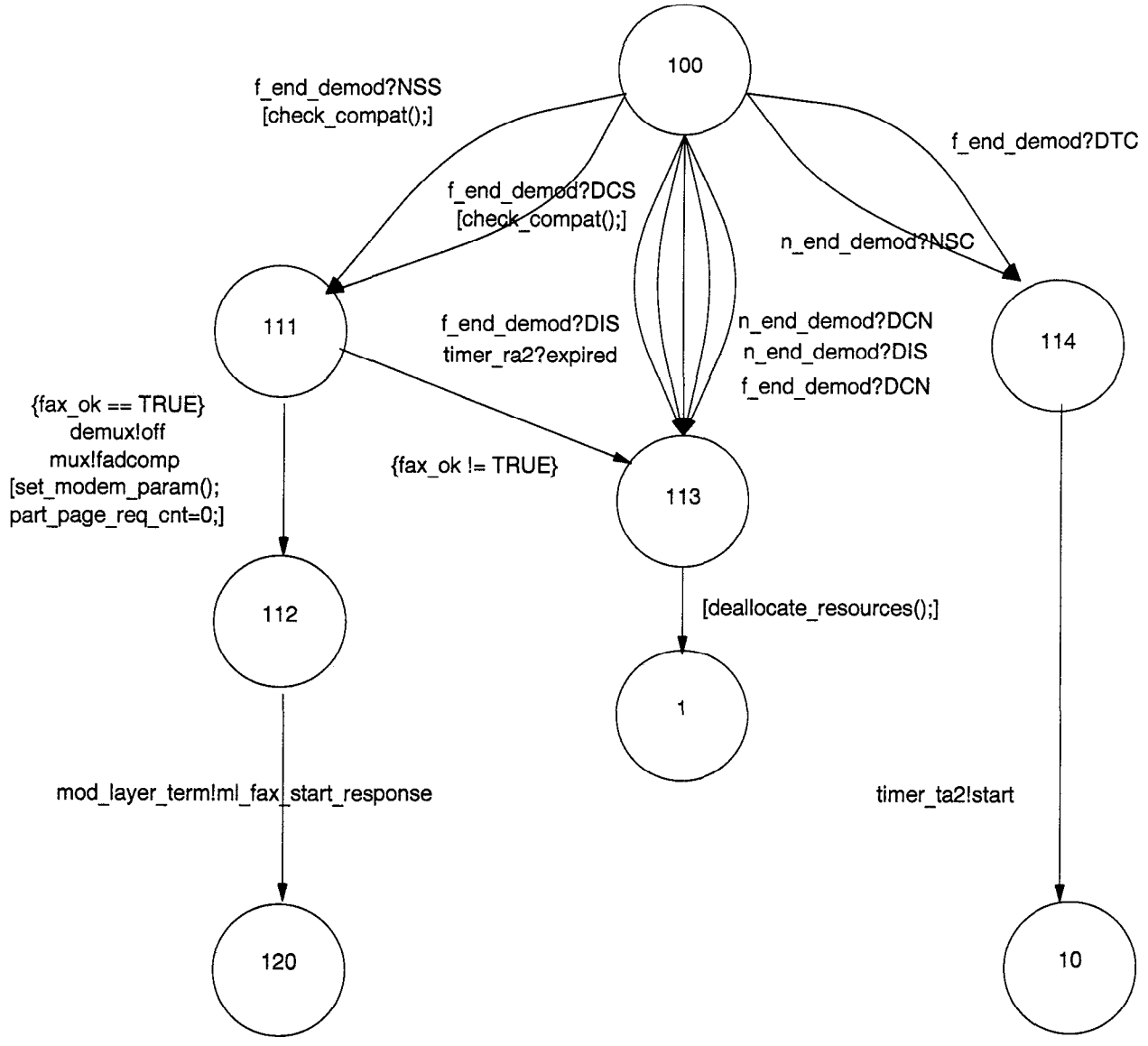


Figure E.12 – call_state: WAIT_FOR_DCS_FROM_FAR_END state (r.a2)

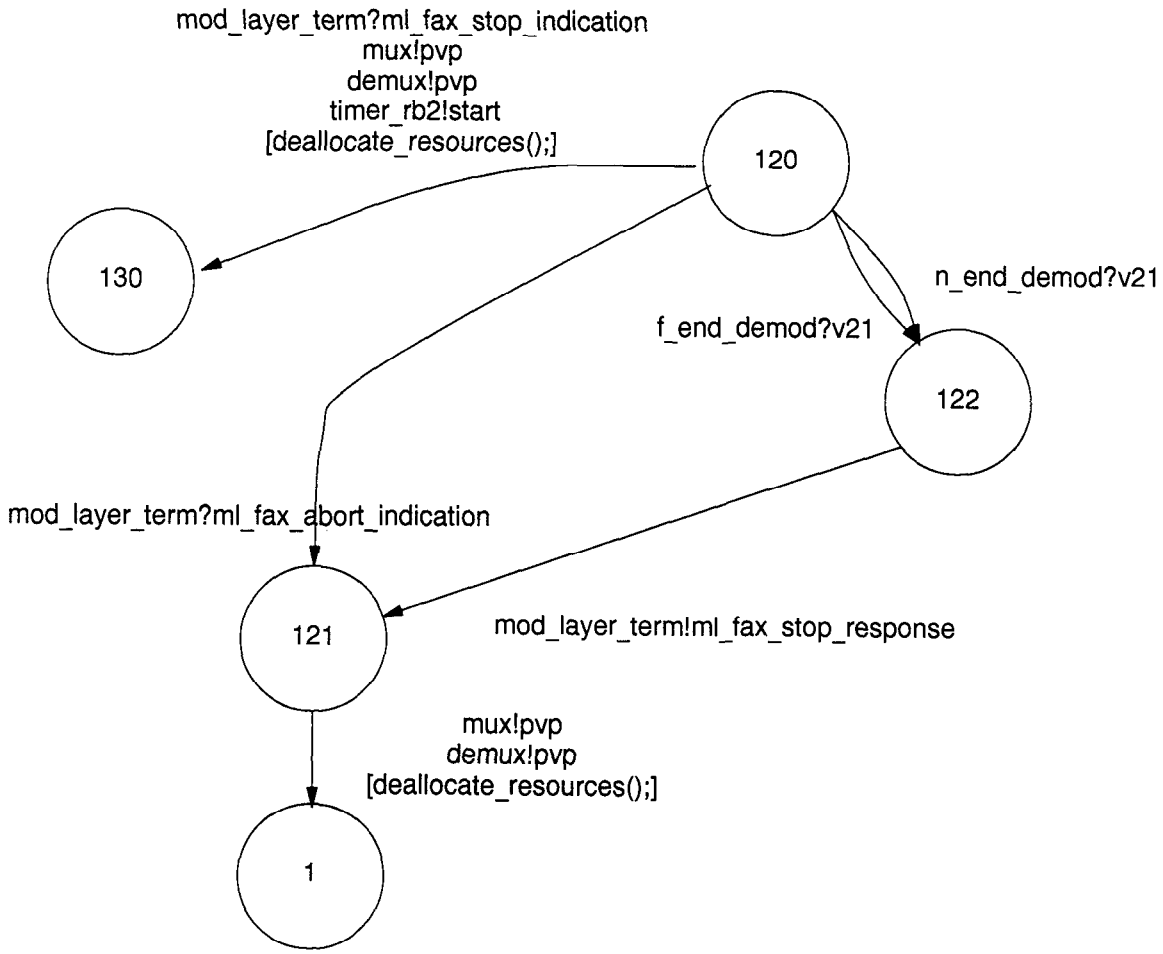


Figure E.13 – call_state: WAIT_FOR_TCF_TO_END state (r.b1)

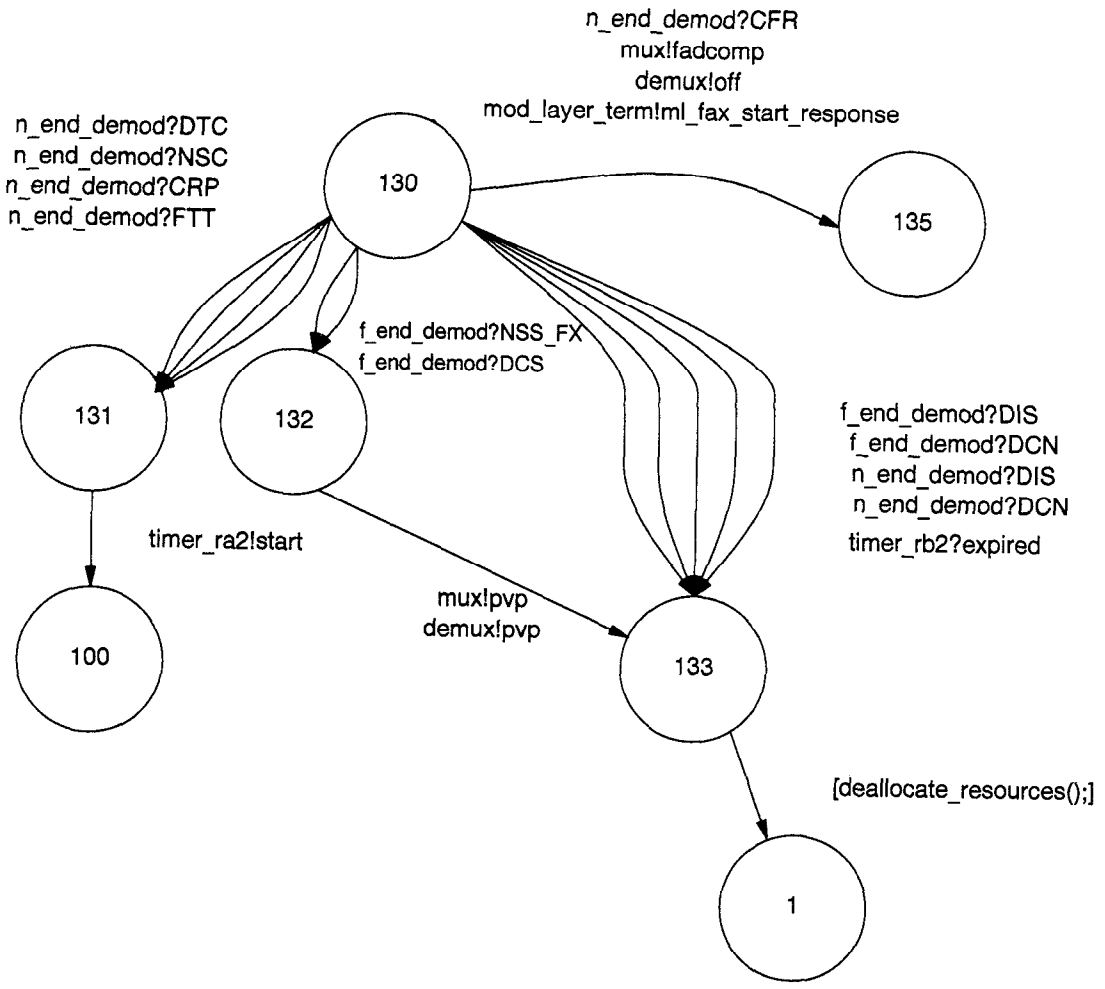


Figure E.14 – call_state: WAIT_FOR_CFR state (r.b2)

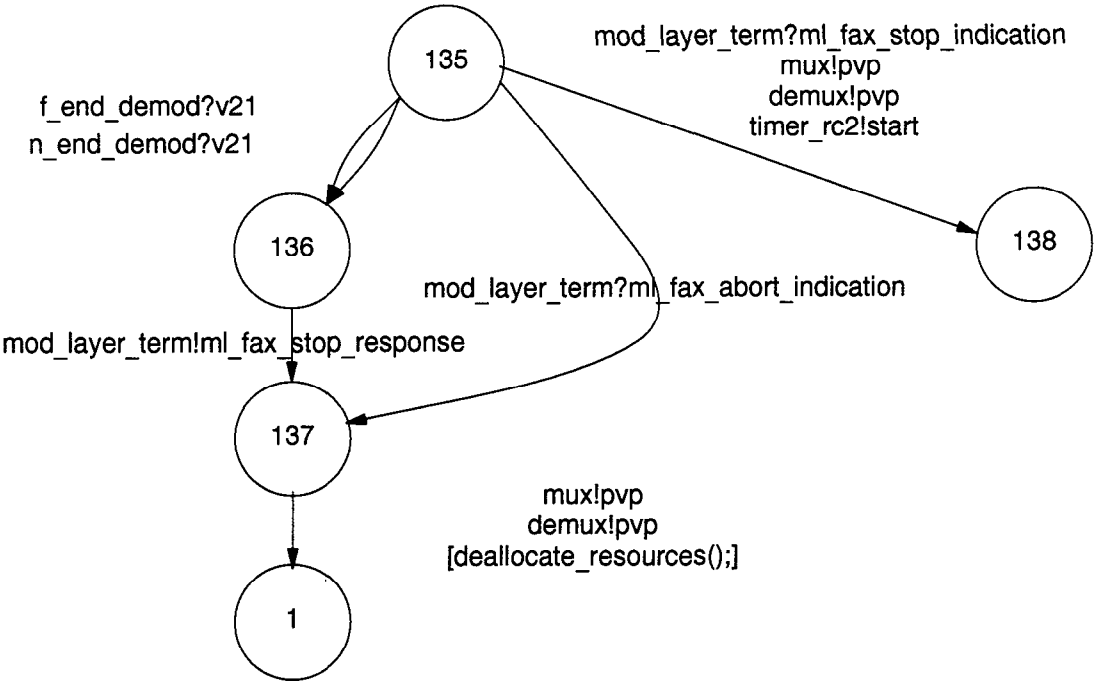


Figure E.15 – call_state: WAIT_FOR_END_OF_PAGE state (r.c1)

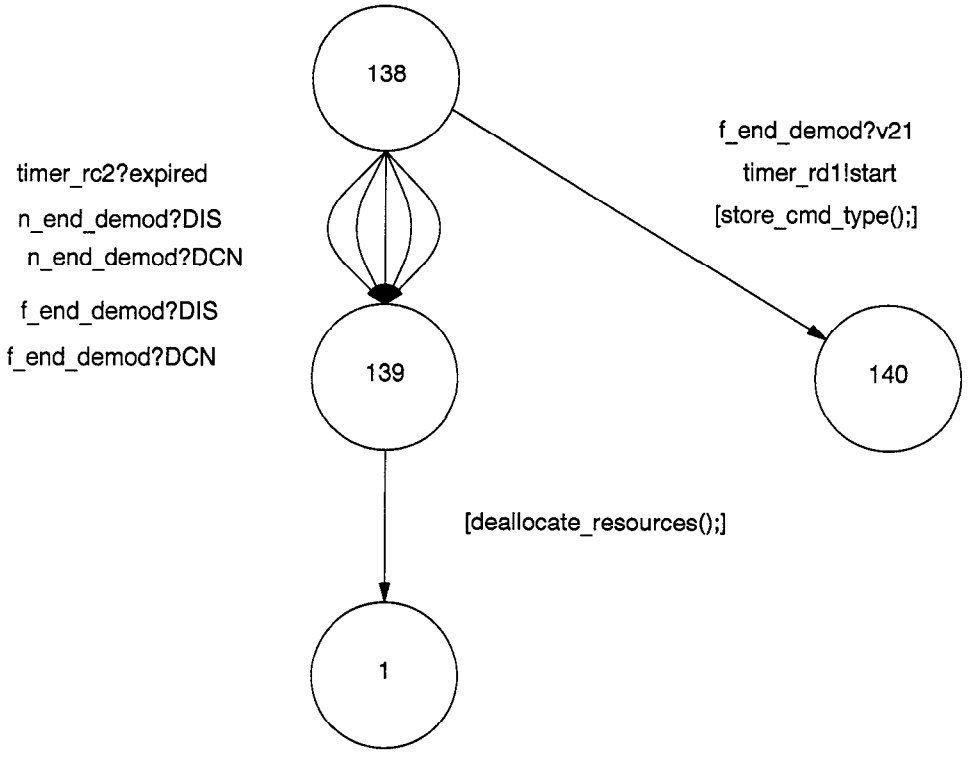


Figure E.16 – call_state: WAIT_FOR_POST_PAGE_CMD state (r.c2)

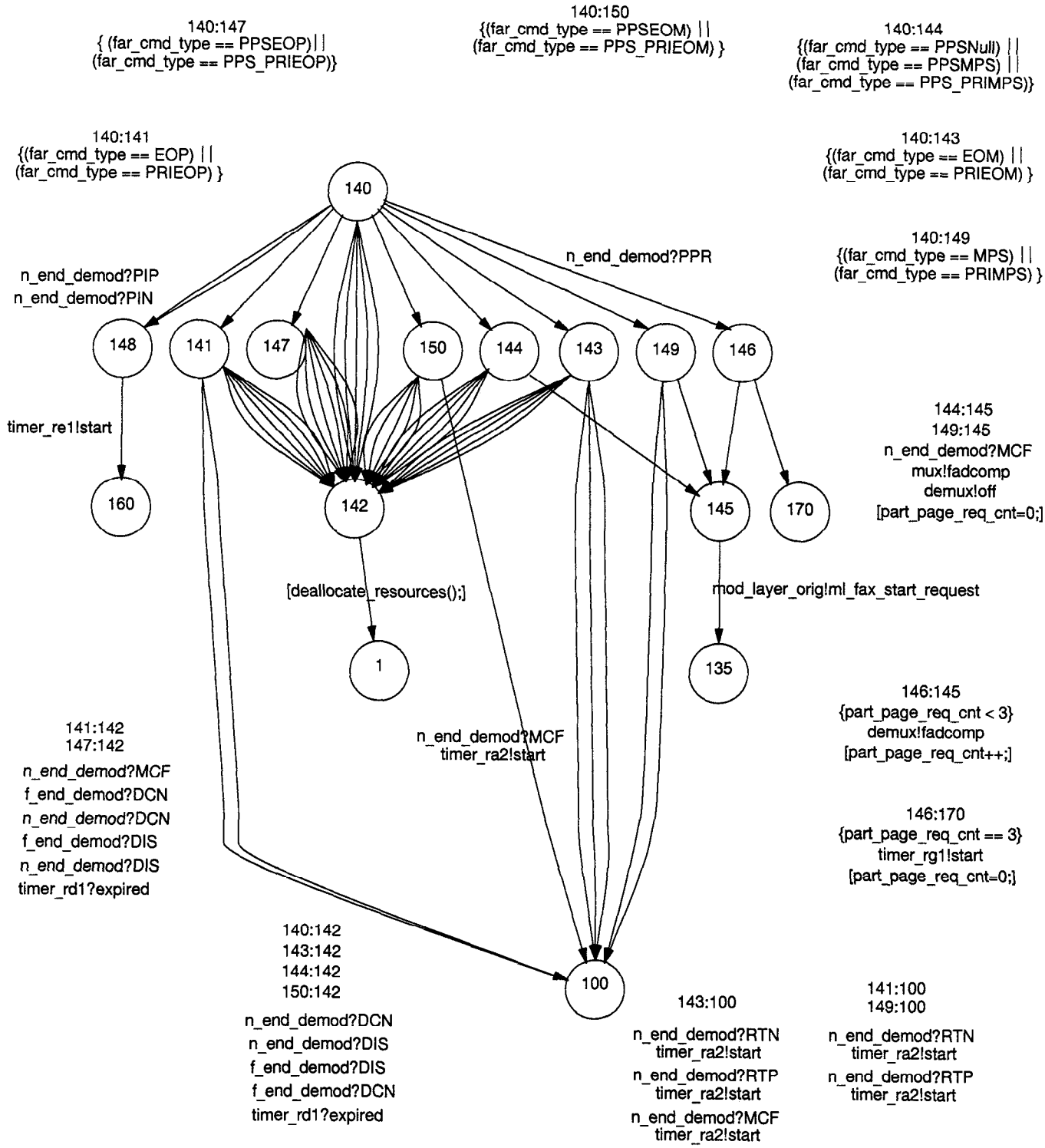


Figure E.17 – call_state: WAIT_FOR_POST_PAGE_RESPONSE state (r.d1)

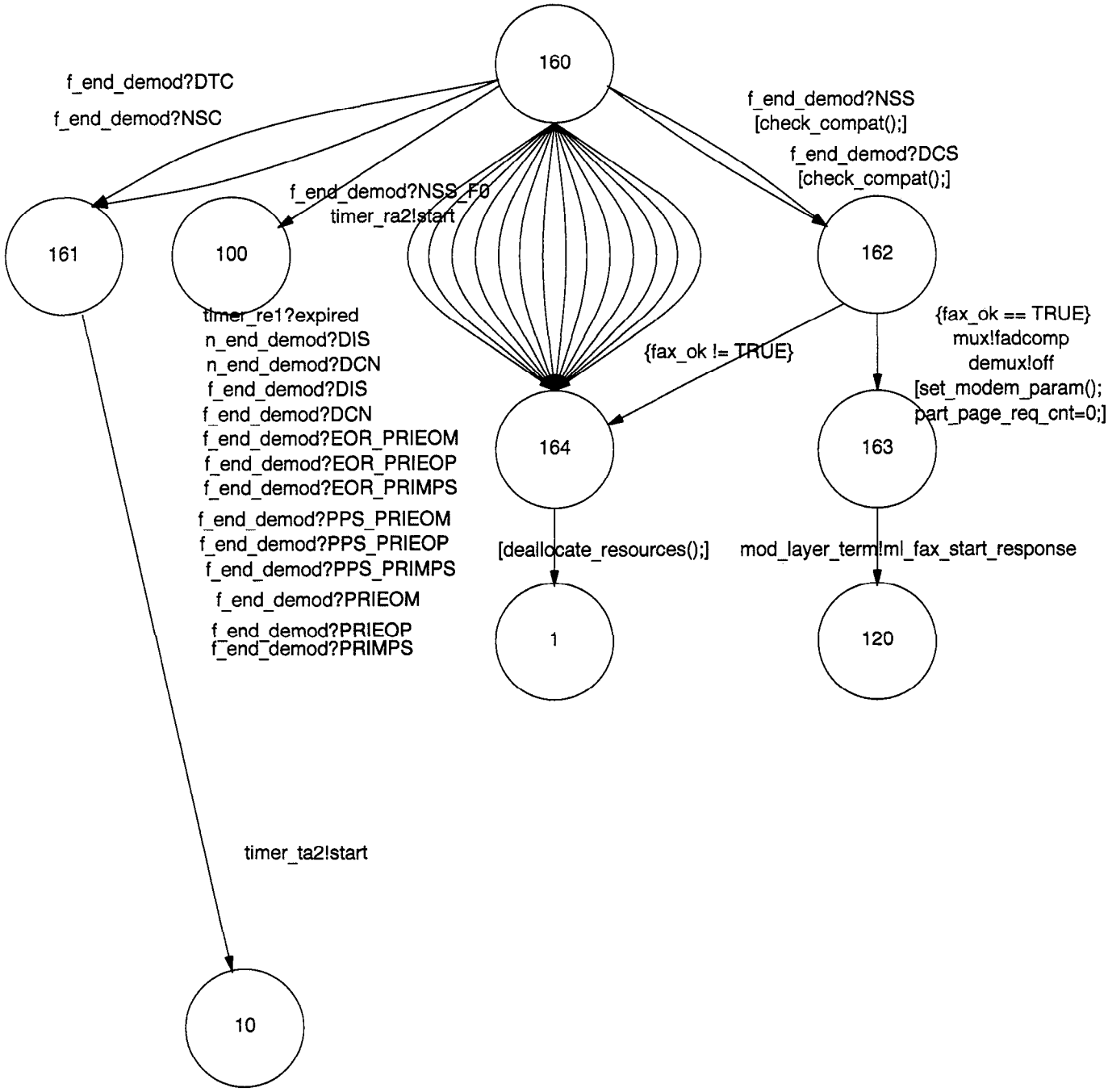


Figure E.18 – call_state: WAIT_FOR_CMD state (r.e1)

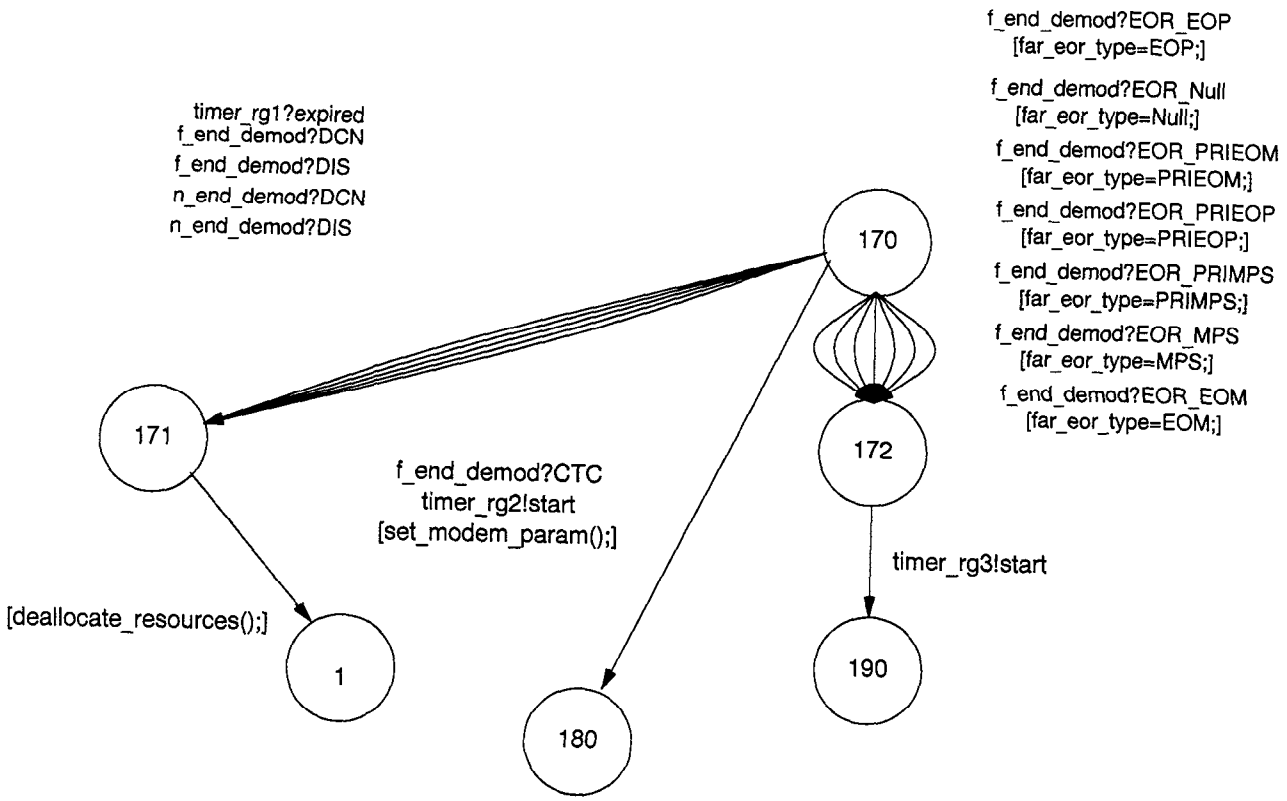


Figure E.19 – call_state: WAIT_FOR_CTC_OR_EOR.X state (r.g1)

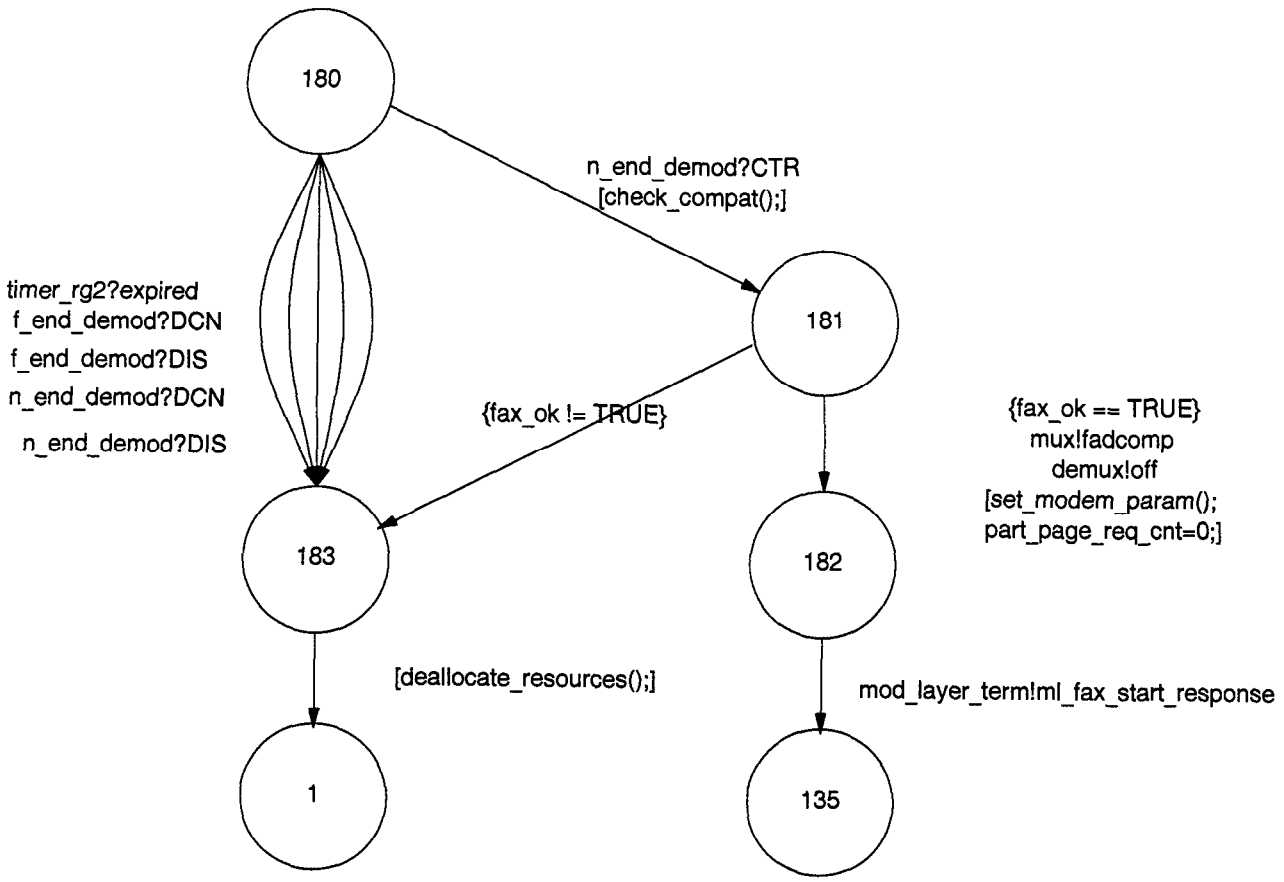


Figure E.20 – call_state: WAIT_FOR_CTR state (r.g2)

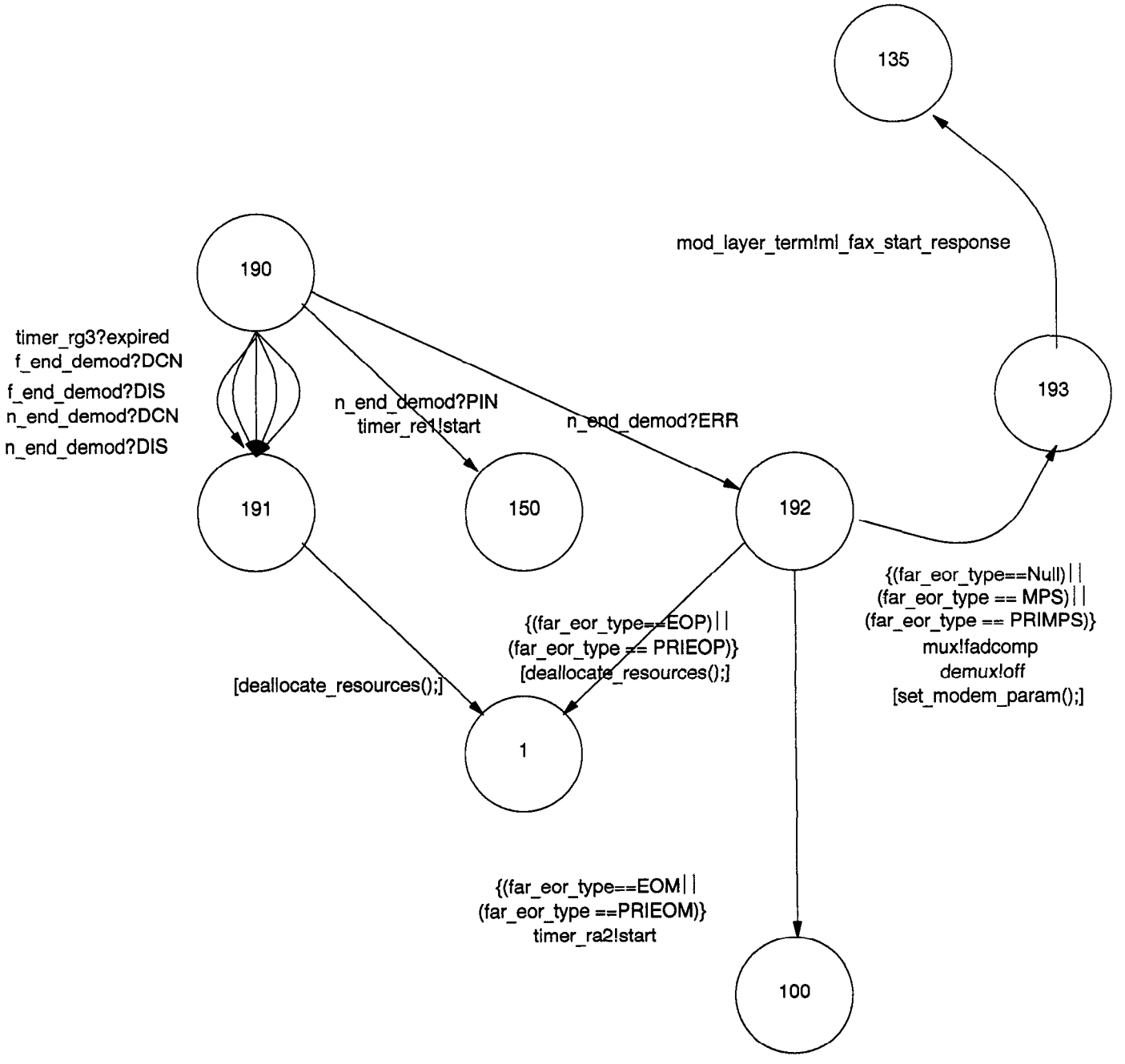


Figure E.21 – call_state: WAIT_FOR_ERR state (r.g3)

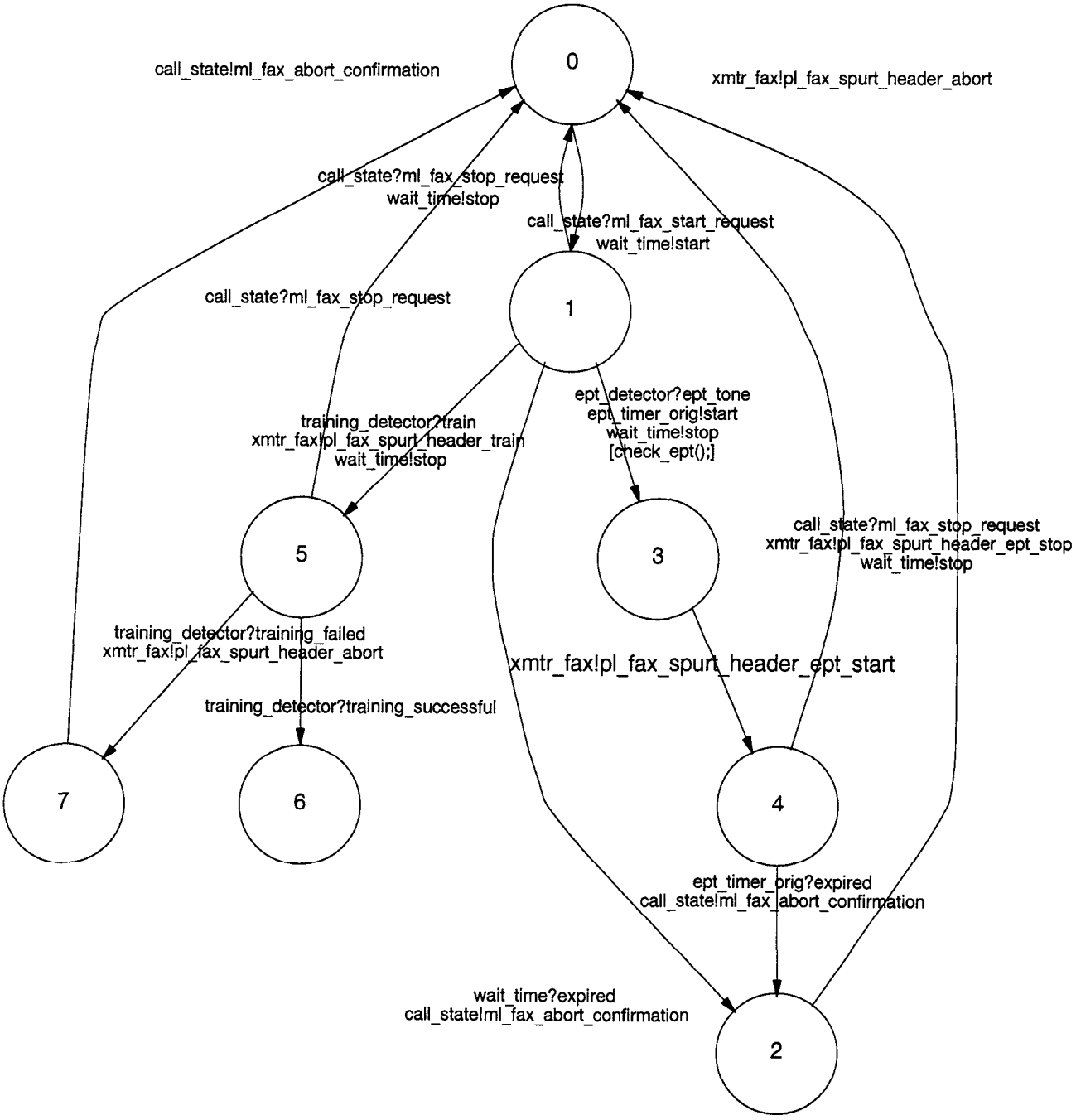


Figure E.22 – mod_layer_orig
Part a

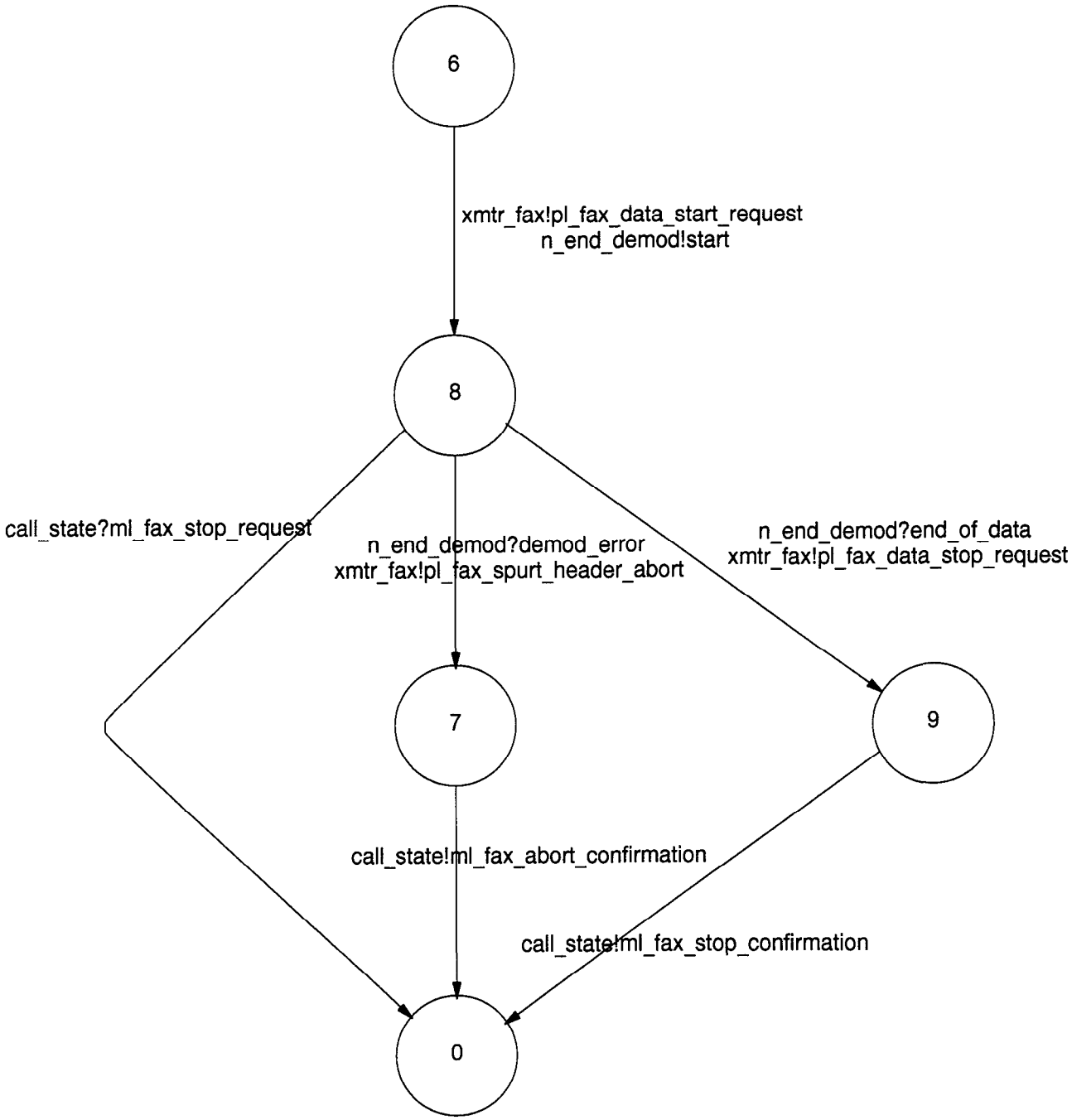


Figure E.22 (concluded) – mod_layer_orig
Part b

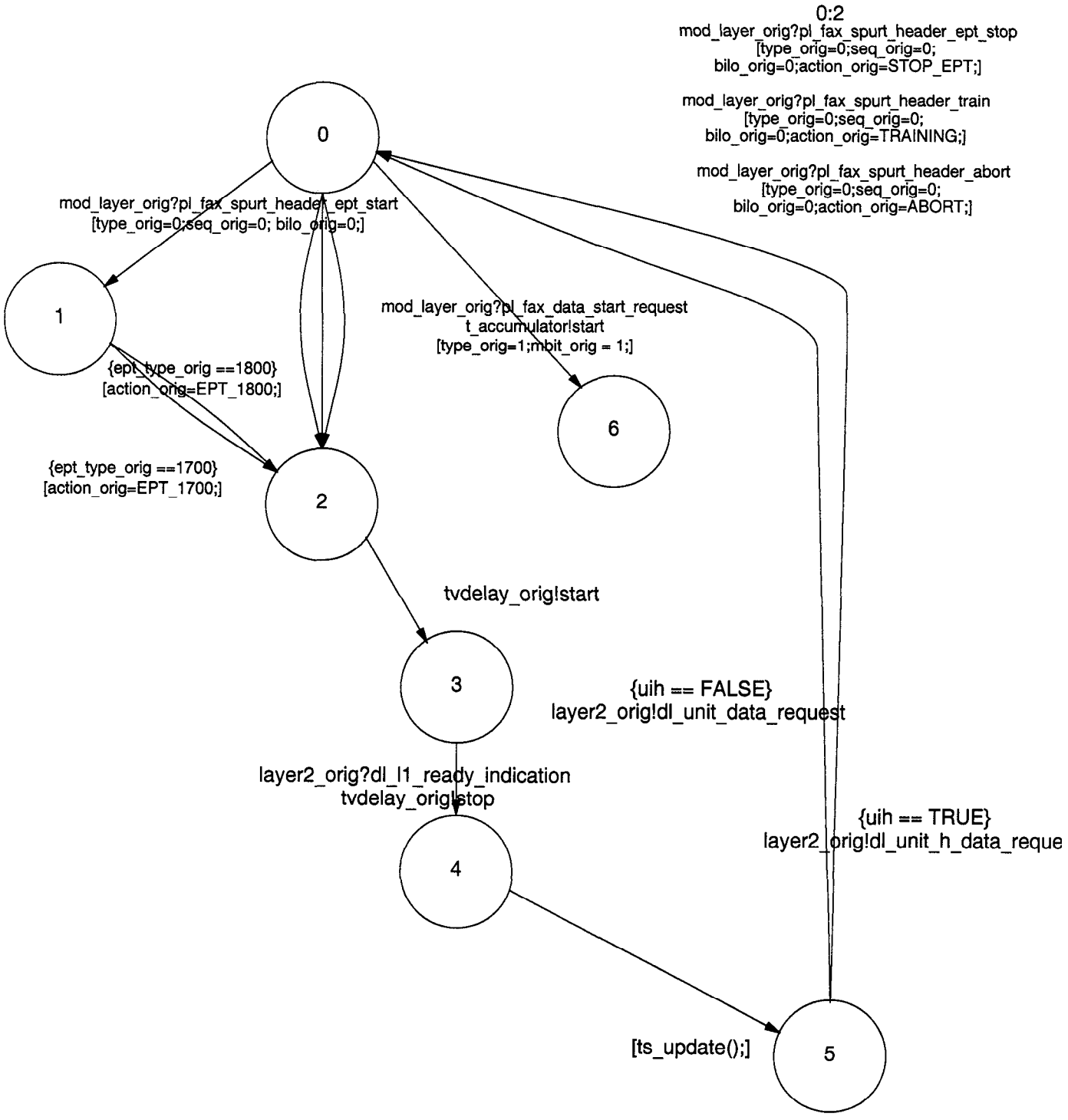


Figure E.23 – xmtr_fax
Part a

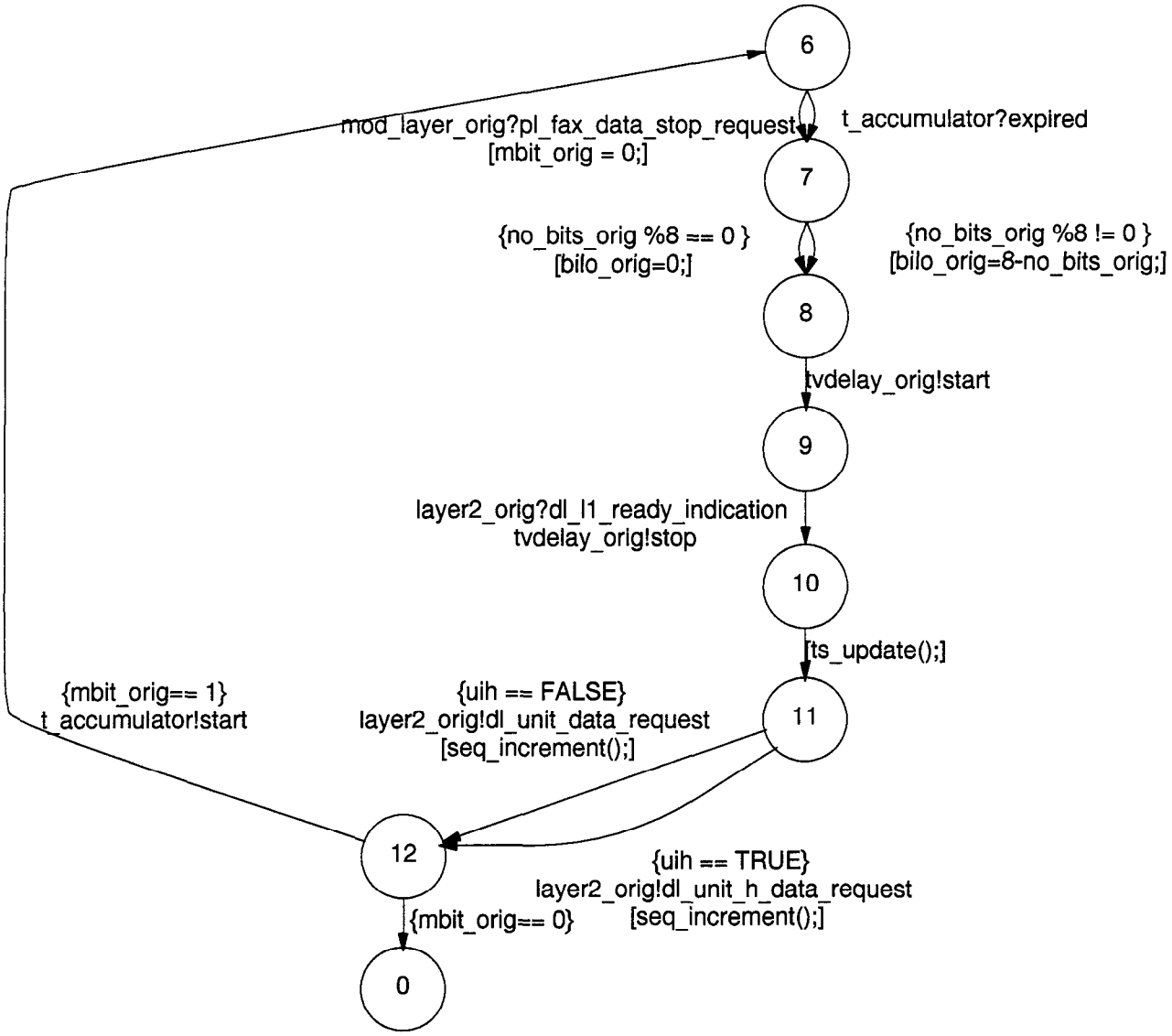


Figure E.23 (concluded) – xmtr_fax
Part b

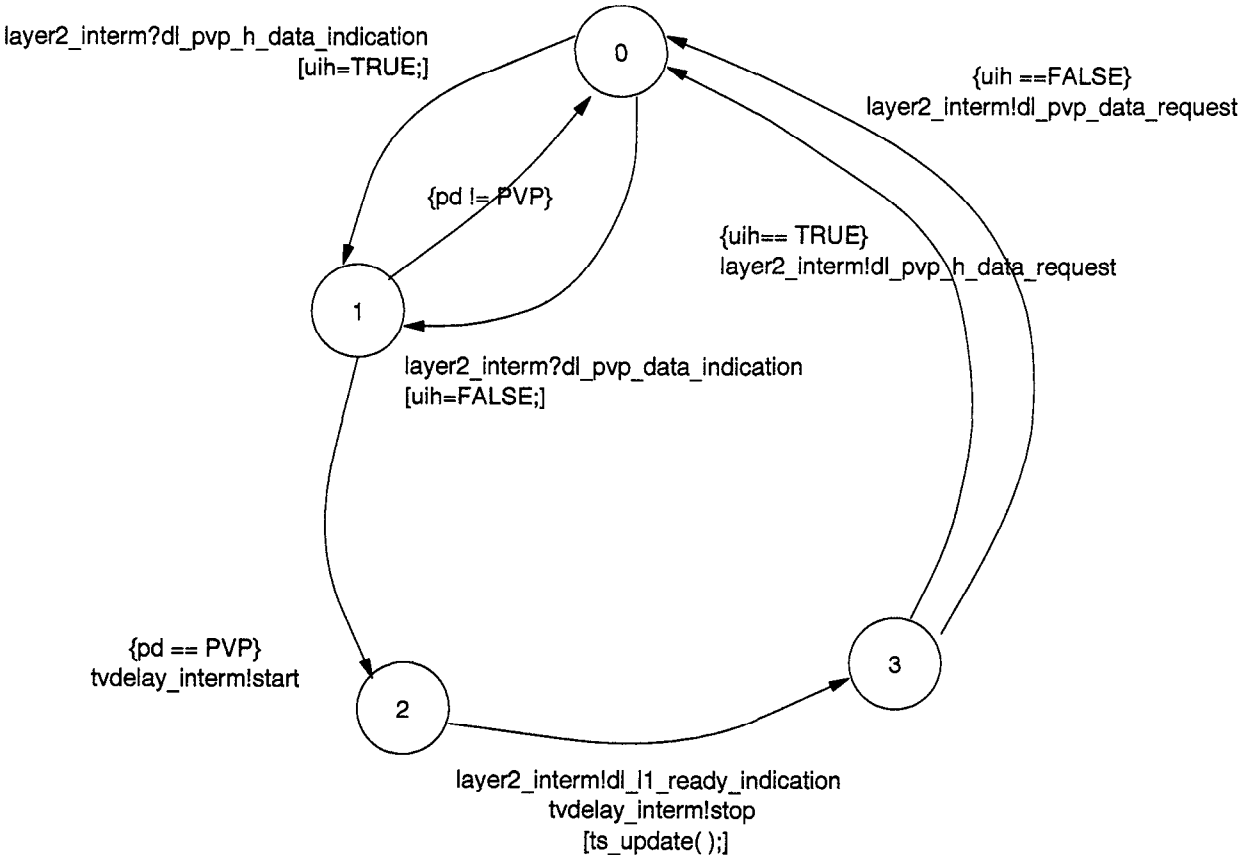


Figure E.24 – interm_pt_fax_relay

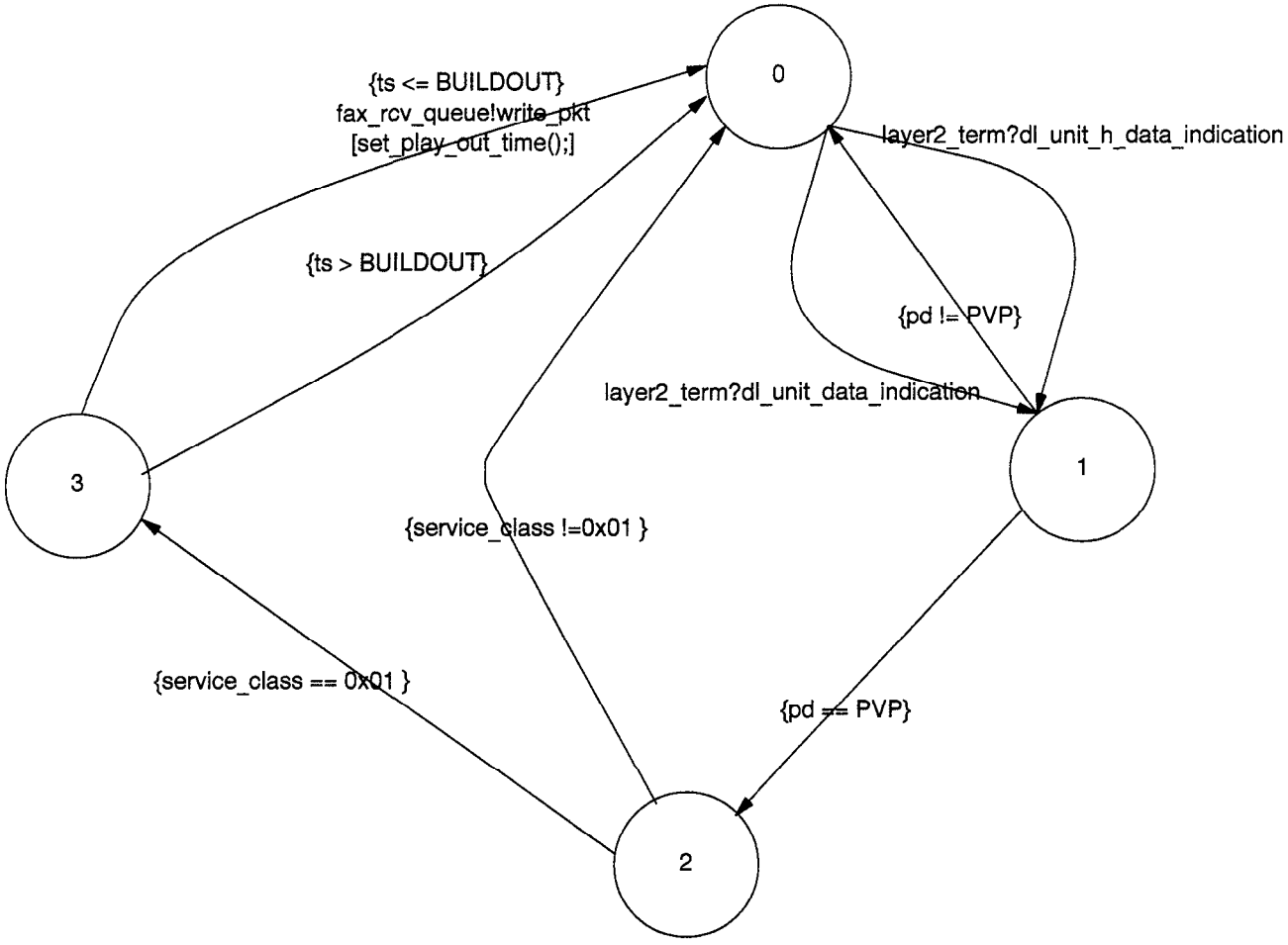


Figure E.25 – rcv_fax

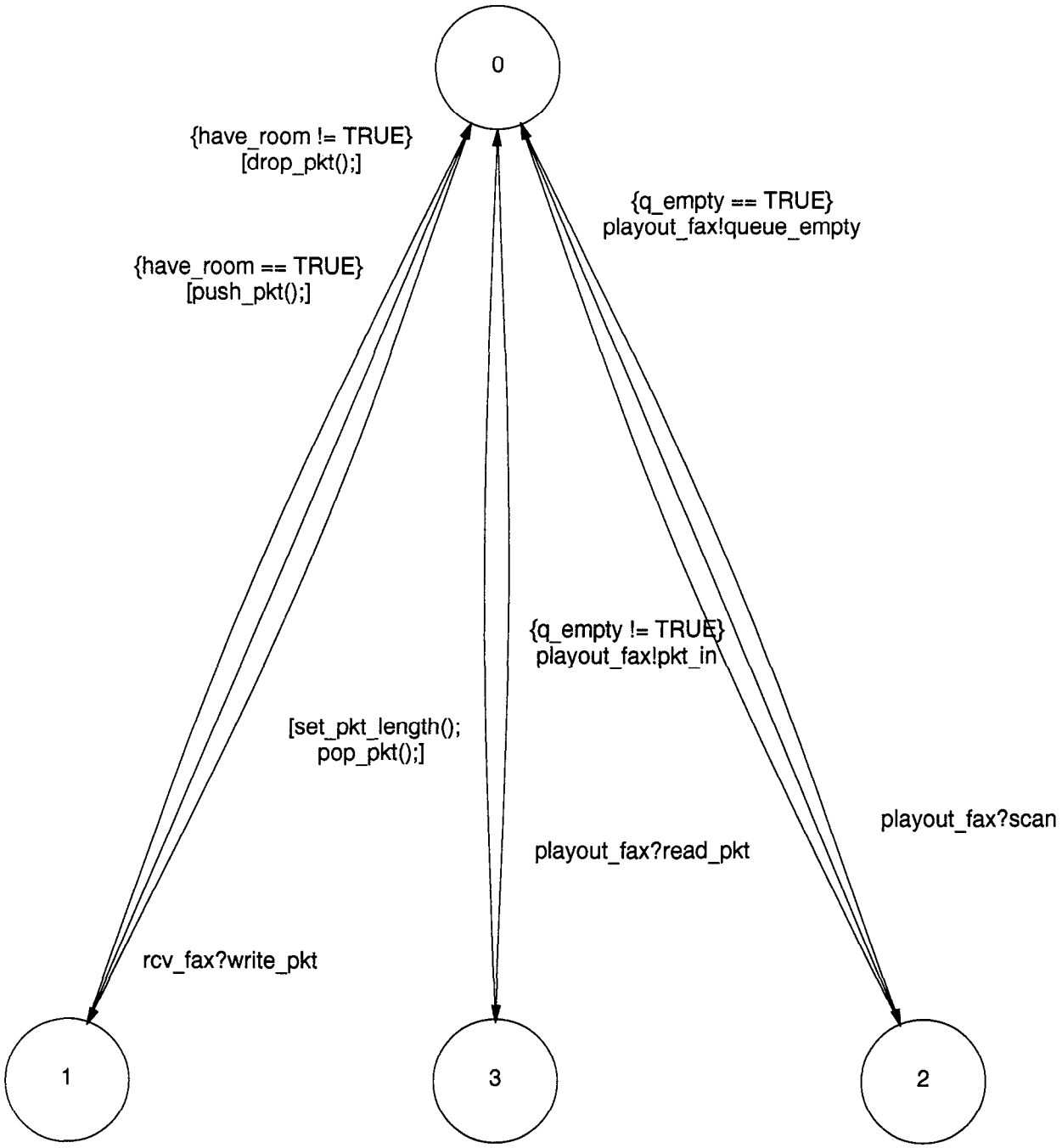


Figure E.26 – fax_rcv_queue

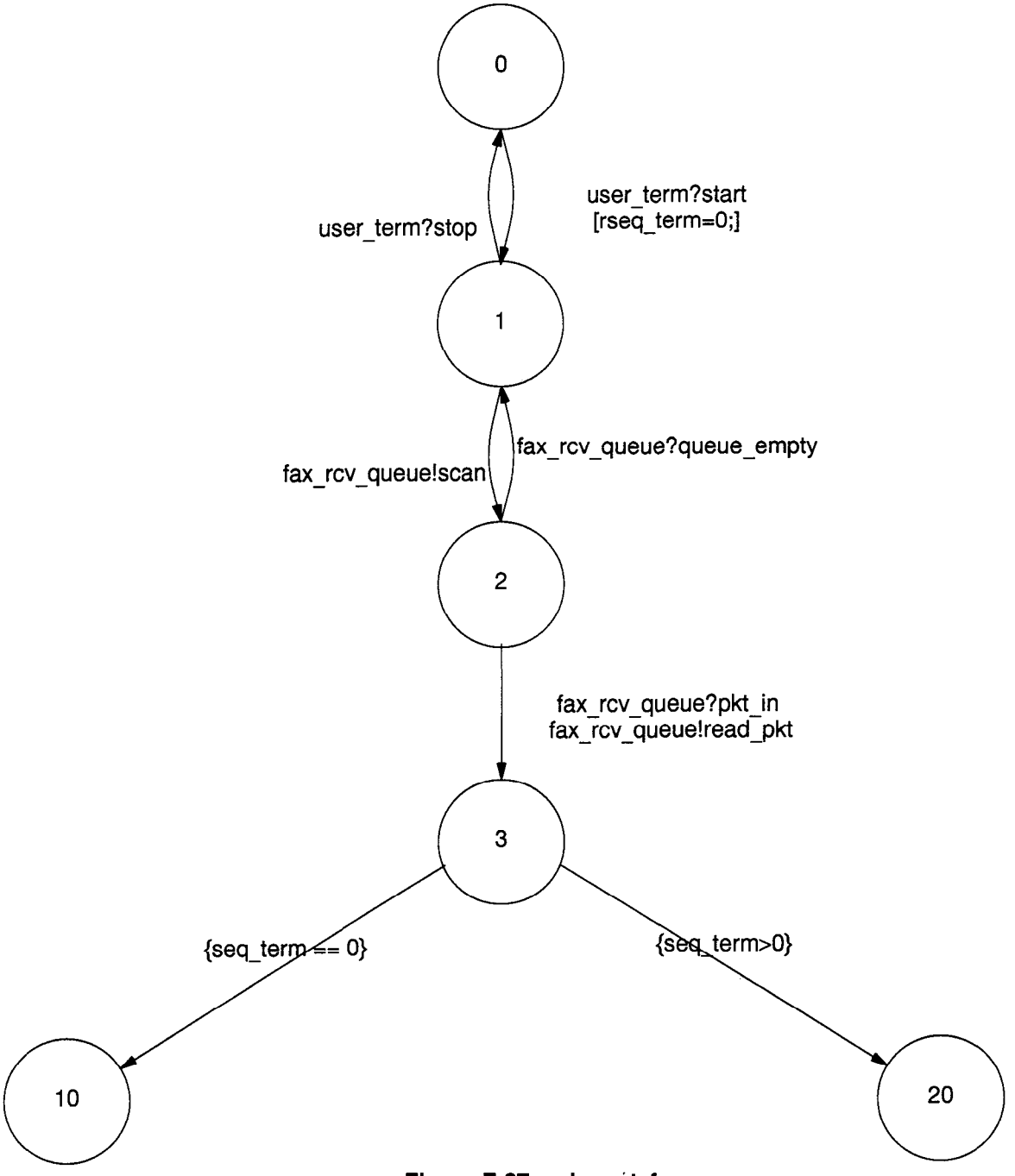


Figure E.27 – playout_fax
Part a

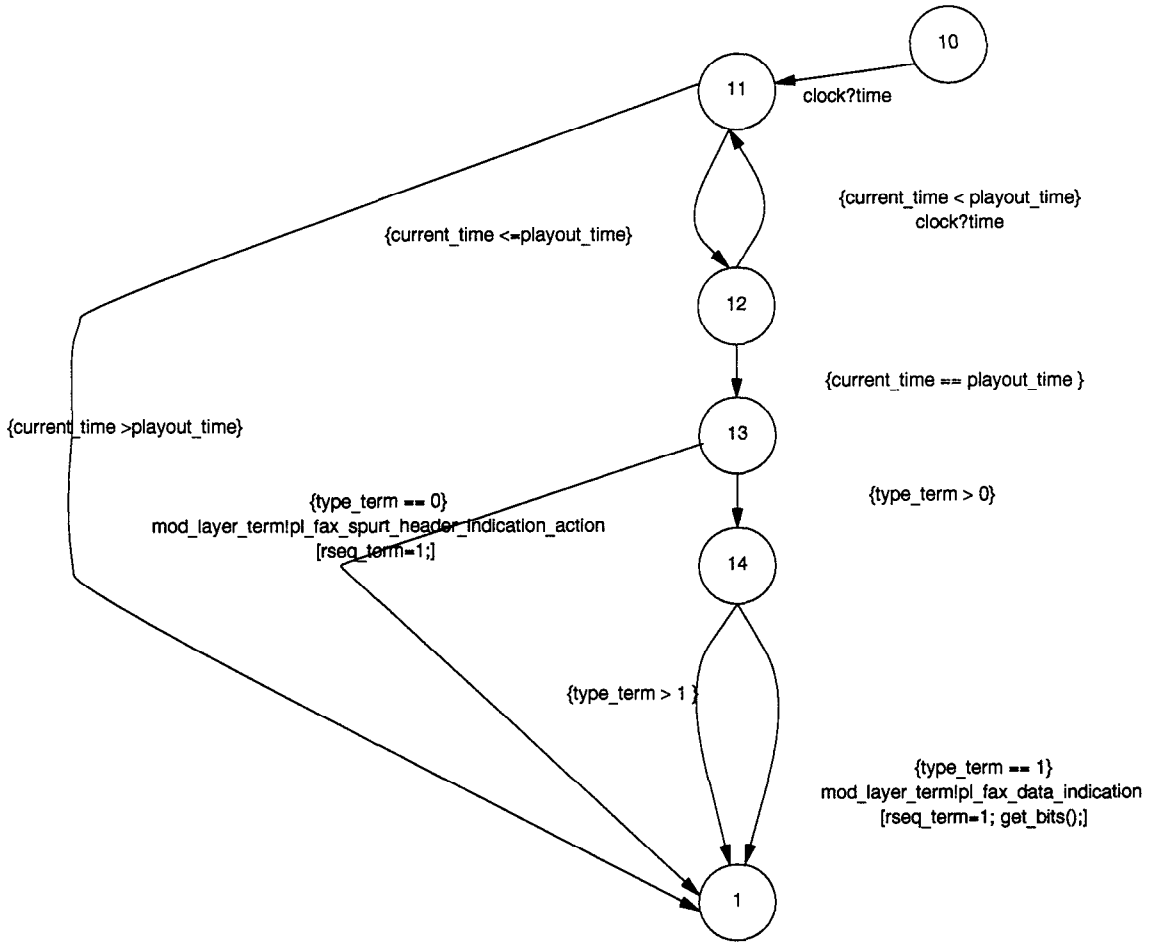


Figure E.27 (continued) – playout_fax
Part b

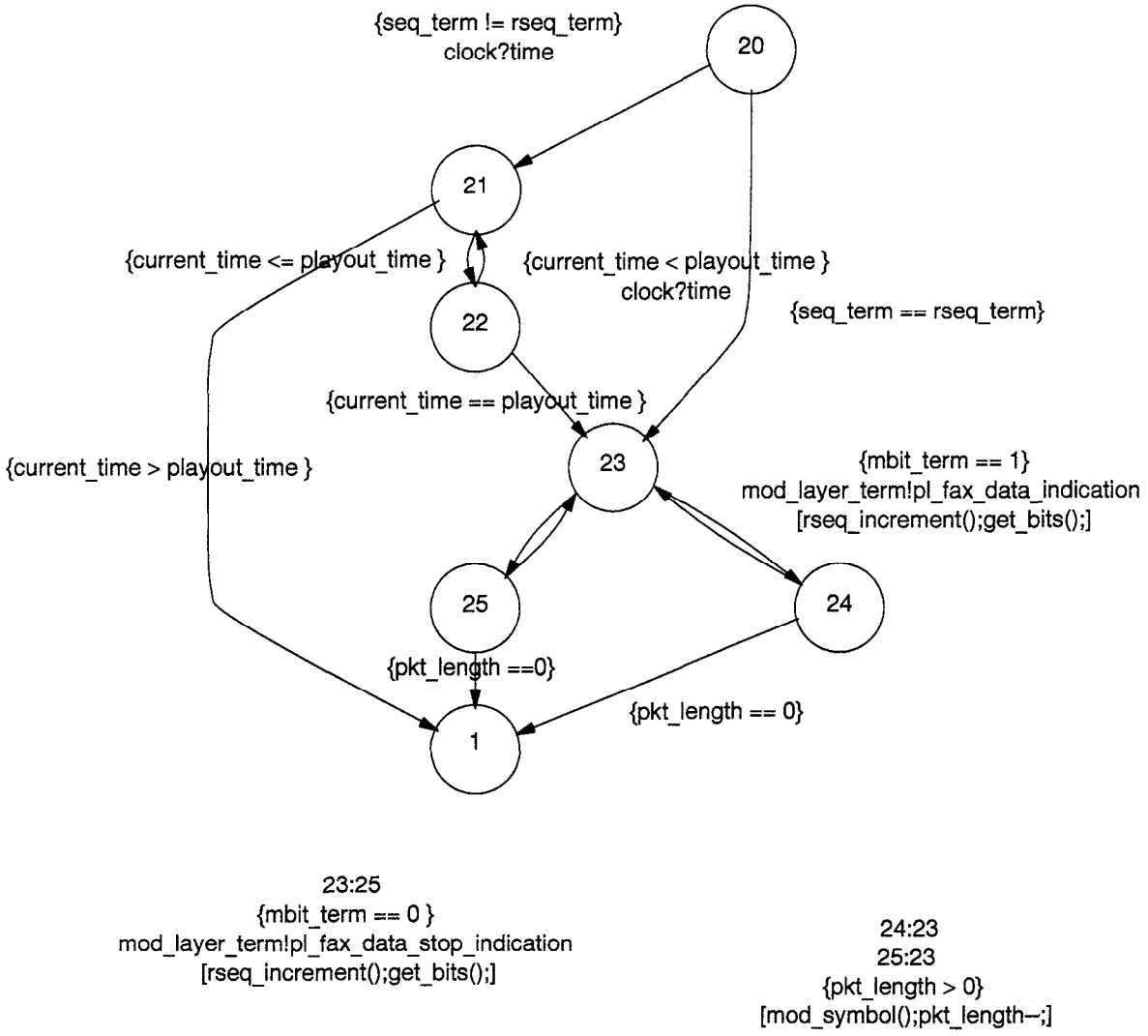


Figure E.27 (concluded) – playout_fax
Part c

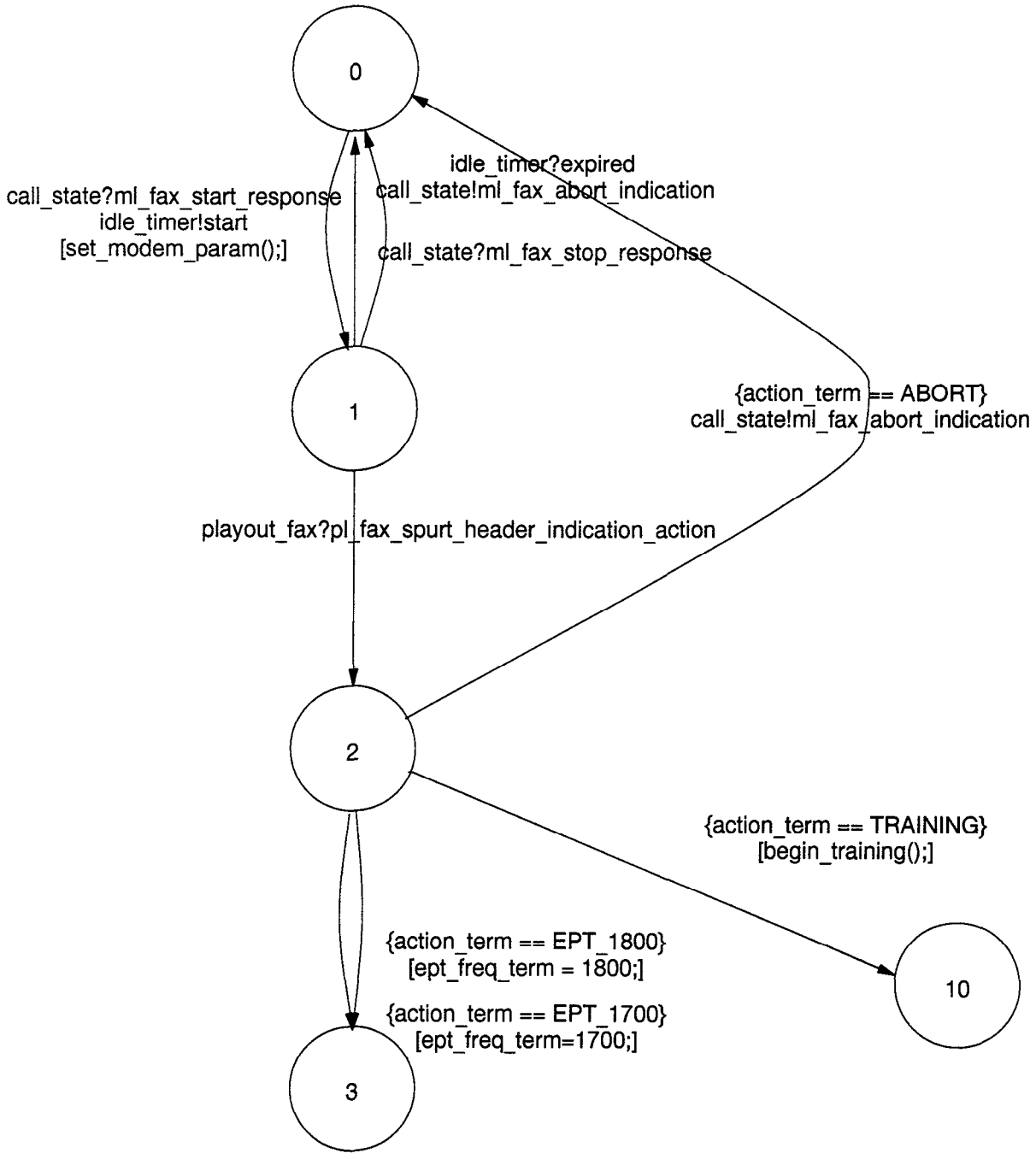


Figure E.28 – mod_layer_term: OFF and IDLE states

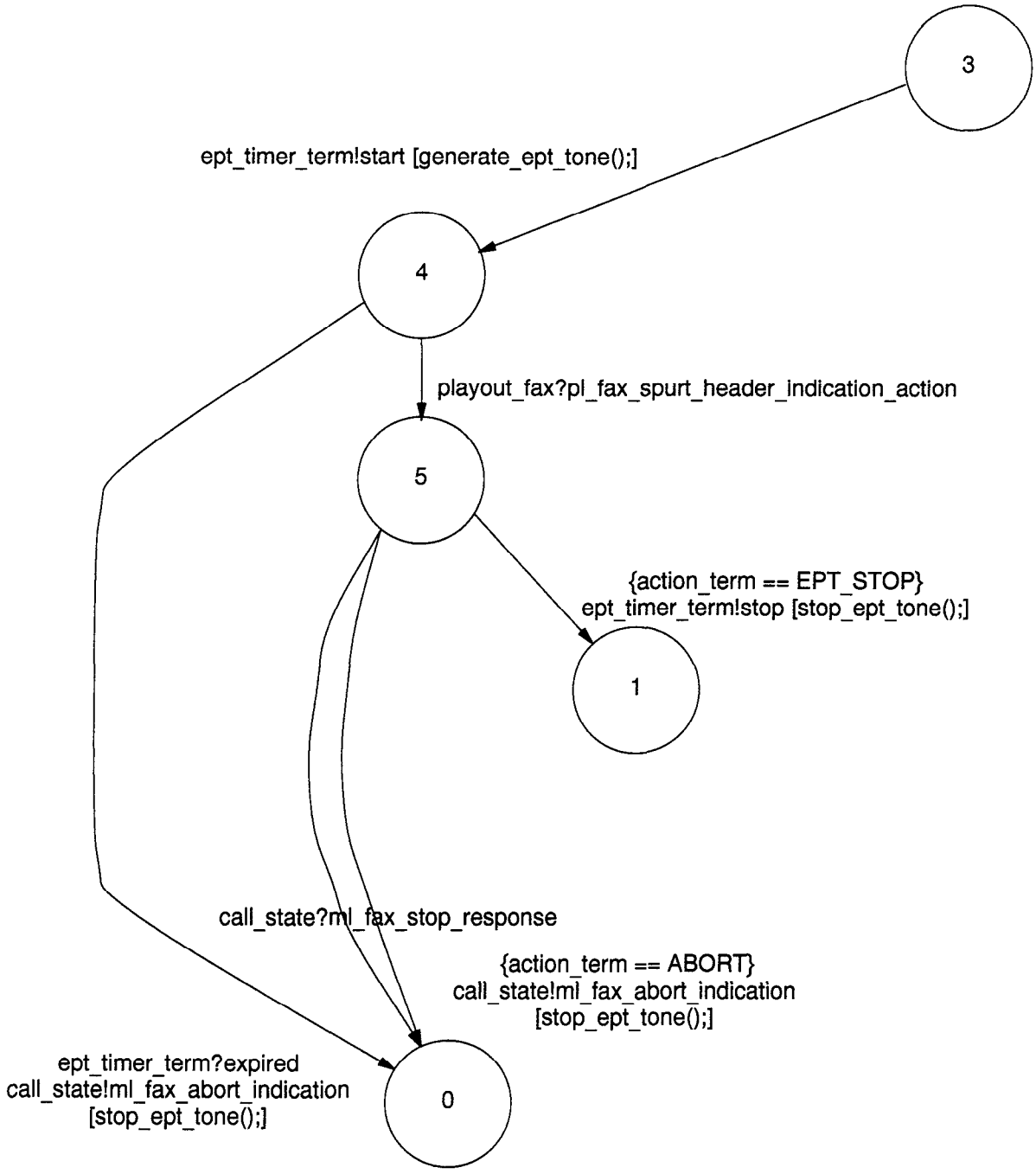


Figure E.29 – mod_layer_term: EPT state

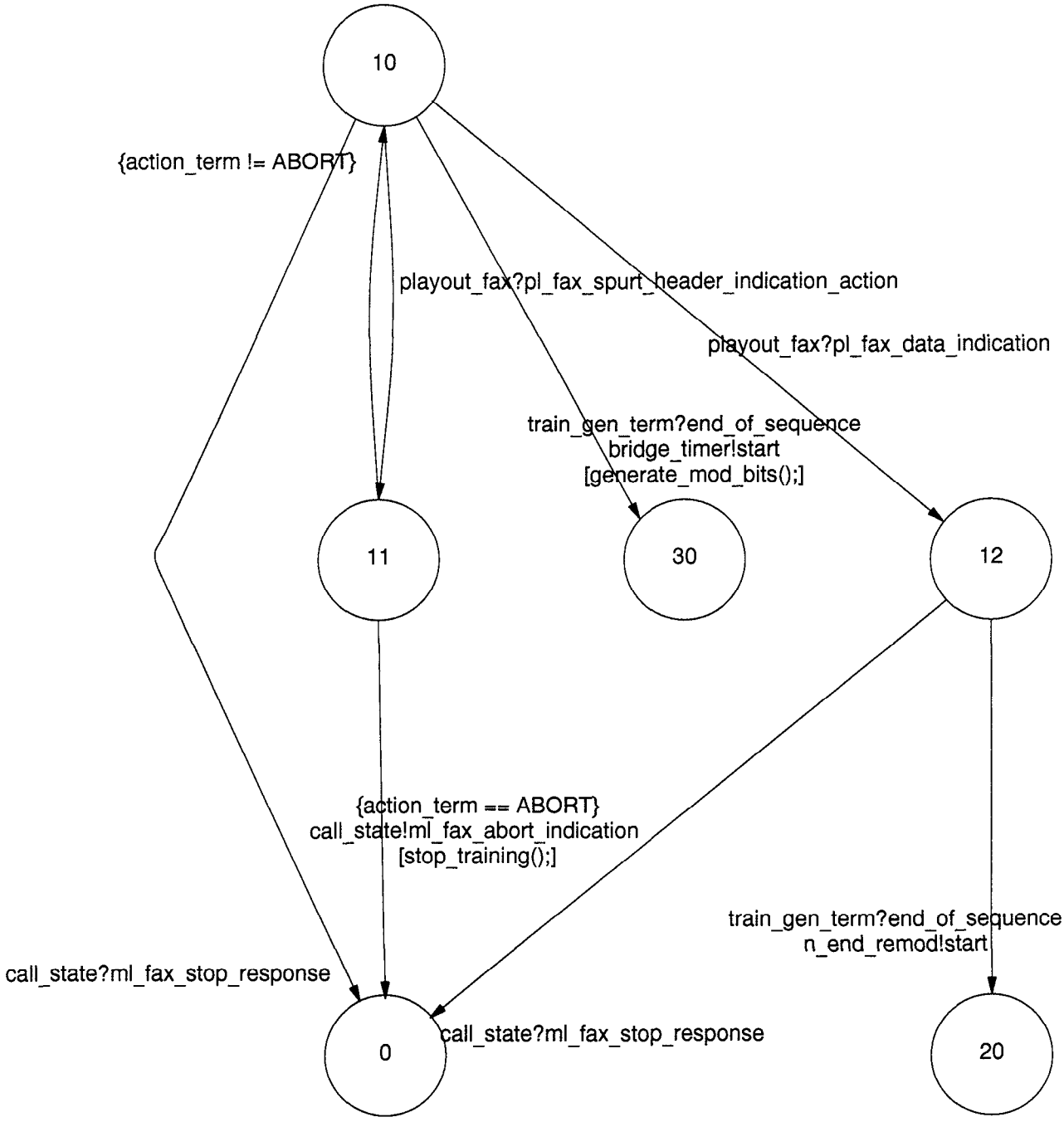


Figure E.30 – mod_layer_term: TRAIN state

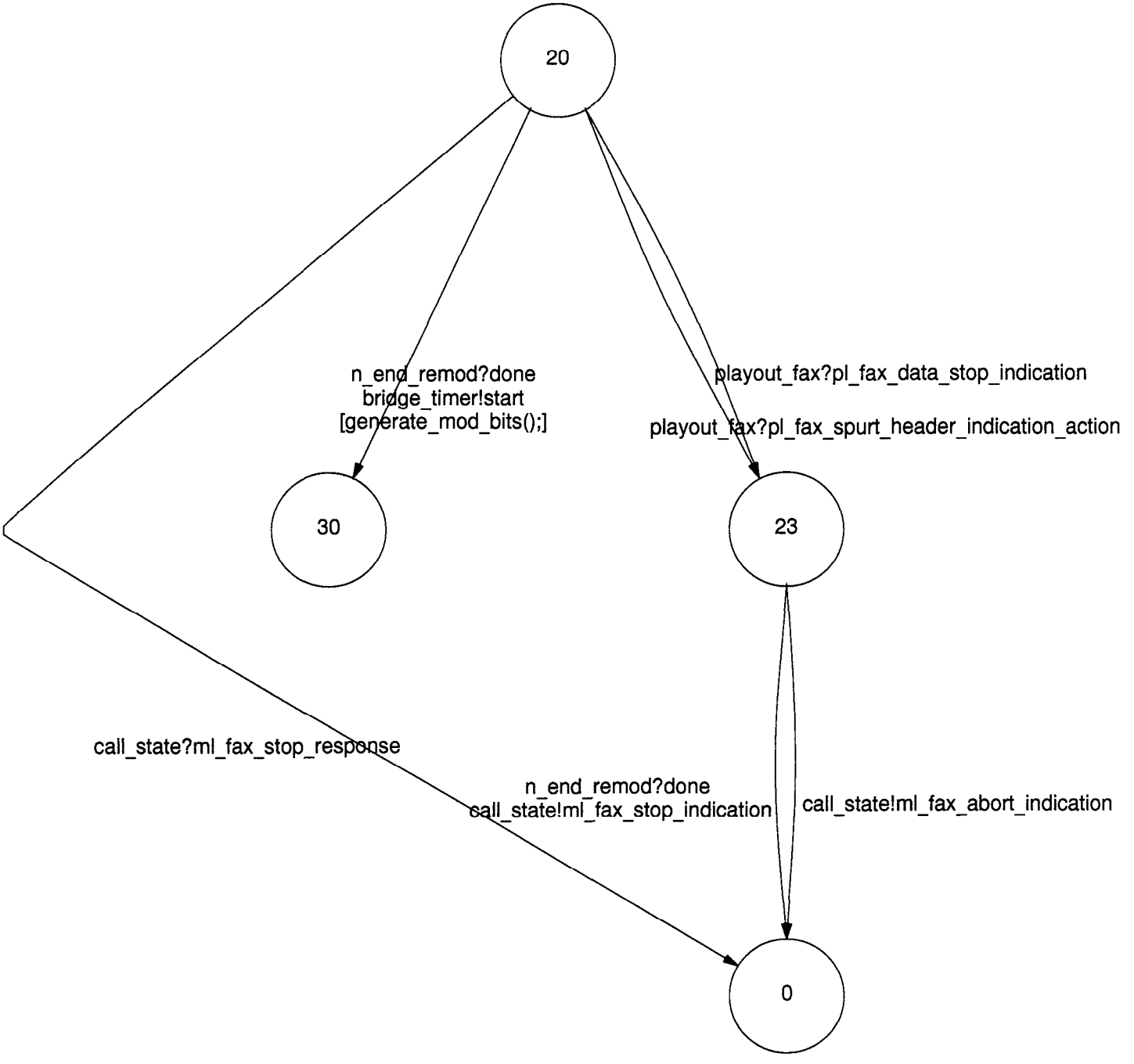


Figure E.31 – mod_layer_term: DATA state

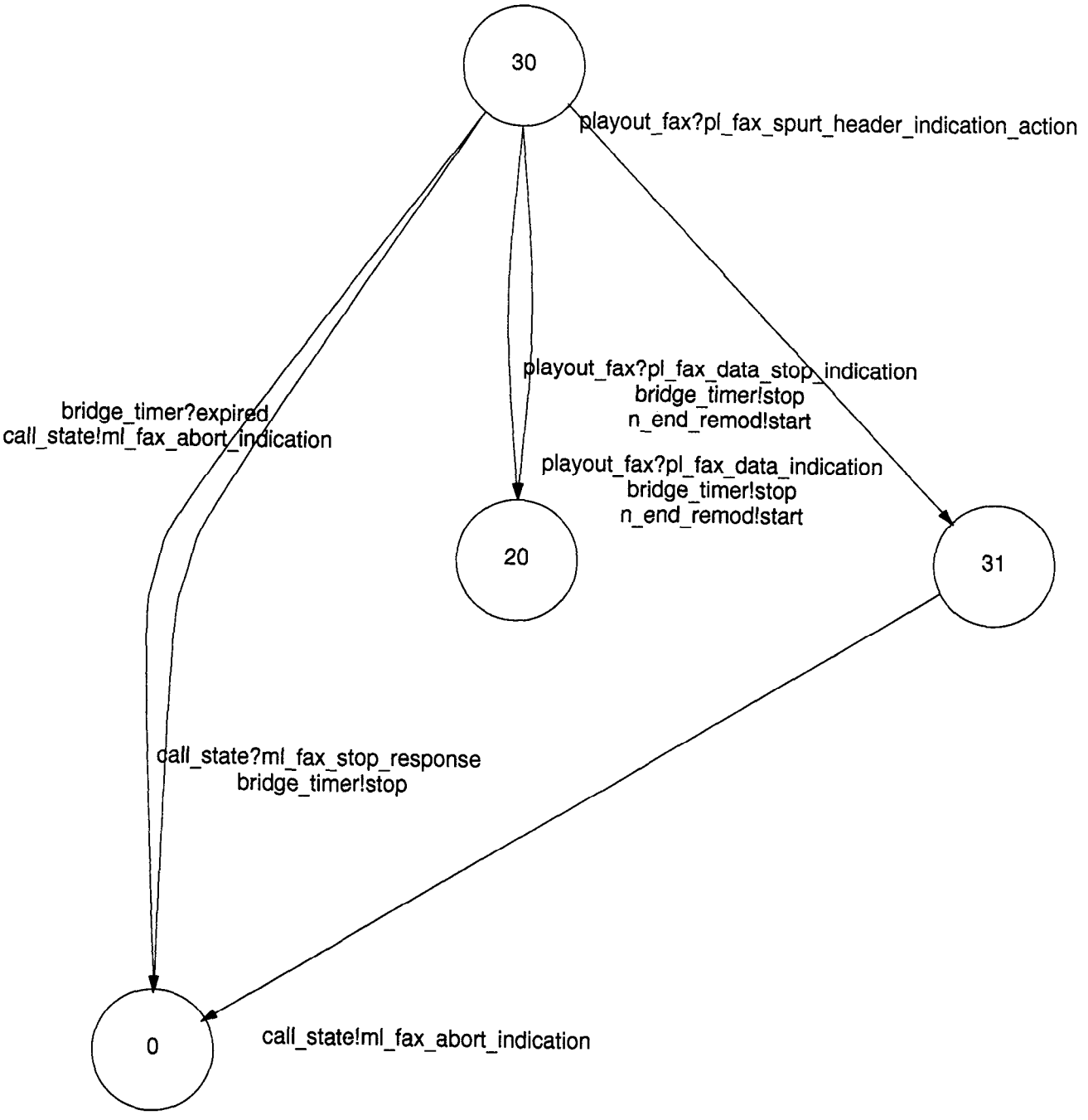


Figure E.32 – mod_layer_term: BRIDGE state

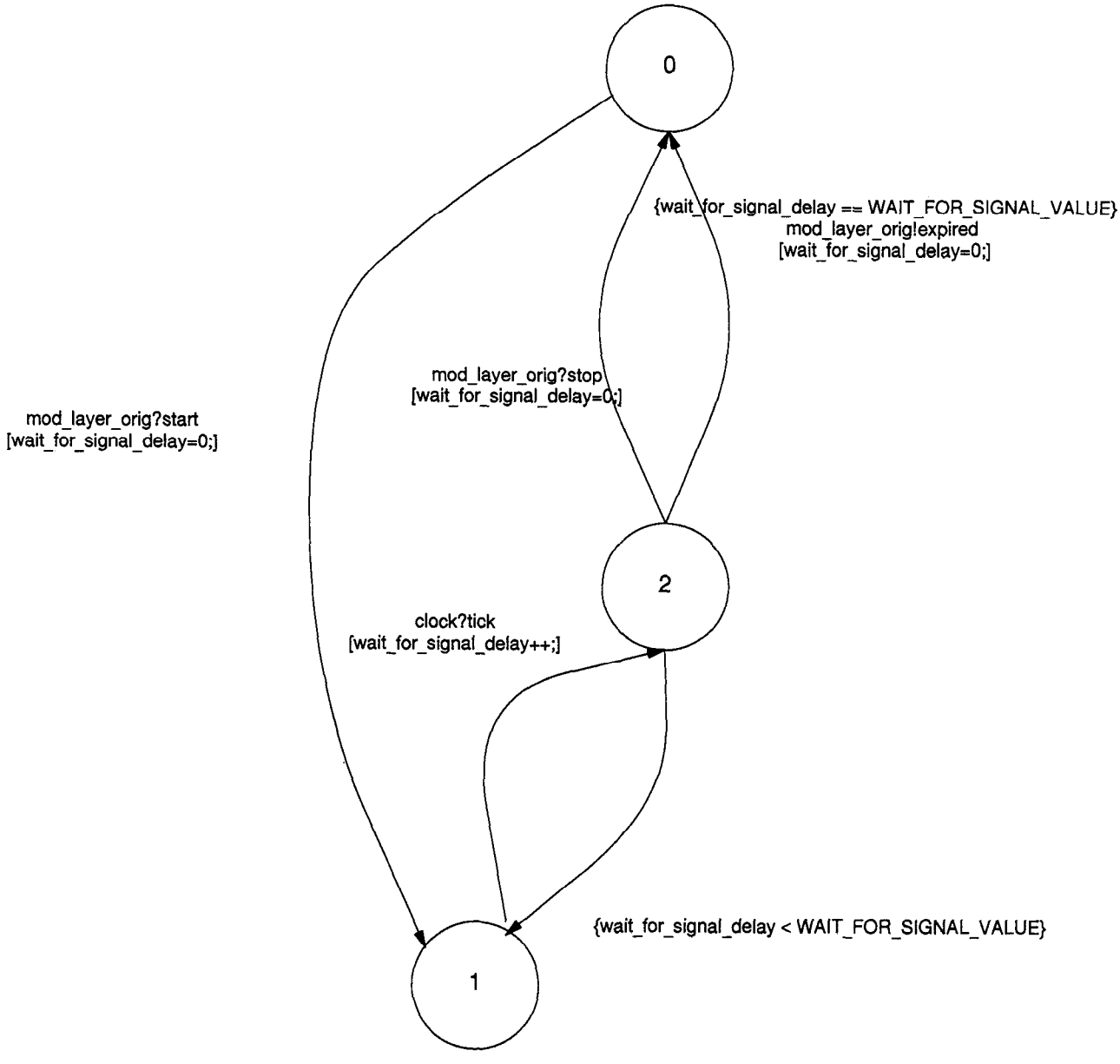


Figure E.33 – wait_time

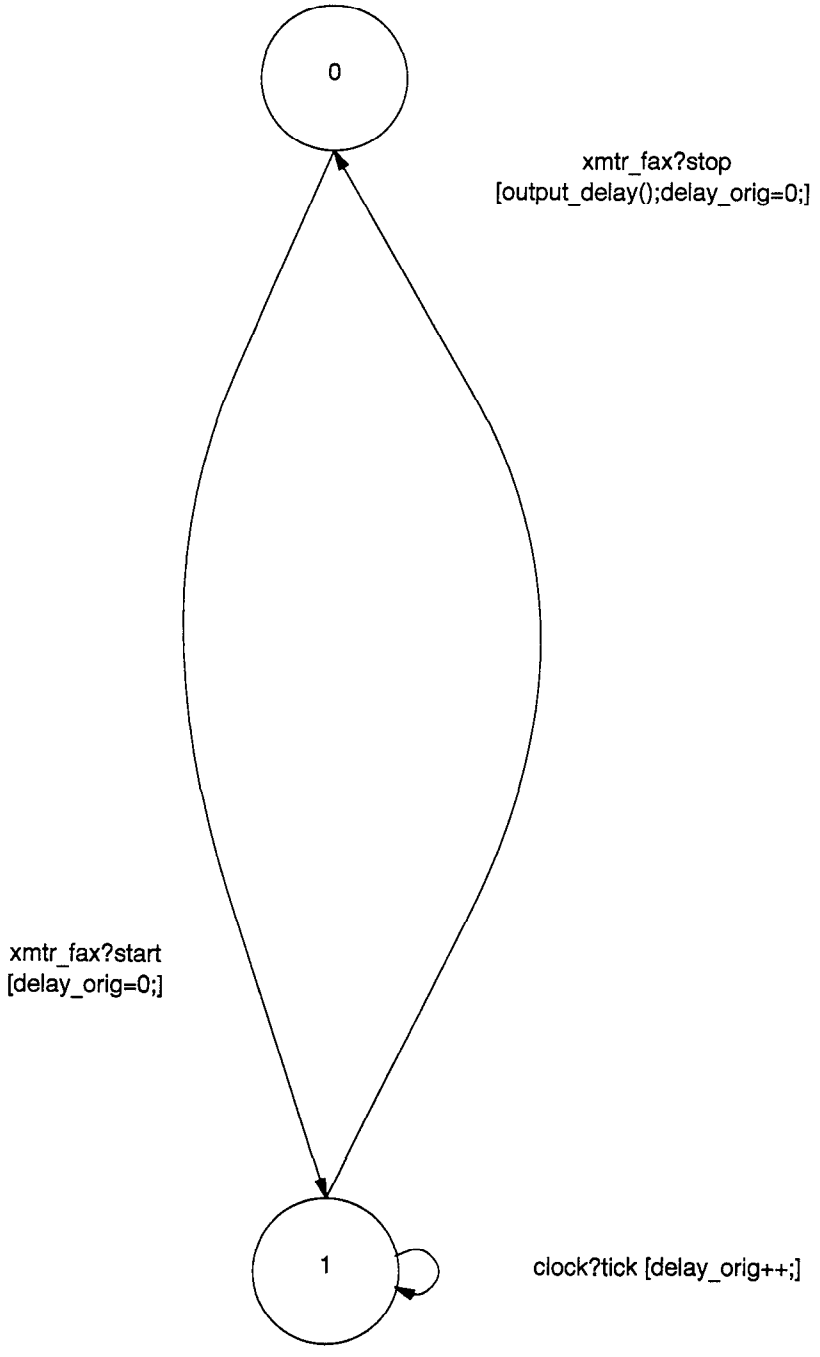


Figure E.34 – tvdelay_orig

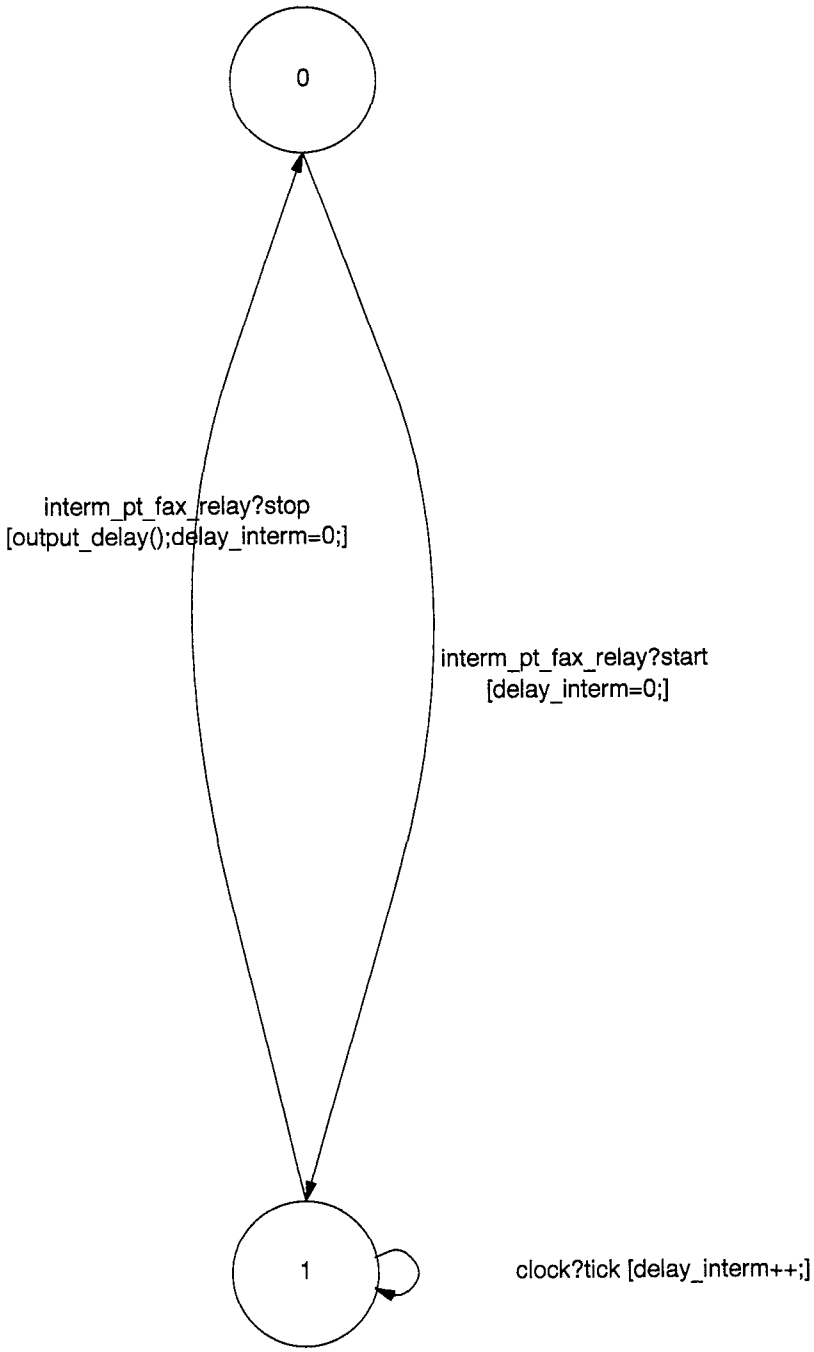


Figure E.35 – tvdelay_interm

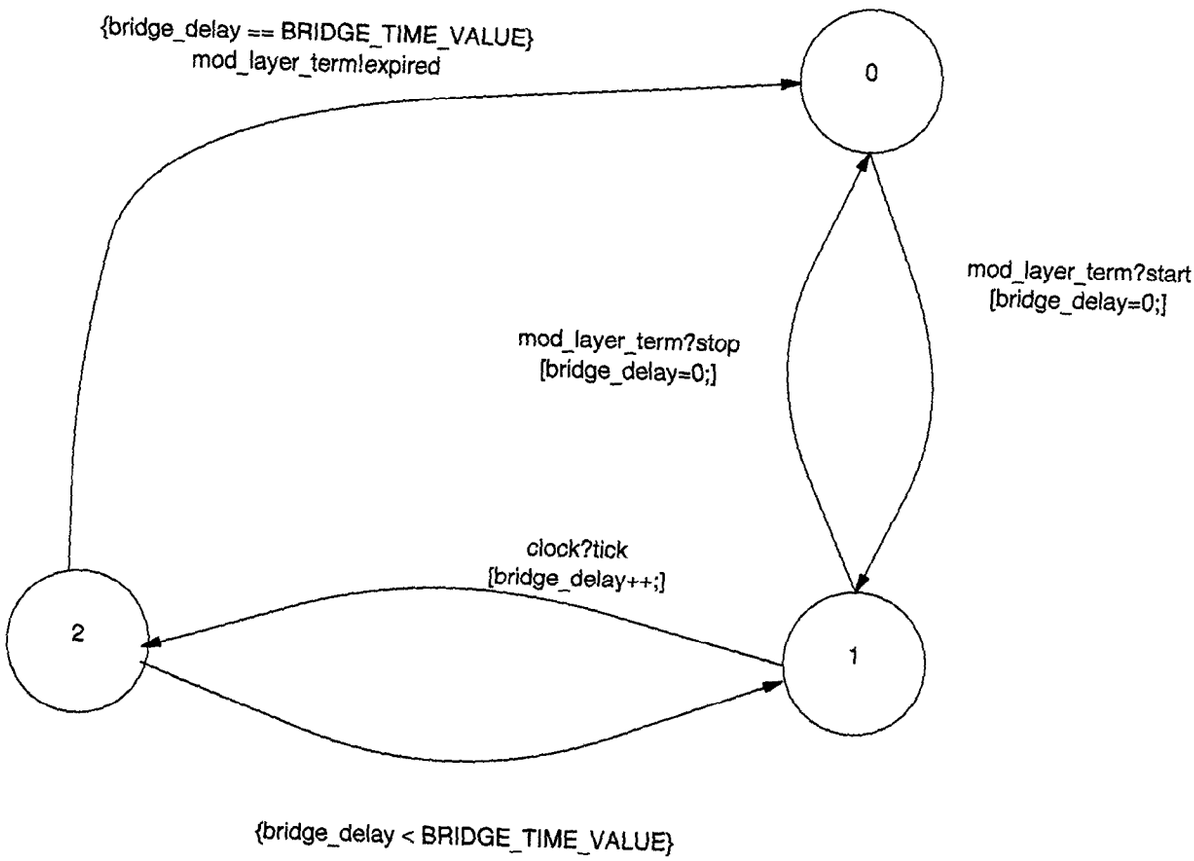


Figure E.36 – bridge_time

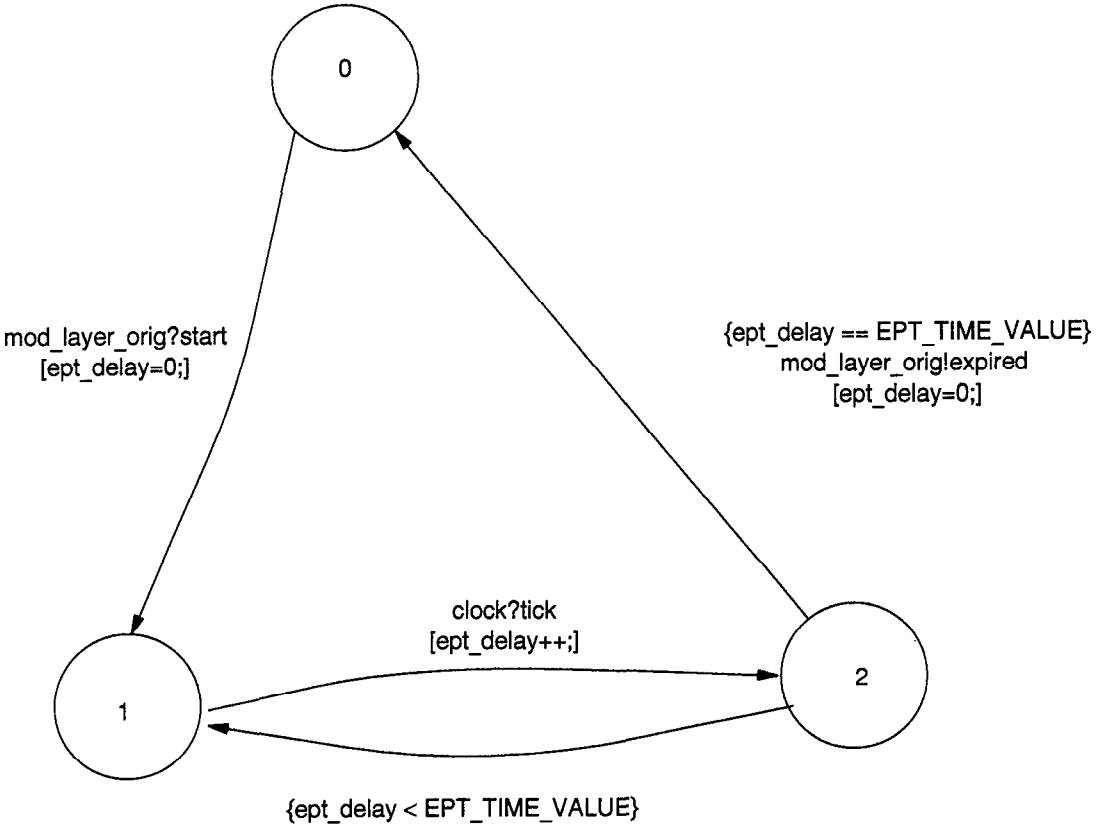


Figure E.37 – ept_timer_orig

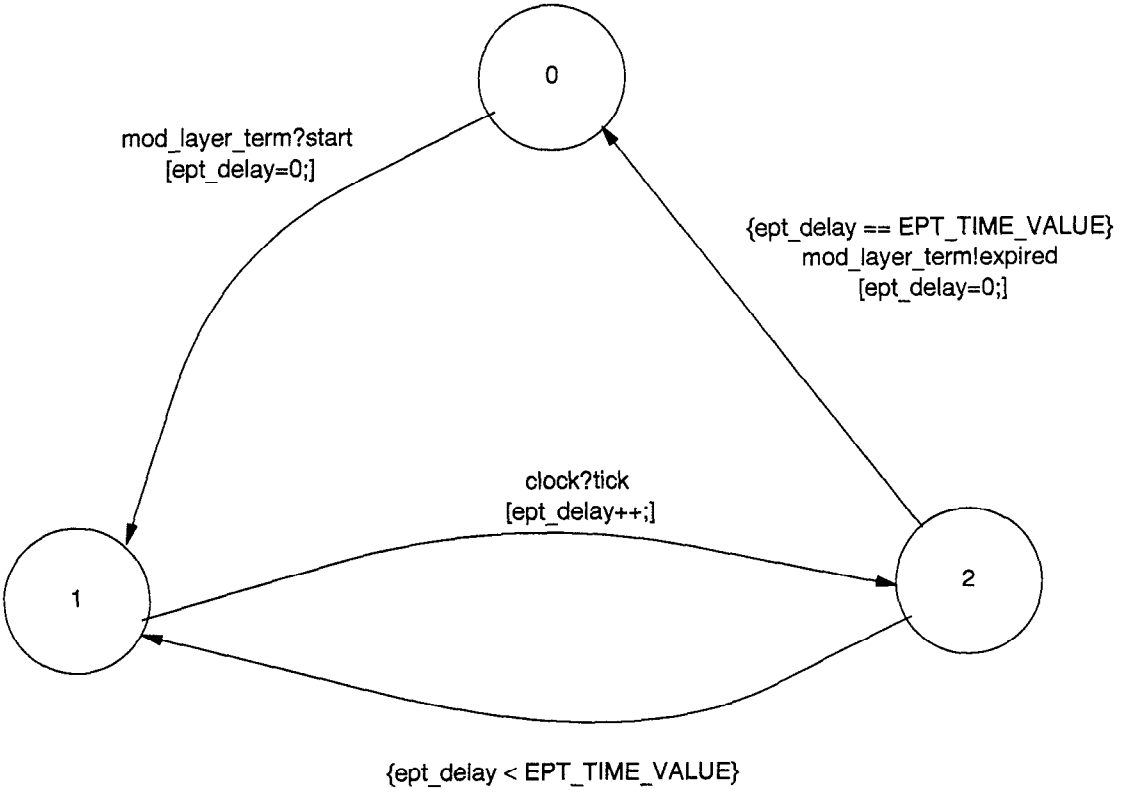


Figure E.38 – ept_timer_term

Annex F
(informative)

Bibliography

AT&T PUB 62411. ACCUNET® T1.5 Service Description and Interface Specifications. October 1985.

Bender, E. C., J. G. Kneuer, and W. J. Lawless. Digital Data System: Local Distribution System. *Bell System Technical Journal.* 54(5): 919-942; May-June 1975.

Mahoney, J. J., Jr., J. J. Mansell, and R. C. Matlack. Digital Data System: User's View of the Network. *Bell System Technical Journal.* 54(5): 883-884; May-June 1975.

The following references pertain only to annex B:

Hoare C.A.R., *Communicating Sequential Processes*, Prentice-Hall, Englewood Cliffs, NJ 1985.

Sabnani, K. and A. Lapone. Protocol analyzer and verifier, pp. 29-34 in *Protocol Specification*, Testing and Verification VI, B. Sarikaya and G. V. Bochmann, eds., Elsevier (North-Holland), Amsterdam, 1987.

Sherif, M. H. and A. S. Krishnakumar. Evaluation of protocol from formal specifications· A case study with LAPD, *GLOBECOM '90*, San Diego, December 2-5, 1990, Vol 2: 879-886.