ARTIC STREAMS Support WAN Driver Interface Reference

Release 1.6

www.radisys.com

007-01232-0002 • September 2003

September 2003

Contents

Before you begin ... xi
- Contents overview ... xi
- Conventions ... xii
  - Adapter names ... xii
  - Terms used in this book ... xii
  - Notations ... xiii
- Reference publications ... xiv
- Customer Support ... xvi
  - Accessing the Web Site ... xvi
  - Calling Technical Support ... xvi

Summary of changes ... xvii
- Release 1.6 ... xvii
- Release 1.4 and Release 1.5 ... xviii
- Release 1.3 ... xviii
- Release 1.2 ... xviii

Chapter 1: Overview
- Supported adapters, hardware, and protocols ... 3

Chapter 2: Protocol descriptions
- HDLC framing ... 7
- Bisynchronous protocol ... 8
- SS7 low-level processing ... 9
  - WAN driver in relation to MTP2 ... 10
  - Special SS7 features ... 11
  - Special TTC SS7 features ... 11
  - SS7 SU Reception (DAEDR) ... 12
  - SU filtering ... 13
  - SU transmission (DAEDT) ... 14
  - Error Rate Monitor (ERM) ... 17
    - Implementation of SUERM for SS7 ... 17
    - Implementation of SUERM for TTC SS7 ... 17
    - Implementation of AERM for SS7 and TTC SS7 ... 18
  - Clear Channel Capability Mode ... 19
    - Physical layer ... 19
    - LSSU/FISU/MSU length indicator/sequence numbering ... 19
    - Acceptance of alignment ... 20
    - Error monitoring ... 20
  - T1/E1/J1 interface ... 22
    - SC-bus implementation ... 24
    - CT-bus implementation ... 25
  - ATM in SS7 environments ... 27

AAL5 protocol reference model 27
Physical layer 28
ATM layer 29
ATM Adaptation Layer 5 (AAL5) 30
Common Part (CP) 30
Service Specific Convergence Sublayer (SSCS) 30
Operation and Maintenance (OAM) 31

Chapter 3: WAN driver STREAMS interface

About minor numbers 33
Configuring the WAN driver 33
Creating STREAMS 34
Non-clone open with SNID decode 35
Non-clone open with no SNID decode 35
Clone open with SNID decode 35
Clone open with no SNID decode 35
Types of WAN driver STREAMS messages and commands 36

Chapter 4: Serial and Multiplexed WAN drivers (command sequences)

Serial synchronous WAN driver running SS7 protocol 37
Serial synchronous WAN driver in HDLC framing mode 40
Serial synchronous WAN driver in bisynchronous mode 42
Multiplexed WAN driver in SS7 or HDLC framing 44
Multiplexed WAN driver in Clear Channel Capability mode 45
Multiplexed WAN driver in ATM mode 46
SC-bus connection scenarios 47
CT-bus connection scenarios 47

Chapter 5: Serial and Multiplexed WAN drivers (common operations)

STREAMS service messages 50
WAN_SID — Set subnetwork ID 51
WAN_REG — Registration message — start hardware 54
WAN_CTL — Connection management 56
WAN_DAT — Data messages for transmission and reception 61
STREAMS management commands 67
W_DISABLE/W_ENABLE — Disable/enable transmission of data 69
W_GETDRVINFO — Get driver configuration information 71
W_GETHWTYPE — Get hardware type 74
W_GETSTATS — Get statistics 78
W_ZEROSTATS — Clear channel statistics 81
W_SETTUNE — Set configuration 83
W_GETTUNE — Get configuration 93
W_SET_SNID — Allocate internal channel and associate SNID to it 94
W_GET_SNID — Get the assigned internal channel ID 101
W_REL_SNID — Release internal channel ID 102

Chapter 6: Signaling System Number 7 (SS7) (specific operations)

Relation between SS7 and HDLC modes 103
STREAMS service messages for SS7 104
WAN_ACTSS7 — Control SS7 features 105
WAN_NOTIFSS7 — Notify SS7 status 107
WAN_RESETSS7 — Reset filtering operation 109

STREAMS management commands for SS7 111
W_SETSS7 — Set SS7 configuration parameters 112
W_GETSS7 — Get SS7 configuration parameters 115
W_SETSS7_JPN — Set TTC SS7 configuration parameters 117
W_GETSS7_JPN — Get TTC SS7 configuration parameters 120
W_SETSS7_CCC — Set SS7 Clear Channel Capability configuration parameters 122
W_GETSS7_CCC — Get SS7 Clear Channel Capability configuration parameters 125

Chapter 7: T1/E1 interface (specific operations)

Identifying the T1/E1 components 128
STREAMS service messages for T1/E1 128
WAN_NOTIFDI — Inform upper level of T1/E1 events 129
WAN_NOTIFTIM — Send a timestamped notification 134
STREAMS management commands for T1/E1 136
W_DI_TEST_CFG — Set test configuration for a physical port 137
W_SET_PHY_PIPE — Define and undefine time slots 140
W_GET_PHY_PIPE — Retrieve time-slot information 144
W_SETCH_MAP — Set up channel map table 146
W_GETCH_MAP — Get channel map table settings 156
W_SETDI — Set attributes common to all digital interfaces 158
W_GETDI — Get attributes common to all digital interfaces 164
W_SETDI_PORT — Set attributes of a physical port 165
W_GETDI_PORT — Get attributes of a physical port 174
W_GETDI_STATS — Get port statistics 176
W_ZERODI_STATS — Clear port statistics 178
W_SETDI_LPBK — Put port in loopback 179
W_SETDI_NOTIF — Set event filter for a physical port 181
W_SET_TIMESTAMP — Set timestamp 183

Chapter 8: ATM (specific operations)

STREAMS service messages for ATM 186
WAN_NOTIF_ATM — Notify ATM cell stream status 187
STREAMS management commands for ATM. 189
W_SET_ATM — Define parameters for a physical layer of an ATM cell stream 190
W_GET_ATM — Obtain ATM physical-layer parameters and current state 193
W_GET_VCC_STATS — Get statistics for a virtual channel 195
W_ZERO_VCC_STATS — Retrieve and clear statistics for a virtual channel 198
W_GET_ATM_STATS — Get statistics for a physical ATM cell stream 200
W_ZERO_ATM_STATS — Retrieve and clear statistics for a physical ATM cell stream 203

Chapter 9: Extensions to Serial WAN driver provided by RadiSys

Interacting with the Serial WAN driver 205
Serial WAN driver STREAMS interface 206
STREAMS service message for the Serial WAN driver 206
WAN_NOTIFY — Notification of errors 207
STREAMS management commands for the Serial WAN driver 208
W_SETLINE — Define line characteristics 209
W_GETLINE — Get line characteristics 220
W_SETSIG — Output signal control 221
W_GETSIG — Return the states of control 223
W_RESET — Reset communications chip 225

W_SENDBREAK — Send break character 227
W_SETMODE — Set port mode 229
W_STATIONADDR — Set filtering address 231

Chapter 10: Configuration and program development

Executable files required for ARTIC environments 233
Load WAN drivers 234
Command-line parameters 235
Initial line characteristics 242
Serial WAN driver in synchronous mode — defaults 242
Multiplexed WAN driver for any of its ports — defaults 242
Multiplexed WAN driver for any of its channels — defaults 242
Interfacing with the WAN driver 243
LED use 244

Glossary 245

Index 251

Figures

Figure 2-1. WAN driver in relation to MTP2 ... 10
Figure 2-2. SU filtering ... 13
Figure 2-3. Format for each type of Signal Unit ... 15
Figure 2-4. 1.5 and 2.0 Mbit/s rate format for each type of Signal Unit ... 20
Figure 2-5. SC-bus switching support ... 24
Figure 2-6. CT-bus switching support ... 26
Figure 2-7. AAL5 protocol reference model ... 27
Figure 2-8. ATM cell at an NNI ... 29
Figure 3-1. Non-clone versus clone open ... 34
Figure 4-1. Serial synchronous WAN driver running SS7 protocol ... 39
Figure 4-2. Serial synchronous WAN driver in HDLC framing mode ... 41
Figure 4-3. Serial synchronous WAN driver in bisynchronous mode ... 43
Figure 5-1. Encoded SNID ... 51
Figure 5-2. Message flow for WAN_SID ... 53
Figure 5-3. Message flow for WAN_REG ... 55
Figure 5-4. Message flow for WAN_CTL ... 60
Figure 5-5. Message flow for WAN_DAT ... 63
Figure 5-6. Message flow for WAN_DAT — ATM protocol mode ... 66
Figure 5-7. Message flow for W_ENABLE/W_DISABLE ... 70
Figure 5-8. Message flow for W_GETDRVINFO ... 73
Figure 5-9. Message flow for W_GETHWTYPE ... 77
Figure 5-10. Message flow for W_GETSTATS ... 80
Figure 5-11. Message flow for W_ZEROSTATS ... 82
Figure 5-12. Format of WAN_interface of WAN_hddef structure ... 86
Figure 5-13. Message flow for W_SETTUNE (SS7) ... 92
Figure 5-14. Message flow for W_GETTUNE ... 93
Figure 5-15. Connection points that make a data path ... 99
Figure 5-16. Message flow for W_SET_SNID ... 100
Figure 5-17. Message flow for W_GET_SNID ... 101
Figure 5-18. Message flow for W_REL_SNID ... 102
Figure 6-1. Message flow for WAN_ACTSS7 ... 106
Figure 6-2. Message flow for WAN_NOTIFSS7 ... 108
Figure 6-3. Message flow for WAN_RESETSS7 ... 110
Figure 6-4. Message flow for W_SETSS7 ... 114
Figure 6-5. Message flow for W_GETSS7 ... 116
Figure 6-6. Message flow for W_SETSS7_JPN ... 119
Figure 6-7. Message flow for W_GETSS7_JPN ... 121
Figure 6-8. Message flow for W_SETSS7_CCC ... 124
Figure 6-9. Message flow for W_GETSS7_CCC ... 126
Figure 7-1. Message flow for W_NOTIFDI ... 133
Figure 7-2. Message flow for WAN_NOTIFTIM ... 135
Figure 7-3. Message flow for W_DI_TEST_CFG ... 139
Figure 7-4. Message flow for W_SET_PHY_PIPE ... 143
Figure 7-5. Message flow for W_GET_PHY_PIPE ... 145
Figure 7-6. Port assignments for ARTIC960 4-Port T1/E1 Mezzanine Card ... 147
Figure 7-7. Port assignments for ARTIC 4-Port T1/E1/J1 DSP PMC and ARTIC 1000/2000 Series ... 149
Figure 7-8. Possible paths ... 154

Figure 7-9. Message flow for W_GETCH_MAP 157
Figure 7-10. Relationship between recovered clocks........ 163
Figure 7-11. Message flow for W_SETDI........ 163
Figure 7-12. Message flow for W_GETDI........ 164
Figure 7-13. Message flow for W_SETDI_PORT 173
Figure 7-14. Message flow for W_GETDI_PORT 175
Figure 7-15. Message flow for W_GETDI_STATS 177
Figure 7-16. Message flow for W_ZERODI_STATS 178
Figure 7-17. Message flow for W_SETDI_LPBK 180
Figure 7-18. Message flow for W_SETDI_NOTIF........ 182
Figure 7-19. Message flow for W_SET_TIMESTAMP 183
Figure 8-1. Message flow for WAN_NOTIF_ATM........ 188
Figure 8-2. Message flow for W_SET_ATM........ 192
Figure 8-3. Message flow for W_GET_ATM........ 194
Figure 8-4. Message flow for W_GET_VCC_STATS........ 197
Figure 8-5. Message flow for W_ZERO_VCC_STATS........ 199
Figure 8-6. Message flow for W_GET_ATM_STATS........ 202
Figure 8-7. Message flow for W_ZERO_ATM_STATS........ 204
Figure 9-1. Message flow for WAN_NOTIFY........ 207
Figure 9-2. Message flow for W_SETLINE 219
Figure 9-3. Message flow for W_GETLINE 220
Figure 9-4. Message flow for W_SETSIG 222
Figure 9-5. Message flow for W_GETSIG 224
Figure 9-6. Message flow for W_RESET 226
Figure 9-7. Message flow for W_SENDBREAK 228
Figure 9-8. Message flow for W_SETMODE 230
Figure 9-9. Message flow for W_STATIONADDR 232

Tables

Table 1-1. Adapters supporting SS7 MTP2/HDLC and monitoring APIs 3
Table 1-2. Adapters supporting SS7 MTP2/HSL monitoring and HDLC Send and Receive APIs 3
Table 1-3. Summary of supported hardware with ARTIC adapters 4
Table 1-4. Summary of supported protocols 5
Table 2-1. HDLC Framing 7
Table 2-2. Valid BISYNC message types 9
Table 2-3. ERM summary 18
Table 2-4. T1/E1 available features 22
Table 5-1. Summary of service messages 50
Table 5-2. STREAMS management commands common to Serial and Multiplexed WAN drivers 68
Table 5-3. Actions taken for WAN_RMT_CC_REW and WAN_RMT_CC_ACT bits 92
Table 5-4. SET_SNID command—combinations for channelled mode for Multiplexed WAN driver 98
Table 6-1. STREAMS service messages and management commands for SS7 103
Table 7-1. STREAMS service message and management commands for the T1/E1 interface 127
Table 8-1. STREAMS service messages and management commands for ATM 185
Table 10-1. W_MONITOR_MODE — possible values 239
Table 10-2. Initialization error codes 241
Table 10-3. Header file organization 243
Table 10-4. LED usage summary 244

Before you begin

This book provides information on the interface between the Wide Area Network (WAN) device driver (referred to in this book as the *WAN driver*) and other subsystems in a STREAMS environment.

Contents overview

The following describes the contents of this book.

Chapter		Page	Description
1	Overview	1	Provides an overview of the WAN driver and a summary of PMCs, electrical interfaces, and supported protocols.
2	Protocol descriptions	7	Provides a brief description of various protocols.
		33	Provides STREAMS-specific details.
4	Serial and Multiplexed WAN drivers (command sequences)	37	Provides message exchanges for the Serial and Multiplexed WAN drivers when running various protocols.
5	Serial and Multiplexed WAN drivers (common operations)	49	Provides operations common to the Serial and the Multiplexed WAN drivers.
6	Signaling System Number 7 (SS7) (specific operations)	103	Provides operations specific to Signaling System Number 7 (SS7). These SS7 operations are common to the Serial and the Multiplexed WAN drivers.
7	T1/E1 interface (specific operations)	127	Provides operations specific to the T1/E1 interface (multiplexed mode).
8	ATM (specific operations)	185	Provides operations specific to the ATM protocol mode, which implements higher speeds for SS7 signaling.
9	Extensions to Serial WAN driver provided by RadiSys	205	Provides extensions to the Serial WAN driver provided by RadiSys.
10	Configuration and program development	233	Provides WAN driver load time configuration, initial port characteristics, and program development information.

Conventions

Adapter names

The following table shows the different ways an adapter can be identified.

Part #	Code Name	Product Name
IOP-CPCI-10000	Tomcat	ARTIC 1000 CompactPCI I/O Platform
IOP-PMC-01000	Hornet	ARTIC 4-Port T1/E1/J1 DSP PMC
IOP-PMC-02000	Spitfire	ARTIC 4-Port T1/E1/J1 Line PMC
IOP-PMC-03000	Remora Rear I/O	ARTIC 4-Port Serial PMC
IOP-RTM-00100	Hornet RTM	ARTIC 8-Port T1/E1/J1 / 2-Port Ethernet Rear Transition Module
IOP-RTM-00300	Remora RTM	ARTIC 8-Port Serial / 2-Port Ethernet Rear Transition Module
87H3800	Stingray	ARTIC960Rx Frame Relay PCI Adapter
87H3450	Tigershark	ARTIC960Hx PCI Adapter
87H3530 87H3550	Mantaray	ARTIC960Rx PCI 3.3V 4MB VR ARTIC960Rx PCI 3.3V 8MB (no VR)
NA	NA	ARTIC960 2-Port Selectable PMC (paired with Mantaray)
87H3410	Remora	ARTIC960 4-Port Selectable PMC
87H3448	Orca	ARTIC960 4-Port T1/E1 Mezzanine Card
NA	NA	ARTIC960 4-Port AIB
NA	NA	Cipher PMC001
NA	NA	Cipher AIB 802

Terms used in this book

The terms used in this book are as follows:

ARTIC960 and ARTIC 1000/2000 Series
: refer always to the RadiSys ARTIC960 and ARTIC 1000/2000 Series adapters.

WAN driver
: refers to a WAN driver that runs in the STREAMS environment on either an ARTIC960 adapter or an ARTIC 1000/2000 Series adapter.

ARTIC960 WAN driver
: refers to a WAN driver that runs in the STREAMS environment on an ARTIC960 adapter.

ARTIC 1000/2000 Series WAN driver
: refers to a WAN driver that runs in the STREAMS environment on an ARTIC 1000/2000 Series adapter.

Serial WAN driver
refers to a WAN driver that provides access to a physical interface capable of serial communications over which multiplexing of data is not possible or available (for example, a 56-kbps leased line).

Multiplexed WAN driver
refers to a WAN driver that provides access to a physical interface over which multiplexing of data as separate logical channels (or time slots) is possible (for example, T1, E1 or CT bus).

Line refers to one of the physical ports controlled by the Serial WAN driver.

Line or channel
for the Multiplexed WAN driver, refers to one of the multiplexed signals on a port (or one of the time slots).

Notations

This manual uses the following conventions:

- All numbers are decimal unless otherwise stated.
- All bit numbering conforms to the industry standard of the most significant bit having the highest bit number
- All counts in this book are assumed to start at zero.
- `Data structures and syntax strings appear in this font.`

Notes indicate important information about the product.

Tips indicate alternate techniques or procedures that you can use to save time or better understand the product.

The globe indicates a World Wide Web address.

The book indicates a book or file.

ESD cautions indicate situations that *may* cause damage to hardware via electro-static discharge (ESD).

Cautions indicate potentially hazardous situations which, if not avoided, may result in minor or moderate injury, or damage to data or hardware. It may also alert you about unsafe practices.

Warnings indicate potentially hazardous situations which, if not avoided, can result in death or serious injury.

Danger indicates imminently hazardous situations which, if not avoided, will result in death or serious injury.

Reference publications

- *ANSI T1.403-1999 Specification*
- *ITU-T Recommendation Q.703, Specification of SS#7 - Signalling Link (MTP2)* (hereafter referred to as ITU-T Q.703)
- *ITU-T Recommendation G.704, Synchronous Frame Structures used at 1544, 6312, 2048, 8488 and 44736 Kbits/s Hierarchical Levels* (hereafter referred to as ITU-T G.704)
- *ITU-T Recommendation G.775, Loss Of Signal (LOS) and Alarm Indication Signal (AIS) Defect Detection and Clearance Criteria* (hereafter referred to as ITU-T G.775)
- *ITU-T Recommendation I.361, B-ISDN ATM Layer Specification*, 11/95 (hereafter referred to as ITU-T I.361)
- *ITU-T Recommendation I.363, B-ISDN ATM Adaptation Layer Specification*, 03/93 (hereafter referred to as ITU-T I.363)
- *ITU-T Recommendation I.363.5, B-ISDN ATM Adaptation Layer Specification:Type 5 AAL*, 08/96 (hereafter referred to as ITU-T I.363.5)
- *ITU-T Recommendation I.610, B-ISDN Operation and Maintenance Principles and Functions*, 11/95 (hereafter referred to as ITU-T I.610)
- *ITU-T Recommendation I.432, B-ISDN User-Network Interface-Physical Layer Specification*, 03/93 (hereafter referred to as ITU-T I.432)
- *ITU-T Recommendation I.432.1, B-ISDN User-Network Interface-Physical Layer Specification: General Characteristics*, 08/96 (hereafter referred to as ITU-T I.432.1)
- *ITU-T Recommendation G.804, ATM cell mapping into Plesiochronous Digital Hierarchy (PDH)*, 11/93 (hereafter referred to as ITU-T G.804)
- *ITU-T Recommendation G.704, Synchronous Frame Structures used at 1544, 6312, 2048, 8488 and 44736 Kbits/s Hierarchical Levels*, 07/95 (hereafter referred to as ITU-T G.704)
- *Generic requirements for CCS Nodes Supporting ATM High-speed Signaling Links (HSLs), Bellcore GR-2878-CORE*, 11/95
- *The ATM Forum Technical Committee-E1 Physical Interface Specification af-phy-0064.000*, 09/96
- *The ATM Forum Technical Committee-DS1 Physical Layer Specification af-phy-0016.000*, 09/94
- *RFC 1406, Definitions of Managed Objects for DS1 and E1 Interface Types, Trunk MIB Working Group*, January 1993
- *Primary Rate User-Network Interface-Layer 1 Specifications ITU-T I.431*
- *RFC 1659 Definitions of Managed Objects for RS-232 like hardware devices using SMIv2*; B.Stewart; July 1994.
- *STREAMS Modules and Drivers, UNIX† SVR4.2*, UNIX Press

- *Infinon PEB2254*
- *VLSI Technology, Inc., SC4000 Universal Timeslot Interchange*
- SCSA architecture:
  - Software model — *SCSA Telephony Application Object Framework*
  - Hardware model — *SCSA*
- *RadiSys*
  - *ARTIC960 Programmer's Guide*
  - *ARTIC960 Programmer's Reference*
  - *ARTIC960 STREAMS Environment Reference*
  - *ARTIC 1000/2000 Software Reference*
  - *SS7 Data Link Layer Software Reference*
- *IBM*
  - *General Information — Binary Synchronous Communications*, GA27-3004-02
  - *Implementation of X.21 Interface General Information*, GA27-3287-03
- *SpiderX25 WAN Implementation Guide, r8.0,* by Spider Systems
- *SpiderISDN WAN Implementation Guide*, r4.0, by Shiva Corporation

Customer Support

Accessing the Web Site

RadiSys maintains an active site on the World Wide Web. The site contains current information about the company and locations of sales offices, new and existing products, contacts for sales, service, and technical support information. You can also send e-mail to RadiSys using the web site. In-depth printable service manuals and other documentation are available for download from the RadiSys web site:

Note: When sending e-mail for technical support, include information about both the hardware and software, plus a detailed description of the problem, including how to reproduce it.

To access the RadiSys web site, enter this URL in your web browser:
`http://www.radisys.com`

Then click on Support and Service to access a link to the documentation, drivers, and BIOS. Documentation is available at this Web site in Adobe† Acrobat† .PDF format, and may be viewed and printed using the free Acrobat† Reader† software. BIOS files are available as self-extracting files. Links are provided to various partners' web sites where any files and tools needed to install drivers are available for download.

Calling Technical Support

1. Have the RadiSys product information, such as name and release level, available.
2. Call Technical Support:
   - In the continental USA, Monday—Friday, 6:00 a.m.—5:00 p.m., Pacific Time, dial 866–385–6167.
   - Outside the USA, dial 503–615–1640 (add long distance/international access codes).
   - In Europe, Monday—Friday, 8:30 a.m.—5:00 p.m., dial +31–36–5365595.

Requests for sales, service, and technical support information receive prompt response.

If you purchased your RadiSys product from a third-party vendor, you can contact that vendor for service and support.

Summary of changes

This document contains the following changes.

Release 1.6

- Changed the ARTIC8260 environment to the ARTIC 1000/2000 Series environment.
- Added Clear Channel Capability mode information, which includes the following new STREAMS management commands:
  - W_SETSS7_CCC
  - W_GETSS7_CCC

  To ensure your adapter supports this mode, contact your RadiSys representative.

- Operations common to the Serial and Multiplexed WAN drivers:
  - Changed a STREAMS management command — W_GETDRVINFO
- Operations specific to Signaling System Number 7 (SS7):
  - Changed a STREAMS service message — WAN_ACTSS7
- Operations specific to T1/E1 interface:
  - Added a new STREAMS service message — WAN_NOTIFTIM
  - Added a new STREAMS management command — W_SET_TIMESTAMP
  - Changed STREAMS management commands:
    - W_SET_PHY_PIPE
    - W_SETDI_PORT
- Command line parameters:
  - Added a new parameter — W_TDM_CLOCK_RATE
  - Added new values for the parameter W_MONITOR_MODE

Release 1.4 and Release 1.5

Added support for TTC SS7, CT bus, and the ARTIC8260 environment.

TTC SS7 includes the following new commands:

- W_GETSS7_JPN
- W_SETSS7_JPN

The following commands have been changed.

- WAN_ACTSS7
- W_GETDRVINFO
- W_GETHWTYPE
- W_SETLINE
- W_SETSS7
- W_SETTUNE

The following command-line parameters have been added:

- BSN_FLAG
- PMC_SELECT
- RX_CRC_SELECT
- SS7_FILTER_COUNT
- W_MONITOR_MODE

Release 1.3

Added support for the SC bus. Changed the following commands:

- WAN_NOTIFDI
- W_SETDI
- W_SETCH_MAP
- W_GETCH_MAP

Release 1.2

- Added Serial WAN driver support for the following ARTIC adapters:
  - RadiSys ARTIC960 Frame Relay PCI Adapter
  - RadiSys ARTIC960 2-Port Selectable PMC
  - Cipher PMC001
- Added Serial WAN driver support for the X.21 electrical interface. This X.21 support is for leased-line, full-duplex, and external clocking only. X.21 is not supported on the Cipher 802 8-port adapter.

- Changed the W_SETLINE command for the Serial WAN driver.
- Added support for the Multiplexed WAN driver:
  - Added new AAL5-specific commands.
  - Changed the following commands:
    - WAN_DAT
    - WAN_SET_SNID
    - W_SETTUNE
    - W_DITEST_CFG
- Added a new header file, wan_atm.h.

1 Overview

This chapter provides an overview of the WAN drivers and a summary of PMCs, electrical interfaces, and supported protocols

The Wide Area Network Device Drivers, referred to in this book as *WAN driver,* are STREAMS drivers that provide physical-layer communications support in the STREAMS environment. STREAMS defines standard interfaces for input and output, and the mechanism is simple and flexible. The WAN drivers provide support for transmitting and receiving data, in addition to providing support for programming the hardware to the appropriate line parameters.

The WAN drivers run in the STREAMS environment on a RadiSys ARTIC960 adapter and an ARTIC 1000/2000 Series adapter.

ARTIC960 Adapter

The STREAMS environment emulation is provided by the On-card STREAMS Subsystem (OSS) module that is loaded on the RadiSys ARTIC960 adapter. AIX†, OS/2†, Windows NT†, Novell, and OEM operating system applications use the STREAMS Access Library (SAL) to gain access to the ARTIC960 STREAMS environment. Such applications often configure additional protocol layer processing (that is, X.25, Frame Relay, SS7) in the ARTIC960 STREAMS environment.

The ARTIC960 WAN driver operates in little endian format. The system unit software can be operating in little or big endian format. If the system unit software is operating in big endian format, the ARTIC960 adapter's memory regions will handle the issues related with little/big endian. Refer to the *RadiSys ARTIC960 STREAMS Environment Reference* for more details.

ARTIC 1000/2000 Series Adapter

The STREAMS environment emulation is provided by the On-card STREAMS kernel that is loaded on the adapter. Windows NT†, Linux, Solaris and OEM operating system applications use the STREAMS Access Library (SAL) to gain access to the ARTIC 1000/2000 Series STREAMS environment.

The ARTIC 1000/2000 Series WAN driver can support up to two PMC adapters configured as a *first* and *second* PMC adapter in a CompactPCI† (cPCI) environment. In addition, in the cPCI environment, a Rear Transition Module (RTM) can be used in conjunction with the ARTIC 1000 Series adapter and PMCs. This RTM is used to connect all cables from the rear of the system unit.

The ARTIC 1000 Series adapter supports two PMC adapters.
The ARTIC 2000 Series adapter supports one PMC adapter.

The ARTIC 1000/2000 Series WAN driver operates in big endian format. The system unit software needs to ensure that data is presented to the WAN driver in big endian format.

The WAN drivers operate in serial or multiplexed mode. When a physical interface provides capability for multiplexing (that is, a T1, E1 or CT bus), the Multiplexed WAN driver is used.

The WAN drivers recognize various Application Interface Boards (AIBs) or PCI Mezzanine Cards (PMCs).

- The structures shown in this book are for illustration purposes. the structures are defined in *include* files that are distributed with the WAN driver available from the World Wide Web (see *Customer Support* on page *xvi* for instructions).
- Electrical interfaces are selected by cable type except for the X.21 electrical interface, which is selected by issuing a W_SETLINE command.
- All reserved fields named as w_reservedx or w_spare *must* be set to zero, unless specified otherwise.
- A combined value of bit-wise ORed fields equaling zero indicates there is no change from previous settings or default settings.

Supported adapters, hardware, and protocols

Table 1-1. Adapters supporting SS7 MTP2/HDLC and monitoring APIs

				Send/Receive		Monitor	
Part Number	**Form Factor**	**Ports**	**Physical Interface**	**Maximum MTP2/HSL links**	**Maximum HSL links**	**Maximum MTP2/HDLC links**	**Maximum HSL links**
IOP-1107-T8	cPCI	8	T1/E1/J1	128	N/A	128	8
IOP-1107-T4	cPCI	4	T1/E1/J1	64	N/A	72	4
IOP-1107-H8	cPCI	8	T1/E1/J1	N/A	4	128	8
IOP-1107-V8	cPCI	8	Serial V.35	8	N/A	8	N/A
IOP-1107-V4	cPCI	4	Serial V.35	4	N/A	4	N/A
IOP-2107-T4	PCI	4	T1/E1/J1	64	N/A	72	4
IOP-2107-V4	PCI	4	Serial V.35	4	N/A	4	N/A
IOP-2507-M4	PCI	4	T1/E1/J1	64	2	72	4

Refer to the RadiSys *SS7 Data Link Layer Software Reference* for information about the SS7 MTP2 and HDLC APIs.

Table 1-2. Adapters supporting SS7 MTP2/HSL monitoring and HDLC Send and Receive APIs

				Send/Receive		Monitor	
Part Number	**Form Factor**	**Ports**	**Physical Interface**	**Maximum MTP2/HSL links**	**Maximum HSL links**	**Maximum MTP2/HDLC links**	**Maximum HSL links**
IOP-1107-V21	cPCI	8	T1/E1/J1	128	4	128	8
IOP-CPCI-11100	cPCI	4	T1/E1/J1	64	2	72	4
IOP-PCI-11100	PCI	4	T1/E1/J1	64	2	72	4
IOP-PCI-11100L	PCI 5V only	4	T1/E1/J1	64	2	72	4

Table 1-3. Summary of supported hardware with ARTIC adapters

PMC name / Part number	# of ports	Channels per port	# of logical channels supported by the hardware	Logical channels available for data transfer	Electrical interfaces	Protocol supported	WAN driver REL file
ARTIC960 PMCs							
ARTIC960 4-Port Selectable PMC 87H3410	4	1	4	4	• RS-232 • RS-449/v.36 • V.35 DTE • V.35 DCE • EIA-530 • X.21	• HDLC framing only (default) • HDLC framing + SS7 low-level processing • Bisynchronous	ric_wvol.rel Synchronous Serial WAN Driver
ARTIC960 4-Port T1/E1 Mezzanine Card	4	32 for E1 24 for T1	96 for T1 or 128 for E1	32	• RJ-48 for T1 or E1 • BNC ungrounded/balanced connector for E1 • BNC grounded/unbalanced connector for E1 • Phone jack connector for T1	• HDLC framing only per channel (default) • HDLC framing + SS7 low-level processing per channel • AAL5 for NNI signaling	ric_wmux.rel Multiplexed WAN Driver ric_aal5.rel Multiplexed WAN Driver
ARTIC 1000/2000 Series							
ARTIC 4-Port Serial PMC 87H3410	4	1	4	4	• RS-232 • RS-449/v.36 • V.35 DTE • V.35 DCE • EIA-530 • X.21	• HDLC framing only (default) • HDLC framing + SS7 low-level processing	rpq_wans.rel Synchronous Serial WAN Driver
ARTIC 4-Port T1/E1/J1 DSP PMC IOP-PMC-5000	4	32 for E1 24 for T1	96 for T1 or 128 for E1	64	• RJ-48 for T1 or E1	• HDLC framing only per channel (default) • HDLC framing + SS7 low-level processing per channel • AAL5 for NNI signaling	rpq_wanm.rel Multiplexed WAN Driver

Table 1-4 contains a summary of supported protocols. For details, see the page indicated for the appropriate protocol.

Table 1-4. Summary of supported protocols

AIB/PMC	Definition	Selected	See Page
HDLC	Stands for High-level Data Link Control and is governed by the *ISO3309* specifications.	Default for the Serial and Multiplexed WAN drivers.	*7*
Bisynchronous	Sends and receives messages in bisynchronous format.	Selected by way of W_SETLINE to the Serial WAN driver.	*8*
SS7 (Signaling System Number 7)	Defines a set of protocols used by the telecommunications industry to provide a way for the transfer of signaling messages between telecom network nodes and exchanges.	Selected when either the Serial or Multiplexed WAN driver is loaded and WAN-ACTSS7 with W_SS7_START is issued on the opened stream.	*9*
ATM (Asynchronous Transfer Mode)	Uses asynchronous time division multiplexing technique to multiplex information flow in fixed blocks called *cells*.	Selected by loading the Multiplexed WAN driver.	*185*

2 Protocol descriptions

This chapter briefly describes the protocols implemented by the WAN driver. A full description is beyond the scope of this book. Refer to the appropriate standards documents for a complete description (see *Reference publications* on page *xiv* for a list of the standards documents).

HDLC framing

The framing structure for High-level Data Link Control (HDLC) is described in the *ISO 3309* document. The frame can be broken down as follows:

Table 2-1. HDLC Framing

Flag	Data or Information	Frame Check Sequence (FCS)	Flag
01111110 binary	Varying number of bits	16 bits	01111110 binary

`Flag`
: All frames start and end with the flag sequence which provides for frame synchronization. A single flag can be used as both the closing flag for one frame and the opening flag for the next frame. The flag value in hexadecimal is 7E.

`Data or Information`
: This can be any sequence of bits.

`Transparency`
: The transmitter inserts a "0" bit after all sequences of 5 contiguous "1" bits of the Data and Frame Check Sequence (FCS) to ensure that a flag sequence is not simulated. The receiver examines the data and FCS field, and discards any "0" bit that directly follows 5 contiguous "1" bits.

`Frame Check Sequence(FCS)`
: The FCS is 16 bits long and generates an FCS based on a polynomial, X**16 + X**12 + X**5 + 1. All bits involved in the data field are used for FCS. Bits inserted for transparency are not included in this calculation. The WAN driver also supports other types of FCS that are selectable in the serial mode of the driver using the W_SETLINE command described in *W_SETLINE — Define line characteristics* on page 209.

`Aborted Frame`
: A frame that ends with a "1" bit sequence of seven or more bits is considered to be an aborted frame.

Bisynchronous protocol

Support for both normal and transparent operation is provided. Both EBCDIC and ASCII text messages can be sent and received. For ASCII data, the WAN driver sends and receives 7-bit data with odd parity. Transmit data will be converted to odd parity. The parity bit will be stripped from received data.

The WAN driver performs low-level BISYNC message-type determination on received data. BISYNC messages without errors are parsed. The BISYNC message type is returned in the M_PROTO header block that accompanies the received data. See page 62 for a description of all BISYNC received message types.

For control frames that contain only control characters, the message type is returned, and no data is transferred from the WAN driver. For example, if an ACK0 was received, the message type WC_ACK0 is returned in the M_PROTO header block, and the received data pointer is null. If a receive error occurs, the error status is reported using wan_notify if indicated in the w_setline w_notifymask, and the message is thrown away.

The application must format its own BISYNC messages for transmission, including beginning and ending control characters, to send to the WAN driver. The application does not have to add leading or imbedded SYNs to a transmit message because the RadiSys ARTIC adapter will add leading and imbedded SYNs to all transmitted messages. The RadiSys ARTIC adapter will append the correct frame check sequence and/or pad where necessary to transmitted messages.

To send transparent data, set WC_BSC_TRANSP along with WC_TX in the M_PROTO header block. The WAN driver will insert beginning, ending, and imbedded DLEs; therefore, the application should not insert DLEs before beginning and ending control characters or within the data.

For more information about BISYNC framing, refer to the *IBM General Information — Binary Synchronous Communications* book.

Table 2-2 contains examples of how BISYNC messages are formatted.

Table 2-2. Valid BISYNC message types

Message Type	Actual Data	
	ASCII	**EBCDIC**
ACK0	10H,30H	10H,70H
ACK1	10H,31H	10H,61H
WACK	10H,3BH	10H,6BH
RVI	10H,3CH	10H,7CH
EOT	04H	37H
NAK	15H	3DH
ENQ	05H	2DH
D D D ENQ	D,D,D,05H	D,D,D,2DH
D D D ACK0	D,D,D,10H,30H	D,D,D,10H,70H
D D D ACK1	D,D,D,10H,31H	D,D,D,10H,61H
D D D NAK	D,D,D,15H	D,D,D,3DH
Non-transparent		
STX D D D ETX	02H,D,D,D,03H	02H,D,D,D,03H
STX D D D ITB	02H,D,D,D,1FH	02H,D,D,D,1FH
STX D D D ETB	02H,D,D,D,17H	02H,D,D,D,26H
STX D D D ENQ	02H,D,D,D,05H	02H,D,D,D,2DH
SOH D D D ITB	01H,D,D,D,1FH	01H,D,D,D,1FH
SOH D D D ETB	01H,D,D,D,17H	01H,D,D,D,26H
SOH D D D ENQ	01H,D,D,D,05H	01H,D,D,D,2DH
SOH D STX D D D ETX	01H,D,02H,D,D,D,03H	01H,D,02H,D,D,D,03H
SOH D STX D D D ITB	01H,D,02H,D,D,D,1FH	01H,D,02H,D,D,D,1FH
SOH D STX D D D ETB	01H,D,02H,D,D,D,17H	01H,D,02H,D,D,D,26H
SOH D STX D D D ENQ	01H,D,02H,D,D,D,05H	01H,D,02H,D,D,D,2DH
Transparent		
DLE STX D D D DLE ETX	10H,02H,D,D,D,10H,03H	10H,02H,D,D,D,10H,03H
DLE STX D D D DLE ITB	10H,02H,D,D,D,10H,1FH	10H,02H,D,D,D,10H,1FH
SOH D DLE STX D D DLE ETX	01H,D,10H,02H,D,D,10H,03H	01H,D,10H,02H,D,D,10H,03H
SOH D DLE STX D D DLE ITB	01H,D,10H,02H,D,D,10H,1FH	01H,D,10H,02H,D,D,10H,1FH
SOH D DLE STX D D DLE ETB	01H,D,10H,02H,D,D,10H,17H	01H,D,10H,02H,D,D,10H,26H
SOH D DLE STX D D DLE ENQ	01H,D,10H,02H,D,D,10H,05H	01H,D,10H,02H,D,D,10H,2DH

D = Data

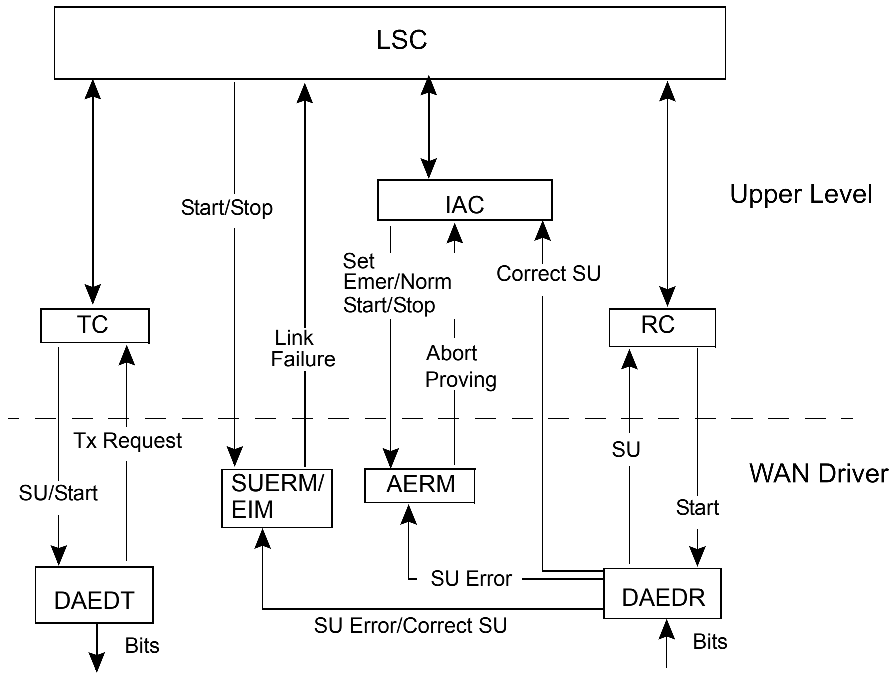
SS7 low-level processing

SS7 (Signaling System Number 7) is a dedicated digital network for performing call control. The SS7 protocol is divided into functional blocks, referred to as *levels*, that are similar to the 7-layer model Open System Interconnect (OSI) defined by the International Standards Organization (ISO). These protocols are defined by the International Telecommunication Union (ITU) and Bellcore. The WAN driver implements Message Transfer Part 1 (MTP1) and some parts of Message Transfer Part 2 (MTP2). Refer to *ITU-T Publications Q.700, Q.701 and Q.703* for a detailed description.

WAN driver in relation to MTP2

The following describes the WAN driver in relation to MTP2 as described by ITU-T Q.703, ANSI to 111-3, and TTC SS7.

Figure 2-1. WAN driver in relation to MTP2



M	Alignment Error Rate Monitor	LSC	Link State Control
DR	Delimitation, Alignment, Error Detection for receive	RC	Reception Control
		SUERM	Signal Unit Error Rate Monitor
DT	Delimitation, Alignment, Error Detection for transmit	TC	Transmit Control
	Initial Alignment Control	SU	Signal Unit or a Frame

The WAN driver implements DAEDR, DAEDT and SUERM as described by the *ITU-T Q.703* specifications. Due to the split nature of the WAN driver and MTP2, the implementation of AERM differs slightly from that described in the *ITU-T Q.703* specifications. These deviations are described in *Error Rate Monitor (ERM)* on page *17* and *WAN_NOTIFSS7 — Notify SS7 status* on page *107*. In addition, the receiver performs SU filtering and the transmitter performs automatic generation of some SUs.

Special SS7 features

SS7 has a unique data link protocol (called level 2 of the Message Transfer Part or MTP2) based on HDLC, which requires the continuous presence of frames on the link. Frames are thus back-to-back. In this way, the MTP can be informed immediately of an upcoming link failure (as soon as erroneous frames or the absence of frames is discovered).

In SS7 terminology, an HDLC frame is called a *Signal Unit* (SU). SUs are classified in three categories:

- Fill In Signal Unit (FISU) of length 5 bytes including FCS
- Link Status Signal Unit (LSSU) of length 6-7 bytes including FCS
- Message Signal Unit (MSU) of length 8-278 bytes including FCS

Special TTC SS7 features

The TTC SS7 standard (the Japanese version of SS7) sends MSUs (user data) without any restrictions (they can be sent back-to-back). If there are no MSUs to send, then FISUs are sent at a specific interval. This interval period is configurable. During alignment, LSSUs are transmitted at specific intervals only. These interval periods are configurable.

The reception is the same as the ITU-T/ANSI standards.

Conditions that activate the *Octet Counting Mode* (OCM) are:

- Too long of a frame, or
- HDLC abort

The conditions are the same as the ITU-T/ANSI standards and actions taken are configurable. If OCM is disabled, every 16 octets do not generate *SU in error* to the ERMs. However, the conditions that activated OCM are treated as *SU in error.* If OCM is enabled, OCM logic works in a manner similar to the ITU-T/ANSI standard.

TTC SS7 runs AERM and SUERM in emergency mode and both have the optional support of a timer, which, if enabled, is used to control when the respective error counters are incremented.

SS7 SU Reception (DAEDR)

The WAN driver implements the DAEDR requirements as described by the flowcharts in the *ITU-T Q.703* specifications. These requirements are summarized as follows:

SU format requirements that are the same as HDLC

- Data must be surrounded by opening and closing flags. The bit pattern of the flag is 01111110.
- Data must finish with 16 check bits (the 16-bit CRC-CCITT) for error detection.
- Inserted 0 bits in the data (to prevent seven consecutive 1's, that is, 1111111) must be detected and removed.

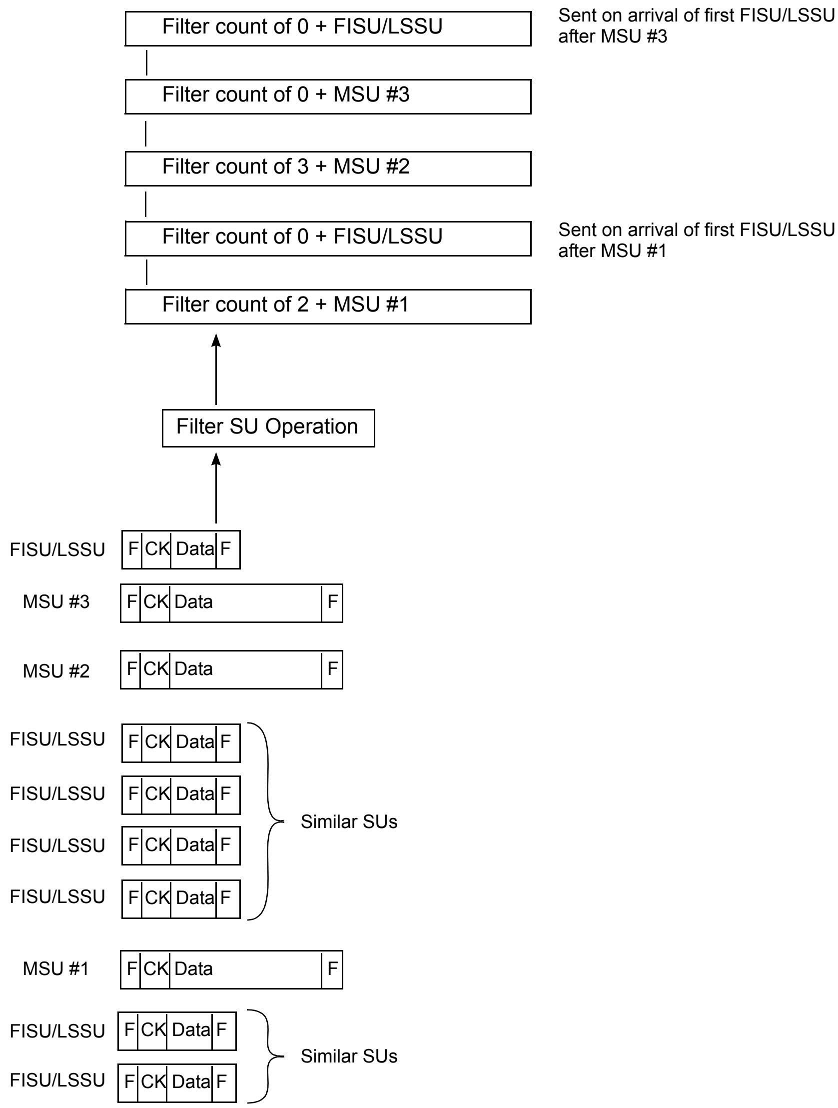
SU format requirements that are unique to SS7

- Frames of less than 5 octets (not counting the flags but including the FCS) are discarded and reported to the ERM.
- Frames of more than 278 octets trigger a special mode called the *Octet Counting Mode* (OCM).
- Loss of alignment (seven consecutive ones in the data) also triggers the OCM.
- While in OCM, erroneous frames and the number of incoming octets affect the counters used by the ERM. See *Error Rate Monitor (ERM)* on page 17 for more details.
- The ERM logic needs to track all frames.

SU filtering

Because SS7 requires the continuous presence of frames, there is a series of similar FISUs or LSSUs that are 5 to 7 octets long. Therefore, the WAN driver performs filtering of similar FISUs or LSSUs that are 5 to 7 octets in length (including FCS). The first different SU following a series of similar SUs are preceded with the number of SUs that were discarded because of filtering. *Figure 2-2* explains the filtering mechanism.

Figure 2-2. SU filtering



Again, filtering applies only to SUs that are 5 to 7 octets long. The filtering mechanism keeps a count of all good SUs because the number of discarded SUs due to filtering must be relayed to the upper level. This count is provided in the M_DATA block that contains the new SU. See *WAN_DAT — Data messages for transmission and reception* on page 61 for details.

Use the SS7_FILTER_COUNT command-line parameter, described on page *238*, to specify the number of duplicate SUs, 5 to 7 octets long, that will not be filtered and will be passed upstream. The example shown in *Figure 2-2* on page *13* shows the default case with the parameter set to 0 (zero). If this parameter is set to 1, one duplicate SU is sent, resulting in two identical SUs being sent upstream.

A special *reset filtering* (*WAN_RESETSS7 — Reset filtering operation* on page *109*) request from the upper level allows the interruption of the filtering mechanism for one SU. Thus, after the WAN driver receives a Reset Filtering request, the SU currently being filtered is sent "up." As usual, this SU is preceded by the filter count. See *WAN_DAT — Data messages for transmission and reception* on page *61* for details.

SU transmission (DAEDT)

The WAN driver implements the DAEDT requirements as described by the flowcharts in the *ITU-T Q.703* specifications. In addition to these flowcharts, the WAN driver performs the automatic generation of certain SUs, described as follows.

SS7 requires the continuous presence of SUs on the signaling link. The WAN driver therefore automatically generates certain SUs without the involvement of the upper layer. To perform this task, the driver must always keep the first two octets of the previously transmitted SU. These two octets hold the following:

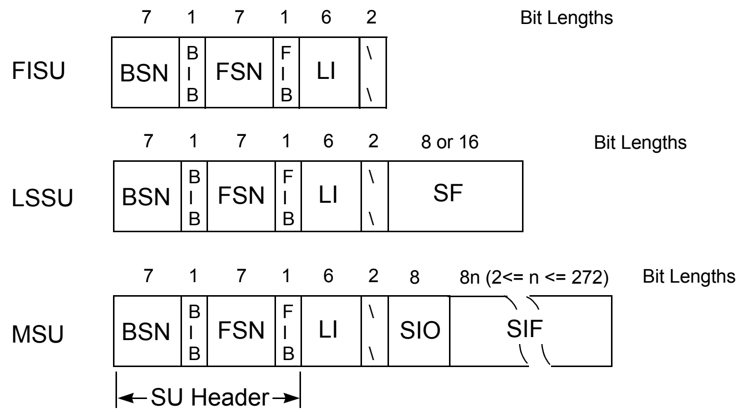
- Backward Sequence Number (BSN)—7 bits
- Backward Indicator Bit (BIB)—1 bit
- Forward Sequence Number (FSN)—7 bits
- Forward Indicator Bit (FIB)—1 bit

These two octets are used to keep messages in sequence, to acknowledge properly received SUs, and to request the retransmission of corrupted SUs. All MTP2 frames start with the BSN, BIB, FSN, and FIB. This group of fields is called the *SU Header* (SUH).

The SUH of the last transmitted SU is stored in the LSUH (L for Last).

Figure 2-3 shows the location of the SUH in each type of SU.

Figure 2-3. Format for each type of Signal Unit



BIB	Backward Indicator Bit	LSUH	Last SU Header
BSN	Backward Sequence Number	MSU	Message Signal Unit
FIB	Forward Indicator Bit	SF	Status Field
FISU	Fill-in Signal Unit	SIF	Signaling Information Field
FSN	Forward Sequence Number	SIO	Service Information Octet
LI	Length Indicator	SUH	SU Header
LSSU	Link Status Signal Unit		

When no SU is available for transmission, the general rule is to continuously transmit a FISU constructed with the LSUH. This rule is superseded when the last transmitted SU is an LSSU. This LSSU is continuously transmitted until another SU is pending for transmission. The exception to this rule is the “Busy” LSSU (called SIB, with SF=0x5), which is transmitted only once. In addition to the LSUH, the WAN driver tracks LSSU retransmission with:

- The LSSU Retransmission Flag (LSSURT)
- The Current Status Field (CSF)

Initially, when transmission starts, the LSUH is 0xFFFF, LSSURT is false, and CSF is 0x0000.

Transmission logic

When transmission of an SU is scheduled, the logic is as follows:

```
If an MSU is the next SU to transmit
    Transmit the MSU
    Obtain the LSUH from the MSU
    LSSURT = false

  Else, if a FISU is the next SU to transmit
    Transmit the FISU
    Obtain the LSUH from the FISU
    LSSURT = false

  Else, if a LSSU is the next SU to transmit
    Transmit the LSSU
    Obtain the LSUH from the LSSU
    If LSSU < > SIB
      LSSURT = true
      Get the CSF from the LSSU
    Else
      LSSURT = false

  Else, there is no SU available for transmission
    If LSSURT
      Transmit a LSSU with the LSUH and the CSF
    Else
      Transmit a FISU with the LSUH
```

Standard HDLC processing is applied on the outgoing frames:

- Data finishes with the 16 check bits (the 16-bit CRC-CCITT) for error detection.
- A 0 is inserted after every sequence of five consecutive 1’s (to ensure that the HDLC flag is not imitated by the data).
- The resulting frame is surrounded by opening and closing flags. The bit pattern of the flag is 01111110.

Error Rate Monitor (ERM)

Depending on the state of the SS7 signaling link, Error Rate Monitor (ERM) is of these two forms:

- If the signaling link is being aligned (not the same as frame alignment), the Alignment ERM (AERM) is active.
- In the normal state, the Signal Unit ERM (SUERM) is active.

The ERM gets indications from frame processing on the occurrence of erroneous and valid SUs. It does not need to look into the SU data. Each type of ERM keeps a counter:

- Cs for SUERM
- Ca for AERM

The active counter is incremented or decremented, and when it reaches a certain threshold, the Link Failure or Abort Proving indication is sent to the upper level. The upper level controls the reset of the counters and must select which counter is active. Link alignment (with the Ca counter) also has the notion of normal versus emergency alignment.

Implementation of SUERM for SS7

The WAN driver implements SUERM based on the flowcharts in the ITU-T Q.703 specifications. An error counter is used to determine if the link has failed. The error counter is incremented by one after each bad SU is received. This error counter is decremented after a window of 256 SUs are received.

Implementation of SUERM for TTC SS7

The WAN driver implementation of SUERM for the TTC SS7 version consists of the optional use of a timer, which is used to determine if the link has failed.

If the timer is enabled for TTC SS7 SUERM, the expiration of the timer causes the decrement of the error counter by one if the last SU received was good. Otherwise, the counter is incremented by w_param_D (the default is 16).

If the timer is not enabled for TTC SS7 SUERM, a bad SU causes the error counter to be incremented by the w_param_D and a good SU causes the error counter to be decremented by one until it reaches zero.

Implementation of AERM for SS7 and TTC SS7

For both the SS7 and the TTC SS7 versions, the AERM differs from ITU-T/ANSI standards in the following ways:

- AERM does not stop automatically when Ca reaches its threshold (Tin or Tie). It issues Abort Proving, resets Ca to zero, and reenters Monitoring state. After AERM is started, it can be stopped by MTP2 only when an explicit Stop AERM request is issued.
- In the Monitoring state, the AERM accepts Set Ti to Tin and Set Ti to Tie input requests.
- Set Ti to Tin and Set Ti to Tie requests reset Ca to zero

The previous changes are necessary in order to avoid a small window where no ERM is active when SIN or SIE are being received. With these modifications, the MTP2 starts the AERM when the alignment procedure is started. *Table 2-3* describes the logic behind each ERM counter. Note that the thresholds are programmable.

Table 2-3. ERM summary

Counters	Incremented when:	Decremented when:	Thresholds	Event triggered
Cs for SUERM	SU in error received	256 SUs (correct or incorrect) received	T = 64	Link Failure
Ca for AERM	SU in error received	Never	• Tin = 4 for normal alignment • Tie = 1 for emergency alignment	Abort proving

Clear Channel Capability Mode

To ensure your adapter supports this mode, contact your RadiSys representative.

The Clear Channel Capability mode supports enhanced MTP2 functions and procedures that are suitable for the operation and control of signalling links at data rates of 1.5 Mbit/s (T1) and 2.0 Mbit/s (E1) as a national option. Refer to the *ITU-T Recommendation Q.703 Annex A and Bellcore GR246* for a detailed description.

The Multiplexed WAN driver supports Clear Channel Capability. Use the WAN_ACTSS7 service message to activate or deactivate Clear Channel Capability mode and to start or stop the *Errored Interval Monitor (ERM)*. If a link failure occurs during EIM monitoring, WAN_ACTSS7 must be used to restart the EIM. See *WAN_ACTSS7 — Control SS7 features* on page 105 for information.

Physical layer

MTP2 messages are directly mapped over T1 or E1 frame structures. The messages are generated and extracted out of a specific set of channels. A maximum of 64 channels per PMC can be used for Clear Channel Capability operation. However, there may be performance-related restrictions.

Use the WAN driver W_SET_PHY_PIPE management command to assign the time slots that make up a physical stream by specifying the w_phy_pipe parameter and the w_options field with the option W_SS7_MODE, described in *W_SET_PHY_PIPE — Define and undefine time slots* on page 140.

The WAN driver assumes that timeslots used for Clear Channel Capability run at 64 Kbps.

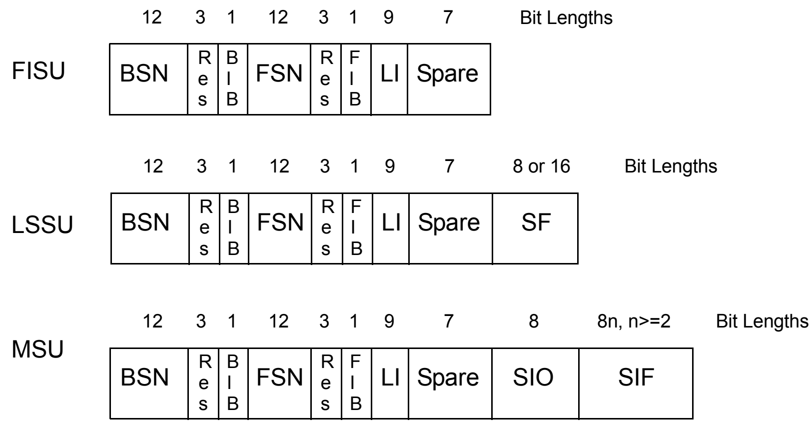
LSSU/FISU/MSU length indicator/sequence numbering

Clear Channel Capability defines an optional extended sequence number format that is 12 bits long. If the extended sequence number format is used:

- The MTP2 Forward Sequence Number (FSN) and Backward Sequence Number (BSN) increase from 7 to 12 bits, providing a cyclic sequence from 0 to 4095.
- The length indicator (LI) increases from 6 to 9 bits and supports messages up to 273 octets. The check for the correct signal unit length is increased by three octets. A length indicator that does not match the message octet count is treated as an SU in error condition. The maximum frame size is 279 octets. The length indicator is in network (big endian) byte order.

Use the WAN driver W_SETSS7_CCC management command to select extended sequence number format. See *W_SETSS7_CCC — Set SS7 Clear Channel Capability configuration parameters* on page 122 for information.

Figure 2-4. 1.5 and 2.0 Mbit/s rate format for each type of Signal Unit



BIB	Backward Indicator Bit	LSUH	Last SU Header
BSN	Backward Sequence Number	MSU	Message Signal Unit
FIB	Forward Indicator Bit	SF	Status Field
FISU	Fill-in Signal Unit	SIF	Signaling Information Field
FSN	Forward Sequence Number	SIO	Service Information Octet
LI	Length Indicator	SUH	SU Header
LSSU	Link Status Signal Unit		

Acceptance of alignment

For Clear Channel Capability, the EIM is applied instead of the SUERM. Octet Counting Mode (OCM) is not used for EIM. However, OCM may be used for Alignment Error Rate Monitor (AERM), which is operational during normal and emergency proving periods.

Error monitoring

The EIM has as its function the estimation of signalling link fault conditions by monitoring errors over a prescribed interval to model the queue buildup on the transmitting end. An interval is errored if one or more SUs are rejected by the acceptance procedure or if a flag is lost. The four fields that determine the EIM are:

- w_ccc_Te — The number of intervals where SUs have been received in error that will cause an error rate high indication to level 3, TE (intervals).
- w_ccc_Ue — The constant UE for incrementing the counter.
- w_ccc_De — The constant DE for decrementing the counter
- w_ccc_T8 — Timer T8, the interval for monitoring errors

The EIM is implemented in the form of an up and down counter:

- Decremented at a fixed rate DE for every interval where no SU is errored, but not below zero, and
- Incremented at a fixed rate UE for every interval where one or more SU errors are detected by the SU acceptance procedure, or where no flag is received but not above threshold.

An excessive error rate shall be indicated whenever the threshold is reached.

The OCM, which provides an estimate of an SU, is not used for the EIM because this procedure is not based on an accounting of individual errors.

When the link is brought into service, the monitor count will start from zero.

For Clear Channel Capability operation, the WAN driver management commands can be used as follows:

- W_SETSS7_CCC — to specify AERM and EIM counter thresholds and the EIM monitoring interval. See *W_SETSS7_CCC — Set SS7 Clear Channel Capability configuration parameters* on page *122* for more information.
- W_GETSS7_CCC — to obtain the type of ERM currently in operation and the ERM counter values for Clear Channel Capability operation. See *W_GETSS7_CCC — Get SS7 Clear Channel Capability configuration parameters* on page *125* for more information.

T1/E1/J1 interface

The T1/E1/J1 interface (hereinafter referred to as *T1/E1*) is capable of providing various alarms, statistics, and data link messaging capabilities. The standards that govern these are as follows:

- *General Aspects of Digital Transmission Systems, ITU-T G.704*
- *General Aspects of Digital Transmission Systems, LOS and AIS defect detection and clearance criteria ITU-T G.775*
- *Primary Rate User-Network Interface-Layer 1 Specifications ITU-T I.431*
- *RFC 1406, Definitions of Managed Objects for DS1 and E1 Interfaces Types, Trunk MIB Working Group*

J1 standards are similar to T1.

Depending on the application, you need to report certain alarm conditions as *disconnects* as soon as they occur. You can choose which ones generate *disconnects*.

SS7 signaling links do not impose specific use of the T1/E1 capabilities (it is up to the SS7 network operator). The WAN driver must thus allow complete control and monitoring of the T1/E1 capabilities. *Table 2-4* shows the T1/E1 features that are accessible from the Multiplexed WAN driver. Terms that are separated by slashes (/) are equivalent.

Table 2-4. T1/E1 available features (Part 1 of 2)

Attributes	T1 (J1 is similar to T1)	E1
Code	• AMI • B8ZS	• AMI • HDB3
Framing	• Super Frame (SF) / 12-frame multiframe / D4 • Extended Super Frame (ESF) / 24-frame multiframe	• Double Frame (DF) • Multiframe (MF) with/without CRC-4
Signaling support	• Channel Associated signaling (CAS)/Robbed-bit signaling not implemented • Common Channel signaling (CCS) not implemented • 4-Kbps Data Link (DL) of ESF not implemented except for RAI and idle code	• TS16 signaling (CAS or CCS) not implemented
CRC Reception and Generation	• CRC-6 Optional for ESF	• CRC-4 optional for MF

Table 2-4. T1/E1 available features (Part 2 of 2)

<table>
<tr><th>Attributes</th><th>T1 (J1 is similar to T1)</th><th>E1</th></tr>
<tr><td rowspan="2">Remote Alarm Indication (RAI) / Yellow Alarm</td><td colspan="2">• Reported when received
• Near-end transmits RAI on Loss Of Frame (LOF) failure</td></tr>
<tr><td>• For SF, signaled in F-bit of twelfth frame or when b2=0 in all channels
• For ESF, signaled with 1111111100000000 pattern in DL
• Near-end transmits RAI on Loss Of Frame (LOF) failure</td><td>• Signaled in TS0 of every other frame</td></tr>
<tr><td>Alarm Indication Signal (AIS) / Blue Alarm</td><td colspan="2">• Reported when received
• When all 1's are received</td></tr>
<tr><td>Loss Of Signal (LOS) / Red Alarm</td><td colspan="2">• Reported when received</td></tr>
<tr><td>Loss Of Frame (LOF) Alignment</td><td colspan="2">• Increments a counter
• Automatically generates a RAI to the far end</td></tr>
<tr><td>Available counters</td><td colspan="2">• Framing errors
• Code violations
• Errored seconds</td></tr>
<tr><td></td><td>• CRC errors for ESF</td><td>• CRC errors for MF
• E-bit errors</td></tr>
<tr><td>Loopback Modes</td><td colspan="2">• Payload — Rx to Tx with framing generated
• Remote — Rx to Tx including framing signal
• Local — Tx back to Rx
• Channel — Tx back to Rx on a channel basis</td></tr>
<tr><td>Chaining</td><td colspan="2">• Entire port can be chained to another port, all channels are connected to their equivalent channel on the other port
• Individual channel can be connected to another channel on the same or different port</td></tr>
<tr><td>Shorts</td><td colspan="2">• Reported when received
• Transmit Line Short (significant only if ternary line interface is used)
• Transmit Line Open</td></tr>
<tr><td rowspan="2">Monitor Mode</td><td colspan="2">To monitor T1/E1 data and HDLC or SS7 traffic of the T1/E1 lines, load the multiplexed WAN driver with the W_MONITOR_MODE=YES command line parameter. In monitor mode:
• Transmitters of all T1/E1 ports are tri-stated and the receiver sensitivity is increased to detect an incoming signal of -20 db resistive attenuation.
• The WAN driver sets the Loss of Signal (LOS) detection limit to 0.16v in short haul mode and 0.10v in long haul mode.</td></tr>
<tr><td>• In T1 mode, the application can control this receiver sensitivity by setting w_signal_mode to the appropriate values when the W_SETDI_PORT ioctl is issued.</td><td></td></tr>
</table>

SC-bus implementation

The ARTIC960 4-Port T1/E1 Mezzanine Card provides an SC-bus connector so that one can connect to other ARTIC960 4-Port T1/E1 Mezzanine Cards or adapters from other vendors over the SC bus.

The SC bus consists of a 16-wire Time Division Multiplexed (TDM) data bus and a message channel for control and signaling. Currently there exists a standard for communicating between adapters in a universal way, *the SCSA architecture*. This architecture is composed of two parts:

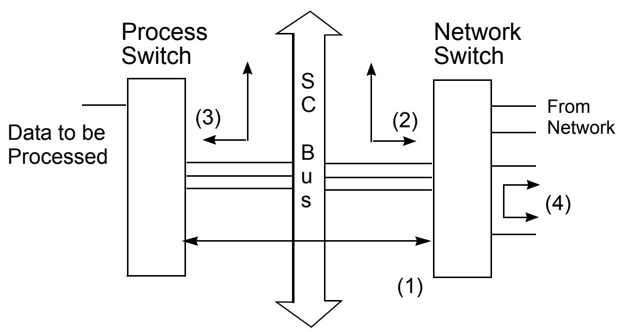
- Software model — *SCSA Telephony Application Object Framework*
- Hardware model — *SCSA*

The WAN driver specification provides support for the SCSA hardware model. The optional messaging channel is not implemented.

SC-bus programming support

The ARTIC960 4-Port T1/E1 Mezzanine Card hardware switching support can be viewed as follows:

Figure 2-5. SC-bus switching support



Data paths are:

1. To and from network switch to processing switch.
2. To and from network switch to other boards connected via the SC bus
3. To and from other boards connected via SC bus to process switch
4. To and from network port

CT-bus implementation

The ARTIC 4-Port T1/E1/J1 DSP PMC provides a CT-bus connector so that one can connect to other ARTIC 4-Port T1/E1/J1 DSP PMCs or adapters from other vendors over the CT bus.

The CT bus is implemented with H.100 or H.110 variants. These are industry standard real-time TDM buses for computer telephony and conform to the Enterprise Computer Telephony Forum (ECTF) standard bus for interoperable computer telephone (CT) systems. The CT bus consists of 32 synchronous serial lines that can be programmed to run at three bit rates, each for 32, 64, or 128 timeslots per line:

- 2.048 Mbps — yields 1024 total timeslots
- 4.096 Mbps — yields 2048 total timeslots
- 8.192 Mbps — yields 4096 total timeslots

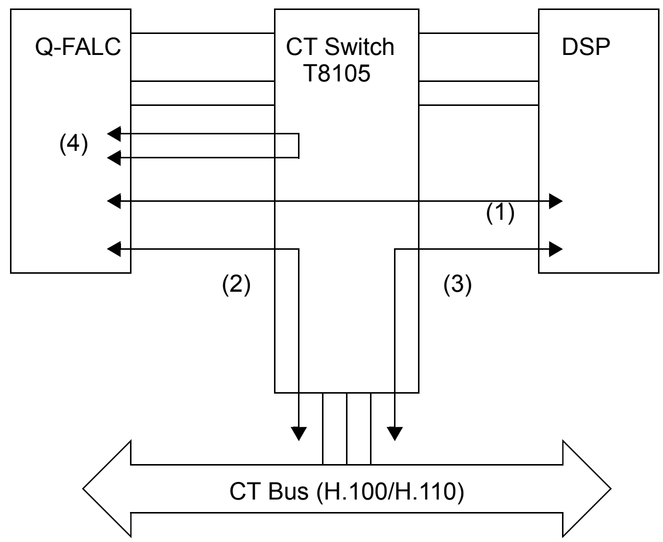
The H.100 bus is used when the PMC is used as a PCI-bus daughter board. A ribbon cable connector on the PMC adapter is used to connect all the CT devices.

The H.110 bus is used when the PMC is configured in a Compact PCI system where the H.110 bus resides in the CompactPCI motherboard and is common with all other CompactPCI adapters using the main cPCI bus.

CT-bus programming support

ARTIC 4-port T1/E1/J1 DSP PMC hardware switching support can be viewed as follows:

Figure 2-6. CT-bus switching support



Data paths are:

1. To and from T1/E1 network switch to DSP processor.
2. To and from T1/E1 network to other boards connected via the CT bus
3. To and from other boards connected via CT bus.
4. To and from T1/E1 network port

ATM in SS7 environments

ATM (Asynchronous Transfer Mode) is a packet-oriented transfer mode and uses asynchronous time division multiplexing technique to multiplex information flow in fixed blocks called *cells*.

In a B-ISDN transport network, ATM is the transfer mode of choice to achieve higher speeds in the SS7 signaling environment. Signaling link functions are provided by the Signaling ATM Adaptation Layer (SAAL).

Implementing higher speeds for signaling is also referred to as High-speed Signaling Link (HSL).

The WAN driver provides support for higher speeds (see *Physical layer* on page *28* for rates).

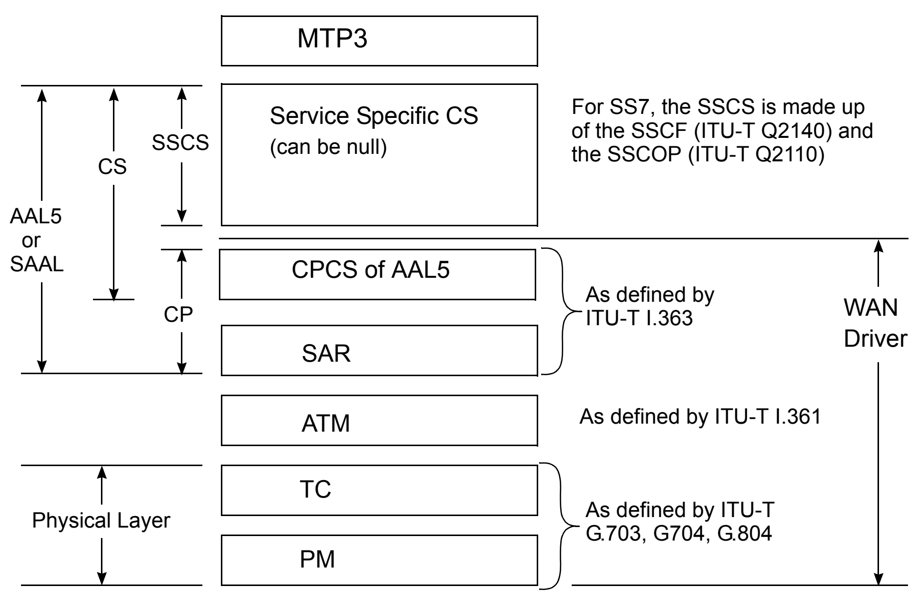
The WAN driver implements:

- Parts of the ATM Adaptation Layer 5 (AAL5) Protocol stack that are generic in nature so that it can be used in SS7 and other environments
- The NNI (Network Node Interface) format for the ATM layer.

AAL5 protocol reference model

The following shows a model of the AAL5 protocol.

Figure 2-7. AAL5 protocol reference model



AAL5	ATM Adaptation Layer 5	SAAL	Signaling ATM Adaptation Layer
CPCS	Common Part Convergence Sublayer	SAR	Segmentation and Reassembly
CP	Common Part	SSCF	Service Specific Coordination Function
CS	Convergence Sublayer	SSCOP	Service Specific Connection Oriented Protocol
MTP3	Message Transfer Part 3	TC	Transmission Convergence
NNI	Network Node Interface		
PM	Physical Medium		

Physical layer

The physical layer provides a means for transporting ATM cells. The following rates (or a fraction thereof) are the physical rates that can be achieved with the WAN driver.

- T1 — 1,544,000 bps
- E1 — 2,048,000 bps

Fractional rates are achieved by combining time slots of the T1 or E1 links.

The WAN driver is capable of combining time slots from the SC-bus. When using fractional rates it is possible to have multiple ATM cell streams. ATM cells are directly mapped into a DS1 or E1 frame. Refer to the *ITU-T G.804* specification for further details.

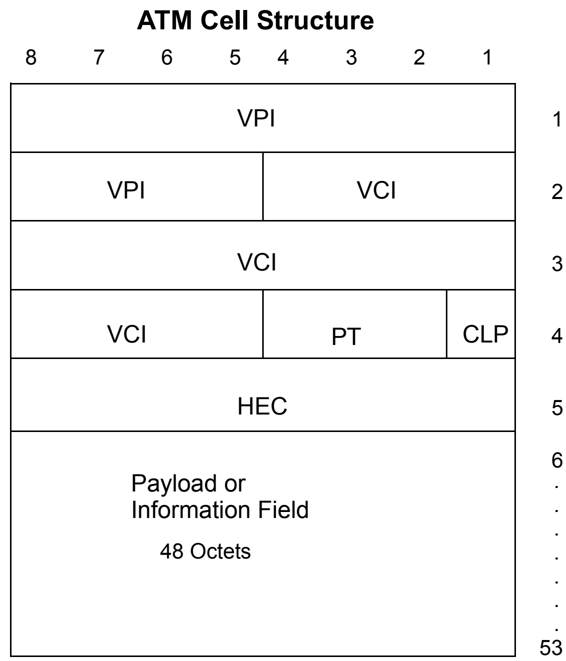
ATM layer

ATM is a specific packet-oriented transfer mode using an asynchronous time division multiplexing technique. ATM provides high efficiency and flexibility as it provides virtual channels instead of dedicated physical channels.

The multiplexed information is organized in a fixed-size block called a *cell*. A cell is 53 bytes in length and it consists of a 5-byte header and a 48-byte payload. Cells originating from a connection end point are delivered at the destination end point in the same order they were originated, hence providing cell sequence integrity.

ATM cells are labeled, using the Virtual Path Identifier (VPI) and Virtual Channel Identifier (VCI) fields. These fields are part of the ATM cell header. These fields provide a way for routing cells through the ATM network. Refer to the *ITU-T I.361* specification for details. *Figure 2-8* shows an ATM cell at an NNI.

Figure 2-8. ATM cell at an NNI



CLP	Cell Loss Priority	VCI	Virtual Channel Identifier
HEC	Header Error Checksum	VPI	Virtual Path Identifier
PT	Payload Type		

ATM Adaptation Layer 5 (AAL5)

The ATM Adaptation Layer 5 (AAL5) enhances the services provided by the ATM Layer to support the functions required by the next higher layer (for example, signaling). The AAL5 consists of the Common Part (CP) and the Service Specific Convergence Sublayer (SSCS). Two modes of services are defined: Message and Streaming mode. See the *ITU-T I.363* specification for a description of these modes.

Common Part (CP)

CP consists of these layers:

- Common Part Convergence Sublayer (CPCS):
  - Receives a variable length frame from its upper layer (1–65535 bytes in length) and pads this frame so that the total length becomes an integral multiple of 48 (the ATM cell payload length).
  - Provides a CRC-32 function to detect errors.
- Segmentation and Reassembly (SAR):
  - Accepts a frame whose length is a multiple of 48
  - Maps the frame into multiple ATM cell payloads
  - Provides a way to identify the begin and end using the payload type field.

Service Specific Convergence Sublayer (SSCS)

Different SSCS protocols to support specific AAL user services or groups of services have been defined. The SSCS may be NULL. For the SS7 signaling environment, SSCS has been broken down into two parts:

- Service Specific Connection Oriented Protocol (SSCOP) — A connection-oriented protocol with error recovery and reliable data transfer services. Refer to the *ITU-T Q.2110* specification for details.
- Service Specific Coordination Function (SSCF) — Maps the services of SSCOP to the requirements of MTP Level 3. For signaling, two types of SSCF are defined at the User-to-Network Interface (UNI) or the Network Node Interface (NNI). Refer to the *ITU-T Q.2140* specification for details on SSCF at NNI.

Operation and Maintenance (OAM)

The *ITU-T* specifications describe how to operate and maintain the physical layer and the ATM layer to provide for:

- Fault management
- Performance management
- Activation/Deactivation of procedures
- System management for end systems.

This activity is done using special Operation and Maintenance (OAM) ATM cells. The WAN driver performs some of the previously mentioned functions at the Virtual Channel (VC) level and the rest can be performed by a separate STREAMS module residing on the base adapter.

3 WAN driver STREAMS interface

This chapter provides information about WAN driver configuration, creating STREAMS, and the types of STREAMS messages and commands.

About minor numbers

The WAN driver follows the UNIX paradigm for defining subdevices. For these devices, the minor numbers are used in the following manner:

The system configuration process defines special files called *device special files* in the UNIX file system. They usually represent a fixed profile to users. The system configuration process assigns fixed numbers, called *minor numbers*, which are passed to the driver when the device special file is opened. The process of opening such a special file is called *specific open* or *non-clone open* in this book.

The system configuration process also defines a wild card special file, called a *clone device*. When a *clone open* is done, the driver assigns the minor number for that open. In this case, the user can perform control functions to the driver or, in other cases, the user can operate the device like a normal device.

The WAN driver defines a variable number of minor numbers for specific opens. The maximum minor numbers is a configurable parameter. The *clone open* minor numbers are assigned after those for a *specific minor number open*. Both these numbers can be conveyed to the WAN driver at the load time through command line parameters. The maximum number for *specific open* could be zero, and also the number of *clone opens* could be made zero by setting the number of *specific opens* equal to the total maximum minor numbers.

Configuring the WAN driver

The WAN driver assumes a support of a configuration utility to get the hardware configuration and control information. The configuration activity can be performed on any stream (a stream is assigned by the system on any *open* to the WAN driver).

The number of logical channels, a stream where data transfer takes place, depends on the hardware and its capabilities. *Table 1-3, "Summary of supported hardware with ARTIC adapters,"* on page *4* lists the maximum number of logical channels for each hardware supported. The number of clone devices is software choice and depends on how the system is architected, and how many processes need to monitor or configure the driver, or both. The WAN driver supports various configuration choices in order to be able to work in various situations. The following sections describe the choices and the configuration steps involved in implementing them.

Creating STREAMS

The RadiSys ARTIC STREAMS environment supports two types of open(). They are defined as follows. (See *Figure 3-1* on page *34* for details.)

`non-clone open()`
: Opens the logical channel identified by the minor number. The WAN driver invokes the hardware open immediately. The SNID that will be associated with the stream is then referenced in subsequent management commands.

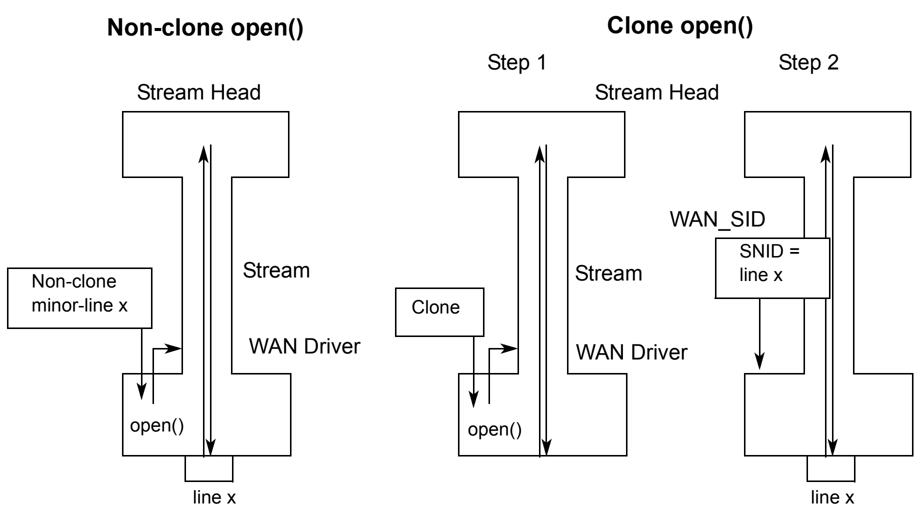
`clone open()`
: Allows the creation of a management path that is not associated with any logical channel and carries only management commands. The hardware open operation is not invoked.

  A WAN_SID message sent down on a clone stream binds it to a logical channel. Beyond that point, a clone stream is equivalent to a non-clone stream. Also, it ceases to be a management stream at that point.

  When opening a CLONE stream to the serial WAN using the SNID_DECODE=NO command line parameter, a W_SET_SNID command must be issued before a WAN_SID command. After the stream is closed, the W_SET_SNID command is still in effect. You can release the SNID by issuing the W_REL_SNID command.

The stream opening procedure differs from the *SpiderX25 WAN Implementation Guide, r8.0,* by Spider Systems.

Figure 3-1. Non-clone versus clone open



Non-clone open with SNID decode

In this mode, a logical channel is preassigned to a minor number. The mapping is:

```
Logical channel number = minor number + 1
```

Synchronous serial WAN driver: Opening the special file for minor number 0 allows you to operate the port number 1. When the WAN_SID message is sent down, the SNID decode must result in the port number for the minor number being opened, or else an error is generated for the WAN_SID service message.

Multiplexed WAN driver: This mode is not supported for this driver.

Non-clone open with no SNID decode

In this mode, a logical channel is preassigned to a minor number with the same mapping as shown in *Non-clone open with SNID decode* on page *35*. The only difference is that the WAN_SID message assigns an identifier only to the stream rather than selecting a port or a channel.

Synchronous serial WAN driver: The minor number also selects the corresponding port.

Multiplexed WAN driver: This mode is not supported for this driver.

Clone open with SNID decode

In this mode, the logical channels are *not* preassigned to minor numbers. The SNID in the WAN_SID message is decoded by the WAN driver to know:

Synchronous Serial WAN driver: Port number.

Multiplexed WAN driver: Physical port and channel number.

Clone open with no SNID decode

In this mode, all logical channels are *not* preassigned to minor numbers. But the configuration utility assigns SNIDs to logical channels when it configures them to the WAN driver by way of the W_SET_SNID command. The SNID in the WAN_SID message is searched by the WAN driver for the following:

Synchronous serial WAN driver: Port number.

Multiplexed WAN driver: Physical port and channel number.

When opening a CLONE stream to the WAN driver using the SNID_DECODE=NO command line parameter, a W_SET_SNID command must be issued before a WAN_SID command. After the stream is closed, the W_SET_SNID command is still in effect. You can choose to release the SNID by issuing the W_REL_SNID command.

Types of WAN driver STREAMS messages and commands

The STREAMS interface of the WAN driver is composed of the following types of messages:

- Service messages — M_PROTO, M_PCPROTO and M_DATA messages that control and provide the reception/transmission of frames for the line associated with the Stream.
- Management commands — M_IOCTL messages that allow management (parameters setting and statistics) of various lines with M_IOACK and M_IOCNAK as responses.
- Error messages — M_ERROR messages that the WAN driver responds with when errors are detected on a service message.

4 Serial and Multiplexed WAN drivers (command sequences)

This chapter lists the order for command sequences to the serial synchronous WAN driver running SS7 protocol and in HDLC framing mode. It also provides SC-bus connection scenarios.

Serial synchronous WAN driver running SS7 protocol

The following is the order in which the upper-level process should issue commands to the WAN driver. The calls made are standard STREAMS application interface calls.

1. Open a stream to the WAN driver using the open() STREAMS call.
2. Set the SNID (Subnetwork ID) for the port opened in step 1 by building an M_PROTO message using the wan_sid structure. Send this message on the opened stream using the putmsg() call. This message can be sent at a later time.

   If the driver was loaded with the SNID_DECODE=YES configuration parameter, the SNID identifies the physical port number.

   If the driver was loaded with the SNID_DECODE=NO configuration parameter, the W_SET_SNID command would associate the physical port number to the SNID.
3. Set the mode of the WAN driver to SS7 by sending the W_ACTSS7 service message. This command starts the SS7 function. Send this message on the opened stream using the putmsg() call.
4. Configure the characteristics of the WAN driver by sending the W_SETSS7 management command. This command sets the attributes of the SS7 link. This command is issued using an ioctl STREAMS call with the line parameters set in structure wan_setss7.
5. Set the line configuration parameters using the W_SETTUNE management command. This command is issued using an ioctl STREAMS call with the line parameters set in structure wan_tune.
6. Register with the WAN driver using the wan_reg structure encapsulated within an M_PROTO message. The wan_type field in the wan_reg structure should be set to WAN_REG. Use the putmsg STREAMS call to send this message to the WAN driver.

7. Once registration is completed, the WAN driver programs the hardware based on the options selected in the W_SETTUNE command. Depending on the interface used, it enables the output signals and checks for the input signals. If the signals are available, it sends up an M_PCPROTO message with the wan_command field set to WC_CONNECT and wan_status set to WAN_SUCCESS in the wan_ctl structure. This indicates to the upper layer that the WAN driver is ready for data transfer. The upper layer at this point can either wait for this message after doing the registration, or it can time out. If the upper layer did not receive this message, it can send down an explicit M_PCPROTO message using the wan_ctl structure with the wan_command set to WC_CONNECT. This message prompts the WAN driver to check for signals and the WAN driver replies with an M_PCPROTO message using the wan_ctl structure with wan_command set to WC_CONCNF and wan_status set to WAN_SUCCESS or WAN_FAIL. This confirms whether the WAN driver can start data transmission and reception.

8. If the upper layer gets an M_PCPROTO message from the WAN driver with the wan_ctl structure and wan_command set to WC_CONNECT, as described in step 7, and if the upper layer is ready for data transfer, it should send down its confirmation (for data transfer) in the form of an M_PCPROTO message using the wan_ctl structure with the wan_command field set to WC_CONCNF and wan_status field to WAN_SUCCESS. This message has to be issued using the putpmsg() STREAMS call. This sets the internal state of the WAN driver to be able to transmit and receive frames.

9. The upper layer can now start transmitting data by sending down M_PROTO messages using the wan_msg structure. The wan_type field of this structure must be set to WAN_DAT. Use the putmsg() STREAMS call to send down this message.

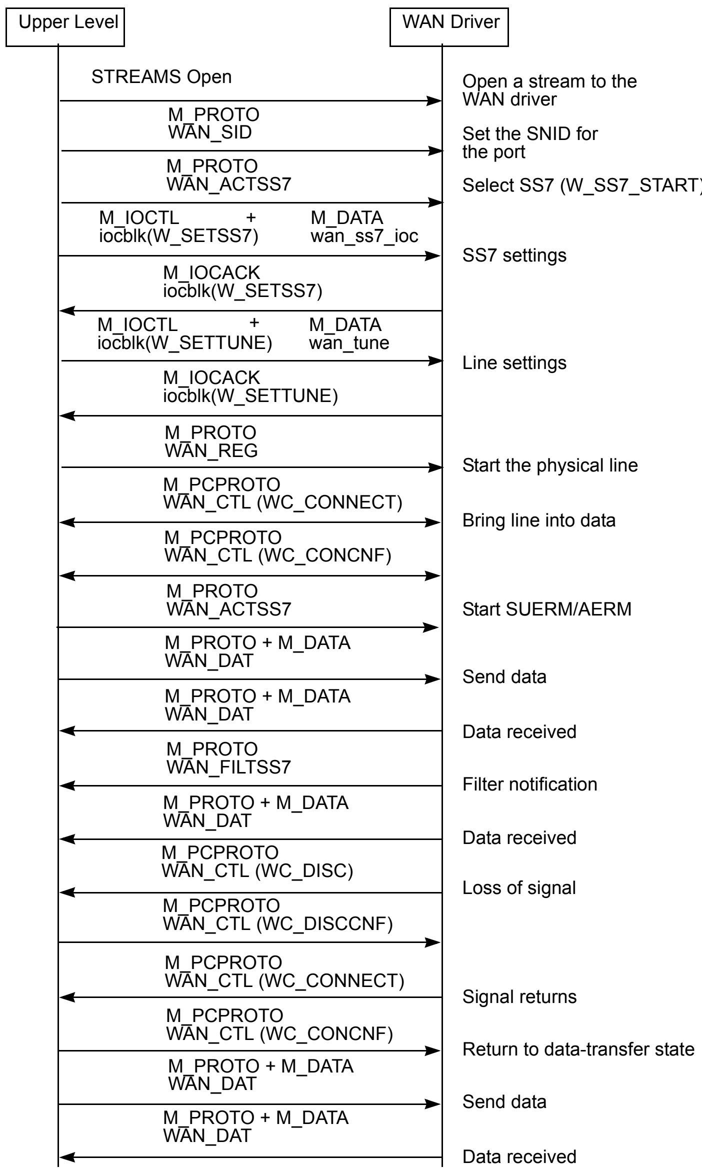
10. Start/Stop SUERM/AERM in the WAN driver by sending the W_ACTSS7 service command. Send this message on the opened stream using the putmsg() call.

11. The upper layer must be prepared to receive the WAN_FILTSS7 message. This is a message initiated by the WAN driver. It carries the number of Signal Units that were discarded due to filtering. This message is sent before the regular WAN_DAT service message which carries the first different SU following a series of similar Signal Units. This is an M_PROTO message.

12. The upper layer should be in a position to handle messages from the WAN driver throughout this sequence. The receiver and transmitter are enabled on a WC_CONCNF when received from the upper layer or when sent to the upper layer. The upper layer can receive messages (at any time during this sequence) by issuing a getmsg() STREAMS call. The upper layer has to decode the type of the message and verify whether it makes sense depending upon the context that it (the upper layer) is in. For example, after sending down the M_PROTO message for registration (WAN_REG), the upper layer should expect an M_PCPROTO message containing the wan_ctl structure with the wan_command field set to WAN_CONNECT.

13. If a control signal drops, the WAN driver sends a WC_DISC to the upper layer. The upper layer *must* send a WC_DISCCNF. The WAN driver checks the presence of the signals on a periodic basis. If the signals are active again, the WAN driver sends a WC_CONNECT to the upper layer, which *must* be acknowledged by a WC_CONCNF. After this, data transfer resumes normally.
14. In the case where the upper layer sends a WC_DISC, the WAN driver does not drop any signals, but suspends transmission and reception of data. The WAN driver replies with WC_DISCCNF. If the upper layer sends a WC_CONNECT, the WAN driver replies with WC_CONCNF and data transfer resumes normally. Signals are dropped only in the case of W_DISABLE.

Figure 4-1. Serial synchronous WAN driver running SS7 protocol



Serial synchronous WAN driver in HDLC framing mode

The following is the order in which the upper level process should issue commands to the WAN driver. The calls made are standard STREAMS application interface calls.

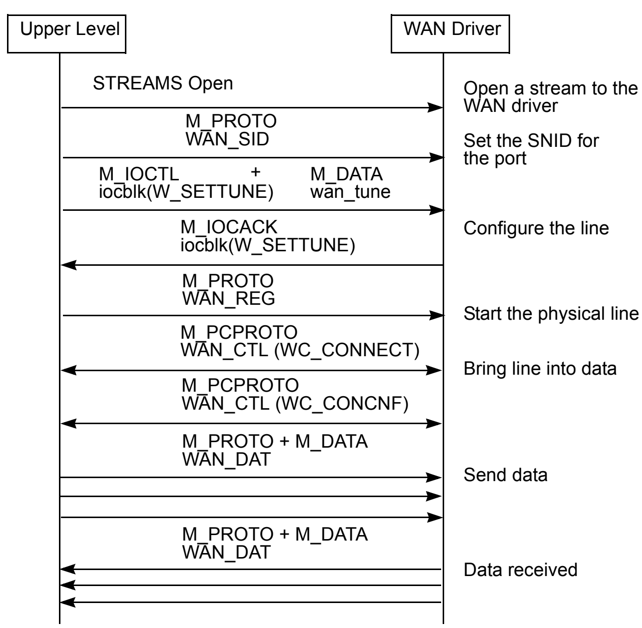
1. Open a stream to the WAN driver using the open() STREAMS call.
2. Set the SNID (Sub Network Id) for the port opened in step 1 by building an M_PROTO message using the wan_sid structure. Send this message on the opened stream using the putmsg() call.

   If the driver was loaded with the SNID_DECODE=YES configuration parameter, the SNID identifies the physical port number.

   If the driver was loaded with the SNID_DECODE=NO configuration parameter, the W_SET_SNID command would associate the physical port number to the SNID.
3. If the default values need to be changed, set the line configuration parameters using the W_SETTUNE command. This command is issued using an ioctl STREAMS call with the line parameters set in the structure wan_tune.
4. Register with the WAN driver using the wan_reg structure encapsulated within an M_PROTO message. The wan_type field in the wan_reg structure should be set to WAN_REG. Use the putmsg STREAMS call to send this message to the WAN driver.
5. Once registration is done, the WAN driver programs the hardware based on the options selected in the W_SETTUNE command. Depending on the interface used, it enables the output signals and checks for the input signals. If the signals are available, it sends up an M_PCPROTO message with wan_command field set to WC_CONNECT using the wan_ctl structure. This indicates to the upper layer that the WAN driver is ready for data transfer. The upper layer at this point can either wait for this message after doing the registration, or it can time out. If the upper layer did not receive this message, it can send down an explicit M_PCPROTO message using the wan_ctl structure with the wan_command set to WC_CONNECT. This message prompts the WAN driver to check for signals and the WAN driver replies with an M_PCPROTO message using the wan_ctl structure with wan_command set to WC_CONCNF and wan_status set to WAN_SUCCESS or WAN_FAIL. This confirms whether the WAN driver can start data transmission and reception.
6. If the upper layer gets an M_PCPROTO message from the WAN driver with the wan_ctl structure and wan_command set to WC_CONNECT (as described in step 5, and if the upper layer is ready for data transfer, it should send down its confirmation (for data transfer) in the form of an M_PCPROTO message using the wan_ctl structure with the wan_command field set to WC_CONCNF and the wan_status field to WAN_SUCCESS. This message has to be issued using the putpmsg() STREAMS call. This sets the internal state of the WAN driver to be able to transmit and receive frames.

7. The upper layer can now start transmitting data by sending down M_PROTO messages using the wan_msg structure. The wan_type field of this structure must be set to WAN_DAT. Use the putmsg() STREAMS call to send down this message.
8. It must be noted that the upper layer should be in a position to handle messages from the WAN driver throughout this sequence. The receiver and transmitter are enabled on a WC_CONCNF when received from the upper layer or when sent to the upper layer. The upper layer can receive messages (at any time during this sequence) by issuing a getmsg() STREAMS call. The upper layer has to decode the type of the message and verify whether it makes sense, depending on the context that it (the upper layer) is in. For example, after sending down the M_PROTO message for registration (WAN_REG), the upper layer should expect an M_PCPROTO message containing the wan_ctl structure with the wan_command field set to WAN_CONNECT.
9. If a control signal drops, the WAN driver sends a WC_DISC to the upper layer. The upper layer *must* send a WC_DISCCNF. The WAN driver checks the presence of the signals on a periodic basis. If the signals are active again, the WAN driver sends a WC_CONNECT to the upper layer, which *must* be acknowledged by a WC_CONCNF. After this, data transfer resumes normally.
10. In the case where the upper layer sends a WC_DISC, the WAN driver does not drop any signals, but suspends transmission and reception of data. The WAN driver replies reply with the WC_DISCCNF. If the upper layer sends a WC_CONNECT, the WAN driver replies with WC_CONCNF and data transfer resume normally. Signals are dropped only in the case of W_DISABLE.

Figure 4-2. Serial synchronous WAN driver in HDLC framing mode



Serial synchronous WAN driver in bisynchronous mode

The following is the order in which the upper level process should issue commands to the WAN driver. The calls made are standard STREAMS application interface calls.

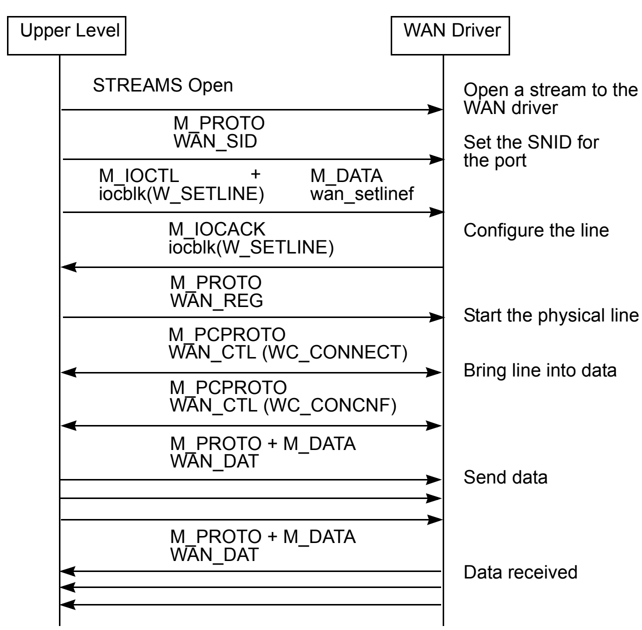
1. Open a stream to the WAN driver using the open() STREAMS call.
2. Set the SNID (Subnetwork ID) for the port opened in step 1 by building an M_PROTO message using the wan_sid structure. Send this message on the opened stream using the putmsg() call.

   If the driver was loaded with the SNID_DECODE=YES configuration parameter, then the SNID identifies the physical port number.

   If the driver was loaded with the SNID_DECODE=NO configuration parameter, then the W_SET_SNID command would associate the physical port number to the SNID.
3. Set the line configuration parameters using the W_SETLINE command. This command is issued using an ioctl STREAMS call with the line parameters set in the structure wan_setlinef.
4. Register with the WAN driver using the wan_reg structure encapsulated within an M_PROTO message. The wan_type field in the wan_reg structure should be set to WAN_REG. Use the putmsg STREAMS call to send this message to the WAN driver.
5. Once registration is done, the WAN driver programs the hardware based on the options selected in the W_SETLINE command. Depending on the interface used, it enables the output signals and checks for the input signals. If the signals are available, it sends up an M_PCPROTO message with wan_command field set to WC_CONNECT using the wan_ctl structure. This indicates to the upper layer that the WAN driver is ready for data transfer. The upper layer at this point can either wait for this message after doing the registration, or it can time out. If the upper layer did not receive this message, it can send down an explicit M_PCPROTO message using the wan_ctl structure with the wan_command set to WC_CONNECT. This message prompts the WAN driver to check for signals and the WAN driver replies with an M_PCPROTO message using the wan_ctl structure with wan_command set to WC_CONCNF and wan_status set to WAN_SUCCESS or WAN_FAIL. This confirms whether the WAN driver can start data transmission and reception.
6. If the upper layer gets an M_PCPROTO message from the WAN driver with the wan_ctl structure and wan_command set to WC_CONNECT (as described in step 5), and if the upper layer is ready for data transfer, it should send down its confirmation (for data transfer) in the form of an M_PCPROTO message using the wan_ctl structure with the wan_command field set to WC_CONCNF and the wan_status field to WAN_SUCCESS. This message has to be issued using the putpmsg() STREAMS call. This sets the internal state of the WAN driver to be able to transmit and receive frames.

7. The upper layer can now start transmitting data by sending down M_PROTO messages using the wan_msg structure. The wan_type field of this structure must be set to WAN_DAT. Use the putmsg() STREAMS call to send down this message.
8. It must be noted that the upper layer should be in a position to handle messages from the WAN driver throughout this sequence. The receiver and transmitter are enabled on a WC_CONCNF when received from the upper layer or when sent to the upper layer. The upper layer can receive messages (at any time during this sequence) by issuing a getmsg() STREAMS call. The upper layer has to decode the type of the message and verify whether it makes sense, depending on the context that it (the upper layer) is in. For example, after sending down the M_PROTO message for registration (WAN_REG), the upper layer should expect an M_PCPROTO message containing the wan_ctl structure with the wan_command field set to WAN_CONNECT.
9. If a control signal drops, the WAN driver sends a WC_DISC to the upper layer. The upper layer *must* send a WC_DISCCNF. The WAN driver checks the presence of the signals on a periodic basis. If the signals are active again, the WAN driver sends a WC_CONNECT to the upper layer, which *must* be acknowledged by a WC_CONCNF. After this, data transfer resumes normally.
10. In the case where the upper layer sends a WC_DISC, the WAN driver does not drop any signals, but suspends transmission and reception of data. The WAN driver replies reply with the WC_DISCCNF. If the upper layer sends a WC_CONNECT, the WAN driver replies with WC_CONCNF and data transfer resume normally. Signals are dropped only in the case of W_DISABLE.

Figure 4-3. Serial synchronous WAN driver in bisynchronous mode



Multiplexed WAN driver in SS7 or HDLC framing

The following takes the user through an ideal configuration scenario and explains how the state of hardware changes.

1. The configuration utility (developed by the user) opens a clone device to the WAN driver.
2. The configuration utility issues one or more W_SETDI_PORT commands to the stream, thus setting the parameters (for example, frame format and CRC) for those ports. At this point, the WAN driver programs the hardware.
3. The configuration utility issues a W_SETDI command to set the clocking source for the ports. This decides which port provides the master clock and also sets up the backup sources.

   Steps 2 and 3 and can be interchanged.

4. Depending on the command-line parameter when the WAN driver is loaded (SNID_DECODE=NO or YES), a series of W_SET_SNID commands or a series of WAN_SID commands are necessary to give identity to each logical channel and map them to physical ports and channels. If logical channels map to physical channels on the SC bus, W_SET_CHMAP commands should be issued to set up the processing paths.
5. The configurable parameters of each logical channel can be set by issuing a series of W_SETTUNE commands (such as maximum frame size).
6. A *clone device open* and subsequent WAN_SID command makes the correlation between the SNID and the logical channel, and the binding would be complete. A *clone device open* is not bound to a specific logical channel until a WAN_SID has been sent.
7. The configuration utility issues a WAN_REG command, which programs the hardware for this channel based on WAN_SID, and takes into account the parameters set by W_SETDI_PORT, W_SETTUNE, and W_SET_CHMAP commands that were issued in the previous steps. After this step, you cannot change the attributes of the hardware port associated with this channel or the configurable parameters for this channel. Switching operating mode (HDLC or SS7) is also not allowed.

   Now the message flow is similar to the Serial WAN driver.

8. At the time of close, depending on the state of the logical channel, the following action is taken: if the channel had been in SS7 mode, it is taken out and put in standard HDLC mode. The channel mapping between the internal and the physical channel is *not* forgotten.

Multiplexed WAN driver in Clear Channel Capability mode

To ensure your adapter supports this mode, contact your RadiSys representative.

The following is the order in which the upper-level process should issue commands to the WAN driver. The calls made are standard STREAMS application interface calls.

1. Open streams to the WAN driver using the open() STREAMS call. Note that all opens to a Multiplexed WAN driver are clone opens. The number of streams opened should be equal to the number of pipes that are to be opened, plus one more stream to perform management commands (before and after the pipes enter the data transfer state).
2. Issue W_SETDI and/or W_SETDI_PORT commands to set up the parameters of the physical links and backup clocks.
3. Issue W_SET_PHY_PIPE commands to specify which time slots are to be combined for the SS7 pipe streams. A unique identifier for a pipe is returned in the w_phy_pipe_id field.
4. Issue one or more W_SET_SNID commands (one for every pipe that is opened). This command ties together the following:
   - The pipe stream identifier, which indicates a combination of time slots over which the physical layer is operating (specified in the w_port_id field)
   - A SNID (unique identifier)
   - An internal channel number returned by the command.
5. Issue one or more W_SETTUNE commands to specify the configurable parameters for the pipes.
6. Send a WAN_SID message on each pipe stream to associate a SNID with the stream.
7. Send a WAN_ACTSS7 message on each pipe stream using the W_CCC_START action to set the mode to SS7 Clear Channel Capability.
8. Issue W_SETSS7_CCC commands to configure the attributes of the SS7 Clear Channel Capability link.
9. At this point, connections can be initiated by issuing WAN_REG and WAN_CTL commands. Once the data transfer mode is entered, WAN_DATs are exchanged between the upper level and the WAN driver.
10. Occasionally, WAN_NOTIFSS7 can be issued by the WAN driver to the upper level.

The data-transfer state can be terminated by issuing WAN_CTL with WC_DISC and performing a close() on the stream.

Multiplexed WAN driver in ATM mode

The following is the order in which the upper-level process should issue commands to the WAN driver. The calls made are standard STREAMS application interface calls.

1. Open streams to the WAN driver using the open() STREAMS call. Note that all opens to a Multiplexed WAN driver are clone opens. The number of streams opened should be equal to the number of virtual channels that are to be opened, plus one more stream to perform management commands (before and after the virtual channels enter the data transfer state).
2. Issue W_SETDI and/or W_SETDI_PORT commands to set up the parameters of the physical links and backup clocks.
3. Issue the W_SET_PHY_PIPE command to specify which time slots are to be combined for the ATM cell stream. Specify a unique identifier in the w_phy_pipe_id field.
4. Issue the W_SET_ATM command to set the parameters related to the physical layer of the ATM. Use the w_phy_pipe_id field to identify the ATM cell stream. This step is optional and can be issued at a later point. However, it cannot be issued after a WAN_REG has been issued on a virtual channel that is operating over this ATM cell stream.
5. Issue one or more W_SET_SNID commands (one for every virtual channel that is to be opened). This command ties together the following:
   - The ATM cell stream, which is a combination of time slots over which the ATM physical layer is operating (specified in the w_port_id field)
   - A VPI/VCI (specified in the w_chnl_id field)
   - A SNID (unique identifier)
   - An internal channel.
6. Issue one or more W_SETTUNE commands to specify the parameters for the CPCS layer.
7. At this point, virtual channels can be started by issuing WAN_SID, WAN_REG and WAN_CTL commands. Once the data transfer mode is entered, WAN_DATs are exchanged between the upper level and the WAN driver. Occasionally, WAN_NOTIF_ATM can be issued by the WAN driver to the upper level.

The data-transfer state can be terminated by issuing WAN_CTL with WC_DISC and performing a close() on the stream.

SC-bus connection scenarios

- Standalone case — The ARTIC960 4-Port T1/E1 Mezzanine Card is not connected to other adapters by way of the SC bus.

  In this case, the user need not issue any commands to configure the SC bus. The default configuration lets the user process data from the network.

- Multiple adapters are connected by way of the SC bus. — However, the ARTIC960 4-Port T1/E1 Mezzanine Card does not forward any data to or from the SC bus.

  In this case, load the WAN driver with proper values for W_SCBUS_XMIT_WIRE and W_SCBUS_RECV_WIRE dedicated wires to avoid conflict. Configure for SC-bus master (because this ARTIC960 4-Port T1/E1 Mezzanine Card is connected to the network) and the proper speed of the SC-bus. Once this is done, the user can process data from the network without any conflicts.

- Multiple adapters are connected by way of the SC bus — One of the ARTIC960 4-Port T1/E1 Mezzanine Cards is connected to the network. All other adapters, and the ARTIC960 4-Port T1/E1 Mezzanine Card, process data from the network.

  In this case, load the WAN driver with proper values for W_SCBUS_XMIT_WIRE and W_SCBUS_RECV_WIRE dedicated wires (this is optional). Also, configure the ARTIC960 4-Port T1/E1 Mezzanine Card that is connected to the network to be the master of the SC-bus. Next, to process data on other adapters, issue W_SETCH_MAP commands to set up the processing paths. Make sure they do not use the dedicated wires (if any are defined). To process data from a network port on the ARTIC960 4-Port T1/E1 Mezzanine Card, you do not need to issue the W_SETCH_MAP command to set the processing path if dedicated wires are defined. Otherwise, the W_SETCH_MAP command must be used to set up the processing paths.

CT-bus connection scenarios

The CT bus is implemented with H.100 or H.110 variants.

- The H.100 bus can be used when the PMC is configured in a PCI system. A ribbon cable connector on the PMC will be used to connect all the CT devices.
- The H.110 bus can be used when the PMC is configured in a Compact PCI system where the H.110 bus resides in the CompactPCI motherboard and is common with all other Compact PCI adapters using the main cPCI bus.

5 Serial and Multiplexed WAN drivers (common operations)

This chapter describes operations that are common to the Serial and the Multiplexed WAN driver when operating under different protocol modes. Supported protocol modes are:

- Synchronous mode (HDLC framing) — The default when either the Serial or Multiplexed WAN driver is loaded.
- Asynchronous mode — Selected by way of W_SETLINE to the Serial WAN driver.
- HDLC framing plus SS7 — Selected when either the Serial or Multiplexed WAN driver is loaded and WAN_ACTSS7 with W_SS7_START is issued on the opened stream.
- Bisynchronous mode — Selected by way of W_SETLINE to the Serial WAN driver.

Most of the streams operations are the same as defined in the *SpiderX25 WAN Implementation Guide, r8.0,* by Spider Systems. However, the following operations are different:

- Encoding of the SNID, described in *WAN_SID — Set subnetwork ID* on page *51*.
- Associating SNID to a port or channel, described in *W_SET_SNID — Allocate internal channel and associate SNID to it* on page *94*.
- Rate licensing mechanism, described in *W_SETTUNE — Set configuration* on page *83*.
- The control of modem signal DCD (data carrier detect), described in *W_SETTUNE — Set configuration* on page *83*.
- The actions taken on W_DISABLE, described on page *68*.
- Currently, there is no SNMP (Simple Network Management Protocol) support.

STREAMS service messages

These are the messages that are sent on the stream associated with the targeted line or channel.

The WAN driver supports different types of service messages. Depending on the value of wan_type, the messages are classified as:

- Initialization
- Registration
- Control
- Data

Each of these types are explained with their respective structures. These are messages, as opposed to commands, and no immediate response message is expected in the opposite direction.

```
union WAN_primitives {
      uint8             wan_type;
      struct wan_reg    wreg;
      struct wan_sid    wsid;
      struct wan_ctl    wctl;
      struct wan_msg    wmsg;
      .........
   };
```

The structures shown in this book are for illustration purposes. The structures are defined in include files that are distributed with the WAN driver.

Table 5-1 summarizes these service messages. Refer to the referenced pages for details.

Table 5-1. Summary of service messages

Message Type	Direction	Parameters	Use	See Page
WAN_SID	Down	SNID	Sent to the driver right after the open. Assigns a SNID to the stream.	51
WAN_REG	Down	SNID on any stream	Registers the upper layer. It indicates that the upper layer is ready to receive data	54
WAN_CTL	Down or Up	• Command type • Remote address • Return result Diagnostics	Controls the connection setup and clear down. Needed when the type of interface has the concept of a *data transfer state*.	56
WAN_DAT	Down or Up	• Command type for Tx or Rx • M_DATA follows with data	Exchanges data messages.	61

WAN_SID — Set subnetwork ID

This message type is used by the upper module when it informs the WAN driver of the subnetwork identifier associated with the stream.

This message can be sent down on any stream, clone or non-clone. Using this message, the user assigns an identity to the stream on which it is sent. A WAN_CTL with WC_CONNECT command is needed before the user stream can enter *data transfer state*.

Only one WAN_SID message can be sent down on a stream.

The following structure is associated with this M_PROTO message:

```
struct wan_sid {
        uint8            wan_type;
        uint8            wan_spare[3];
        uint32           wan_snid;
   };
```

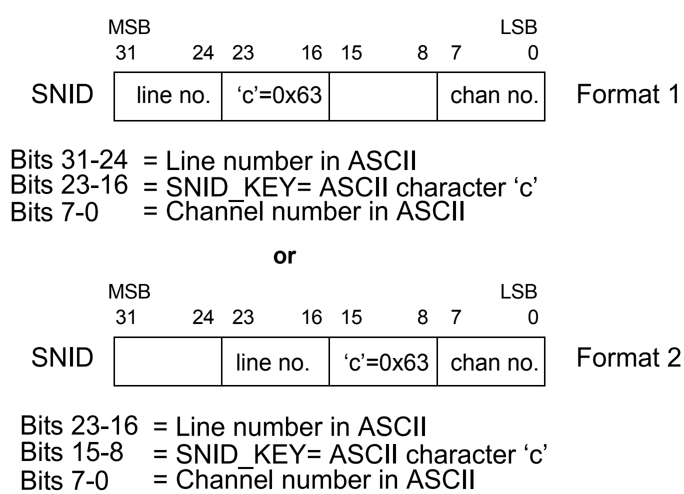
Parameters

`wan_type` This is set to WAN_SID.

`wan_snid` The subnetwork identifier. There are two formats in which this can be specified.

- As a 32-bit integer. In this case, the SNID is a number identifying the channel or line to the Management Entity. The assignment of the stream to a particular line or channel must be achieved in some other way.
- Certain bits of the 32-bit integer occupy the line number. This line number is encoded in ASCII format. The line number is extracted by subtracting hexadecimal 30. This mode is chosen by loading the WAN driver with the command line parameter SNID_DECODE=YES (not supported in ATM mode or pipes). This encoding can be in one of two possible forms, described in *Figure 5-1*.

Figure 5-1. Encoded SNID



The WAN driver first looks at bit positions 8 through 15 for ASCII ‘c’ (SNID_KEY).

- If a ‘c’ is found, bits 16 through 23 carry the line number encoded in ASCII; else bit positions 16 through 23 are examined for ASCII ‘c’ (SNID_KEY).
- If ‘c’ is found in those positions (16 through 23), bits 24 through 31 carry the line number encoded in ASCII. The line number ranges from 1 to 4.
- For the Serial WAN driver, the line number is the same as the port number. The Serial WAN driver ignores bits 0 through 7.
- For the Multiplexed WAN driver:
  - The line number refers to the T1/E1 port number ranging from 1 to 4.
  - The channel number field refers to the time slot within that T1/E1 line, ranging from 1 to H for T1 and 2 to P for E1.

Error codes

0	The message was successfully processed. There is no indication of this in the reverse direction. In case of an error, an M_ERROR message is sent upstream with the appropriate error code. Note that the stream is unusable in such an event.
ENODEV	Either the SNID format cannot be deciphered or cannot be found in SNID_DECODE=NO mode
EINVAL	The message size does not match.
EEXIST	The SNID supplied is already used by another stream.
ERANGE	Either the line or channel number decoded from the SNID field is too large for the current hardware configuration or SNID_KEY is not detected.
EBUSY	Either the channel is currently used by another stream, or the channel is chained to another channel due to chaining at the port level or individual channel basis.
ENOSR	The WAN driver received more WAN_SID messages than the maximum number of logical channels it can support.
EIO	The WAN_SID is in the wrong state.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.

A WAN_SID can be issued again if a W_DISABLE had been issued previously. To return to the *connected state*, issue a WAN_REG.

Figure 5-2. Message flow for WAN_SID

WAN_REG — Registration message — start hardware

This message type is used by the upper module when it would like to register itself with the WAN driver. The WAN driver activates the hardware associated with the line or channel.

Unlike other M_PROTO messages, this message can be sent on any stream.

The following structure is associated with this M_PROTO message:

```
struct wan_reg {
        uint8           wan_type;
        uint8           wan_spare[3];
        uint32          wan_snid;
   };
```

Parameters

wan_type
: This is set to WAN_REG.

wan_snid
: The subnetwork identifier. See the description of the wan_snid parameter on page *51*.

Error codes

0
: The message was successfully processed. There is no indication of this in the reverse direction. In case of an error, an M_ERROR message is sent upstream with the appropriate error code. Note that the stream is unusable in such an event.

ENODEV
: Either the SNID cannot be found among the SNIDs, the SNID format cannot be deciphered, or WAN_SID was not issued.

EINVAL
: The message size does not match.

EXDEV
: The configuration for the port was in conflict, hence was not programmed. That is, the current operational mode of the hardware does not match the cable ID of the attached cable.

EBUSY
: The port is already activated.

EIO
: Either the line or channel is disconnected or in the wrong state, or does not have an associated DSP channel.

ENXIO
: A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.

ENOMEM
: Insufficient memory to register the line or channel.

E2BIG — The host's maximum receive-buffer size is too small to hold the largest frame.

If the hardware cannot be started for any reason, an M_ERROR message is sent upstream

Figure 5-3. Message flow for WAN_REG

WAN_CTL — Connection management

This message type is used by the upper module and the WAN driver to exchange control messages relating to connection setup and clear down.

This message is sent down a particular stream after it has been bound to a line or channel by way of the WAN_SID message.

The following structure is associated with this M_PCPROTO message:

```
struct wan_ctl {
        uint8           wan_type;
        uint8           wan_command;
        uint8           wan_remtype;
        uint8           wan_remsize;
        uint8           wan_remaddr[20];
        uint8           wan_status;
        uint8           wan_diag;
   };
```

Parameters

wan_type This is set to WAN_CTL.

wan_command

Identifies the action to be taken by the recipient on receipt of the message. There are four commands: WC_CONNECT, WC_CONCNF, WC_DISC, and WC_DISCCNF.

WC_CONNECT

When Received by the WAN Driver — Causes it to take appropriate action on the interface hardware to bring the line into a data-transfer state, that is, enables reception and transmission on the line or the T1/E1 channel.

When Sent by the WAN Driver — Indicates to the upper module that the line is ready to enter a data-transfer state and is awaiting WC_CONCNF from the upper layer. If the upper layer does not send WC_CONCNF, the WAN driver does not enter the data-transfer state.

If there is no cable connected to the adapter, the WAN driver waits until one is connected, and then sends WC_CONCNF with the wan_success field set to WAN_SUCCESS or WAN_FAIL when the line is ready.

In both cases, none of the following fields is used:

- wan_remtype
- wan_remsize
- wan_remaddr
- wan_status
- wan_diag

`WC_CONCNF`

When Received by the WAN Driver — Is an indication from the upper layer as to whether it accepts or rejects a previous connect request.

When Sent by the WAN Driver — Is an indication to the upper module in response to a previous connect request as to the result of an attempt to bring the line into data transfer state.

- For the Serial WAN driver, this means proper modem signals are up for the appropriate interface.
- For the Multiplexed WAN driver, this means flags have been detected on that channel.

Both sides are ready for data transfer if wan_status indicates WAN_SUCCESS, meaning idle flags will be transmitted. If SS7 mode was selected on that stream (with W_ACTSS7 and W_START_SS7), the transmission algorithm, described in *Transmission logic* on page 16, is taken into effect after the upper layer attempts to transmit the first SU.

In both cases:

- wan_status is the connection result status and is one of WAN_SUCCESS or WAN_FAIL.
- wan_diag contains any additional hardware or system diagnostic.
- wan_remtype, wan_remaddr and wan_remsize are not used (undefined).

`WC_DISC`

When Received by the WAN Driver — Causes it to take appropriate action on the interface hardware to take the line out of data transfer state, that is, disable reception and transmission on the line or channel.

- For the Serial WAN driver, control signals are not affected.
- For the Multiplexed WAN driver, the idle code is transmitted on the channel.

When Sent by the WAN Driver — Is an indication to the upper module that the line has just exited from data transfer state.

- For the Serial WAN driver, this indicates that one or more modem signals (DCD, CTS or DSR) are down (wan_diag is set to 0) or the nominal rate exceeds the chosen license rate (wan_diag is set to EACCES).
- For the Multiplexed WAN driver, this indicates that a Loss Of Signal failure or errors dictated by *ITU-T Recommendation G.775* have been detected on the T1/E1 port.

In both cases:

- wan_diag — Contains any additional hardware or system diagnostic.
- wan_remtype, wan_remaddr, wan_status and wan_remsize are not used (undefined).

If a cable is disconnected when the stream is in a data-transfer state, a WC_DISC is sent to the upper layer. The WAN driver polls every second to check if the cable is plugged back in. The operator can plug in the same or a different type of cable.

For the Serial WAN driver, this programs the hardware based on the cable that was plugged in and, if appropriate control signals are present, a WC_CONNECT message is sent to the upper layer. The serial WAN driver waits indefinitely for the control signals.

For the Multiplexed WAN driver, this compares the cable type with the current operational mode of the driver. If they match, the driver waits for flag characters to arrive before sending the WC_CONNECT to the upper layer.

See *W_SETDI_PORT — Set attributes of a physical port* on page *165* for additional details.

WC_DISCCNF

When Received by the WAN Driver — Is an indication from the upper layer as to whether it accepts or rejects a previous disconnect request.

When Sent by the WAN Driver — Is an indication to the upper module, in response to a previous disconnect request, as to the result of an attempt to remove the line from a data transfer state. Currently the WAN driver does not reject a disconnect request.

In both cases, the fields show the following:

- wan_status is the disconnection result status and is one of WAN_SUCCESS or WAN_FAIL.
- wan_diag contains any additional hardware or system diagnostic.
- wan_remtype, wan_remaddr, wan_status and wan_remsize are not used (undefined).

`wan_status`
: This field carries WAN_SUCCESS or WAN_FAIL.

`wan_diag`
: Additional information codes or reasons for failure.

  - For the Serial WAN driver, this field carries the result of WC_DISC, described previously.
  - For the Multiplexed WAN driver, see the wan_event field description in *WAN_NOTIFDI — Inform upper level of T1/E1 events* on page *129* for status bits reported.

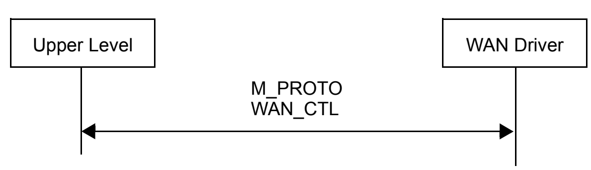
Error codes

0	The message was successfully processed. There is no indication of this in the reverse direction. In case of an error, an M_ERROR message is sent upstream with the appropriate error code. The stream is unusable in such an event.
EINVAL	Either the wan_command was not understood or the message size does not match.
ENXIO	Either the default configuration for the port was in conflict, and hence was not programmed, or there was a severe hardware error. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.
EIO	The line or channel is in the wrong state.
E2BIG	The host's maximum receive-buffer size is too small to hold the largest frame.

- Unless specified otherwise, the fields wan_remtype, wan_remaddr, wan_status, and wan_remsize are not used and should be set to zero when sending the message downstream. The same is done on upstream.
- In case WC_CONNECT or WC_DISC are crossed, an explicit confirmation is still required.
- A disconnect does not mean the hardware is de-programmed. It only means that some signals necessary for transmission of messages are lost. When they return to normal status, the port/channel can be operated.
- For the Serial WAN driver, if a cable is removed during normal operation and a new cable is plugged in, the WAN driver checks for the proper control signals for this new cable type. If these control signals are present, a WC_CONNECT with WAN_SUCCESS is sent to the upper layer.

.

Figure 5-4. Message flow for WAN_CTL



WAN_DAT — Data messages for transmission and reception

This message type is used by the upper module and the WAN driver to exchange (transmit and receive) data messages on the connection (Virtual Channel Connection (VCC) for the ATM protocol). The received messages will be of command type WC_RX, and the transmit messages will be of command type WC_TX.

For the ATM protocol mode, based on how the VCC is set up (W_SETTUNE), this message carries CPCS, ATM, or OAM data.

The following structure is associated with the M_PROTO block of this service message:

```
struct wan_msg {
        uint8     wan_type;
        uint8     wan_command;
};
```

ATM protocol mode

For the ATM protocol mode, the following structures are associated with the first M_DATA block of this service message:

```
#ifdef INCLUDE_ATM
struct wan_msg_atm_rx {
        uint8     wan_aal5_rx_status[8];
        uint8     wan_atm_pt;
        uint8     wan_atm_clp;
        uint8     wan_aal5_uu;
        uint8     wan_aal5_cpi;
};
struct wan_msg_atm_tx {
        uint8     wan_atm_pt;
        uint8     wan_atm_clp;
        uint8     wan_aal5_uu;
        uint8     wan_aal5_cpi;
};

#endif
#define  rx_begin(p)   (uint8 *)((struct wan_msg_atm_rx *)p+1)
#define  tx_begin(p)   (uint8 *)((struct wan_msg_atm_tx *)p+1)
```

Also see *Parameters Specific to the ATM Protocol Mode* on page 64 for more information.

SS7 mode

For SS7 mode, the M_DATA block in the receive direction is formatted as follows:

```
#ifdef SS7_MODE
struct M_DATA_BLK {
  uint8  wan_l2_rsv[RX_HDR_SPACE];/* Defined by the configuration param.*/
  uint32 wan_fltr_cnt ;     /* Filter count 0-0xffffffff, if in SS7 mode */
  uint8  wan_begin[1] ;     /* Actual data stored here                   */
}
#endif
```

Other protocol modes

For all other protocol modes, the M_DATA block in the receive direction is formatted as follows:

```
#ifdef  HDLC_MODE
struct M_DATA_BLK
  uint8  wan_l2_rsv[RX_HDR_SPACE];/* Defined by the configuration param.*/
  uint8  wan_begin[1] ;           /* Actual data stored here           */
#endif
```

Parameters

`wan_type` Input. This is set to WAN_DAT (for all protocols).

`wan_command`

Input/Output.

WC_TX Input for transmit.

This bit needs to be set on all messages to be transmitted.

WC_BSC_TRANSP

Input to specify transparent data. Set this bit along with WC_TX for bisynchronous protocol only.

WC_RX Output for received messages.

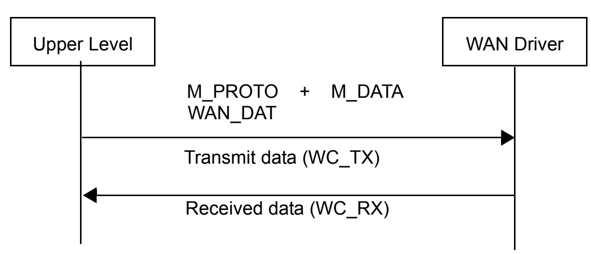
For BISYNC transparent mode, the WAN driver sets one of the following receive message types along with WC_RX.

Message Type	Message Description
WC_ETB	ETB
WC_ETX	ETX
WC_ACK0	ACK0
WC_ACK1	ACK1
WC_WACK	WAK
WC_NAK	NAK
WC_ENQ	ENQ
WC_EOT	EOT
WC_RVI	RVI
WC_DISC_BSC	DLE EOT (DISCONNECT)
WC_STX_ITB	STX ITB
WC_STX_ETB	STX ETB
WC_STX_ETX	STX ETX
WC_STX_ENQ	STX ENQ (TTD)
WC_SOH_ITB	SOH ITB
WC_SOH_ETB	SOH ETB
WC_SOH_ETX	SOH ETX
WC_SOH_ENQ	SOH ENQ
WC_DATA_ACK0	data ACK0

WC_DATA_ACK1	data ACK1
WC_DATA_NAK	data NAK
WC_DATA_ENQ	data ENQ

- The data is contained in one or more M_DATA blocks following the M_PROTO header block. There is no significance in the division of data between M_DATA blocks. If the upper layers can make sure that the data is contained in a single M_DATA block, the user can set ONE_DATA_MSG_ONLY=YES command-line parameter.
- The ONE_DATA_MSG_ONLY=YES command-line parameter is recommended when using the ATM protocol.
- This interface can be changed using the DATA_MSG_ONLY=YES command line parameter, in which case, only M_DATA blocks are present (no M_PROTO header). In some cases, it may be worthwhile to avoid the overhead of allocation and freeing of the little M_PROTO header. BISYNC does not support DATA_MSG_ONLY=YES.
- This M_DATA interface can be effectively used only if the data blocks do not go up to the stream head, because the stream head could concatenate all M_DATA blocks while delivering them across the read () system call interface.
- RX_HDR_SPACE is defined at configuration time when the driver is loaded. See *Command-line parameters* on page 235 for more details. The field wan_l2_rsv is used by the layers above the driver. The field wan_fltr_cnt is used only when the stream is in SS7 mode.
- When in SS7 mode, the WAN driver assumes that the FISU, LSSU and the first three bytes of an MSU are contained in one M_DATA block.

Figure 5-5. Message flow for WAN_DAT

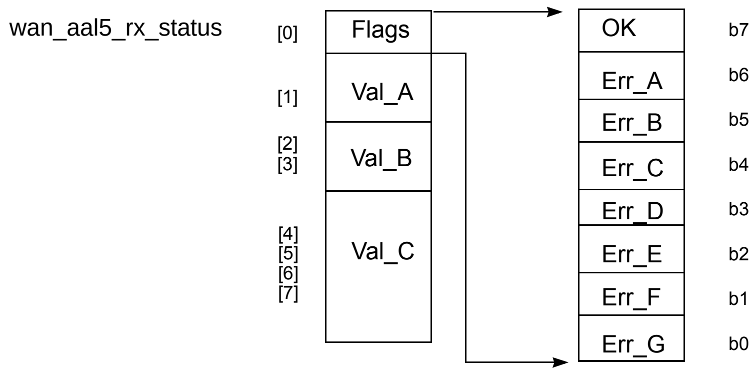


Parameters Specific to the ATM Protocol Mode

The M_DATA block in the receive and transmit direction is formatted as defined by the wan_msg_atm_rx and wan_msg_atm_tx structures respectively. The following defines the structure's members.

wan_aal5_rx_status

Output. This is set by the WAN driver on a received frame. The following lists the format of this field and is in accordance with the *ITU-T 363.5* specifications. This field is filled in only if the WAN_CRPT_DATA_DLVR_ FLAG field is set to TRUE in the W_SETTUNE command.



where:

OK	Is set if no errors were detected.
Err_A	Is set if an illegal CRC remainder was detected.
Err_B	Is set if an illegal CPI was detected.
Err_C	Is set if the value of the Length field in the perceived CPCS-PDU trailer is 0.
Err_D	Is set if an illegal length of a PAD field was detected.
Err_E	Is set if the value of the Length field in the perceived CPCS-PDU trailer exceeds the value of the WAN_maxframe parameter.
Err_F	Is set if the CPCS-SDU length exceeds the value of the WAN_crpt_sdu_dlvr_len parameter.
Err_G	Is set if a reassembly timer expiration has occurred prior to completion of the CPCS-SDU assembly. In this case, Val_A, Val_B, and Val_C have no information.
Val_A	Contains the second octet of the assumed CPCS-PDU trailer (CPI). If OK is set, this field is ignored.
Val_B	Contains the third and fourth octets of the assumed CPCS-PDU trailer (Length). If OK is set, this field is ignored.
Val_C	Contains the last four octets of the assumed CPCS-PDU trailer (CRC). If OK is set, this field is ignored.

`wan_atm_pt`

Input/Output. This field contains the payload type as explained in the *ITU-TI I.361* specifications and is specified by the least-significant 3 bits (rightmost) of this field. (Bits 3, 2, and 1 of this field, with bit 1 being the least-significant bit, correspond to bits 4, 3 and 2 of the ITU-T specifications.)

Receive Direction (from the line)
For a CPCS-SAR VCC, the WAN driver sets this to the payload type indicated by the last SAR-UNITDATA. Note that the congestion indicator is a bit within the payload type.

Transmit Direction (to the line)
Depending on the mode of the VCC (the WAN driver does not make any checks for bits being improper for a particular VCC mode), the following applies:

WAN_CPCS Mode

The WAN driver copies bits 3 and 1 of this field into the payload type field's bits 4 and 2, respectively, for every SAR-UNITDATA. In addition:

- The WAN driver controls the setting of bit 3 of the payload type field, which is the ATM user-to-ATM user indication bit. Hence, bit 2 of this field must be set to zero.
- Bit 3 of this field indicates whether the message is a CPCS PDU or a F5 OAM cell.

WAN_CRC10 or WAN_ATM_CELLS Mode

The payload type field's bits 4, 3 and 2 are set according to bits 3, 2 and 1 of this field, respectively.

`wan_atm_clp`

Input/Output. This field indicates the cell loss priority and is specified in the least significant bit.

Receive Direction
For a CPCS-SAR VCC, this is a binary OR of all SAR-UNITDATAs that made up this CPCS PDU.

Transmit Direction
For a CPCS-SAR VCC, the WAN driver sets this in every SAR-UNITDATA.

`wan_aal5_uu`

Input/Output. Depending on the direction, the following applies.

Receive Direction
Set by the WAN driver to indicate the received CPCS user-to-user information.

Transmit Direction
Set by the upper layer to indicate the CPCS user-to-user information.

`wan_aal5_cpi`

Input/Output. Depending on the direction, the following applies.

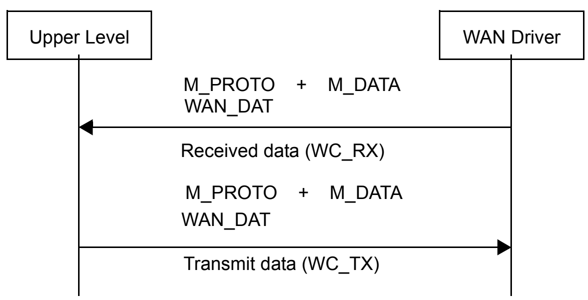
Receive Direction
Set by the WAN driver to indicate the received CPCS Common Part indicator.

Transmit Direction
Set by the upper layer to indicate the CPCS Common Part indicator.

- In the transmit direction, if the data length is greater than a cell's payload length, and the VCC is in raw mode (that is, W_SETTUNE with WAN_ATM_CELLS), then the WAN driver will segment the data into multiple ATM cells with PT and CLP as specified by the wan_atm_pt and wan_atm_clp fields.
- In the receive direction, if the VCC is in raw mode (that is, W_SETTUNE with WAN_ATM_CELLS), the WAN driver does not do any OAM processing or CRC-10 on these cells; they will be passed up to the upper level on an *as-is* basis

.

Figure 5-6. Message flow for WAN_DAT — ATM protocol mode



STREAMS management commands

These are M_IOCTL commands that are exchanged on any stream. They contain one M_IOCTL message block with an iocblk structure in its data block followed by zero or more M_DATA message blocks. The WAN driver knows the associated channel or line through the use of the given SNID. All M_IOCTL commands are replied to by the WAN driver setting the ioctl message block type to M_IOCACK or M_IOCNAK for success or failure respectively. For the following commands, the actions or information passed is different for Serial and Multiplexed WAN drivers.

- W_SETTUNE
- W_GETTUNE
- W_DISABLE
- W_ENABLE

Commands that are not supported or not valid for the driver (that is, the Serial WAN driver getting commands for T1/E1) will be replied by M_IOCNAK.

Table 5-2, "STREAMS management commands common to Serial and Multiplexed WAN drivers," on page 68 summarizes the management commands.

Table 5-2. STREAMS management commands common to Serial and Multiplexed WAN drivers

ioctl Command	M_DATA Content besides SNID	Use	See Page
W_DISABLE		To disable a port; allows for a temporary disable without doing close.	69
W_ENABLE		To enable a port; allows for re-enabling of a port that was temporarily disabled without doing close.	69
W_GETDRVINFO	Various driver parameters; no SNID is associated with this.	To obtain information, such as version number and command-line parameter settings.	71
W_GETHWTYPE	Electrical interface	To obtain the type of cable attached and current operational mode.	74
W_GETSTATS	Table of statistics	To read the statistics associated with a line or channel.	78
W_ZEROSTATS	Table of statistics	To reset the statistics for a line or channel.	81
W_SETTUNE	Table of tuning values	To set the configurable parameters for a line.	83
W_GETTUNE	Table of tuning values	To obtain the configurable parameters for a line.	93
W_SET_SNID	wan_set_snid_ioc	To allocate internal channel ID and associate SNID to it	94
W_GET_SNID	wan_set_snid_ioc	To obtain the internal channel ID and physical port/channel associated with the SNID	101
W_REL_SNID	wan_rel_snid_ioc	To free internal channel ID	102

W_DISABLE/W_ENABLE — Disable/enable transmission of data

When W_DISABLE is received by the WAN driver, all received data is discarded (if the line or channel is capable of receiving data). Any request for transmission of data is ignored.

For the Serial WAN driver:

- The output control signals are turned off depending on the interface board.
- W_DISABLE results in WAN_CTL/WC_DISC on the corresponding data stream. A subsequent W_ENABLE would result in WAN_CTL/WC_CONNECT if appropriate control signals are present.

For the Multiplexed WAN driver:

- The transmitter transmits the flag-idle pattern.
- The user can disable a channel, change the current mapping, and then enable it.
- W_DISABLE results in WAN_CTL/WC_DISC on the corresponding data stream. A subsequent W_ENABLE would result in WAN_CTL/WC_CONNECT if the channel starts receiving flags again.
- When a WAN_SID is issued on the stream for the first time, the channel is enabled by default. Subsequently, a WAN_REG would bring the channel into data transfer. Now, if a W_DISABLE is issued, then W_ENABLE must be done, even if the corresponding data stream is closed and some other stream uses WAN_SID.

W_ENABLE is sent to re-enable the reception and transmission of data.

The following structure is associated with this command:

```
struct wan_hdioc {
   uint8            w_type;
   uint8            w_spare[3];
   uint32           w_snid;
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_DISABLE or W_ENABLE.

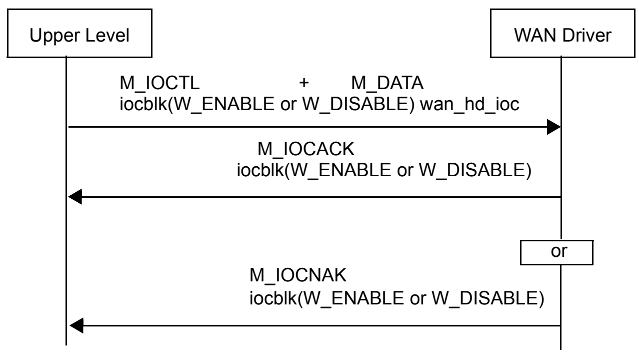
w_type
: Input. This is set to WAN_PLAIN.

w_snid
: Input. The subnetwork identifier. See the description of the wan_snid parameter on page *51*.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
ENODEV	Either the SNID cannot be found among the SNIDs or the SNID format cannot be deciphered.
EINVAL	The message size does not match.
EIO	The new mapping, done while the channel was disabled, put this channel into the wrong state. See error EIO for WAN_REG on page *54*.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.

Figure 5-7. Message flow for W_ENABLE/W_DISABLE



W_GETDRVINFO — Get driver configuration information

When W_GETDRVINFO is received by the WAN driver, all the information regarding the driver is sent back to the upper layer.

The following structure is associated with this command:

```
typedef struct  wan_params {
        uint32  w_max_non_clone ;
        uint32  w_max_opens ;
        uint32  w_snid_decode ;
        uint32  w_snid_key ;
        uint32  w_data_msg_only ;
        uint32  w_one_data_msg_only ;
        uint32  w_test_interface ;
        uint32  w_tx_blks ;
        uint32  w_rx_blks ;
        uint32  w_rx_hdr_space ;
        uint32  w_scbus_xmit_wire ;
        uint32  w_scbus_recv_wire ;
        uint32  w_scbus_framing_mode ;
        uint32  w_net_switch_mode ;
        uint32  w_interface_type ;
        uint32  w_bsn_flag ;
        uint32  w_logical_port_base ;
        uint32  w_pmc_select ;
        uint32  w_rx_crc_select
        uint32  w_ss7_filter_count ;
        uint32  w_monitor_mode ;
        uint32  w_tdm_clock_rate ;
} wan_params_t  ;

typedef struct wan_mux_hw {
        uint32  w_num_of_dsps ;
        uint32  w_chans_per_dsp ;
        uint32  w_num_of_proc_ports ;
} wan_mux_hw_t;

typedef struct  wan_devinfo {
        uint32        w_wan_ver ;
        uint32        w_func_spec_ver ;
        uint32        w_wan_type ;
        wan_params_t  w_params ;
        wan_mux_hw_t  w_mux_hw ;
} wan_devinfo_t ;

struct wan_drvinfo {
  uint8               w_type;
  uint8               w_spare[3];
  wan_devinfo_t       w_devinfo ;
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_GETDRVINFO.

w_type
: Input. This is set to WAN_GETDRVINFO.

w_wan_ver
: Output. This is set to the version number of the driver code that is loaded.

w_func_spec_ver
: Output. This is set to the version number of the functional specifications of the WAN driver.

w_wan_type
: Output. This is set to the type of the WAN driver (W_SERIAL, W_MUX or W_ASYNC).

wan_params structure
: Output. See *Command-line parameters* on page *235* for details about the individual parameters.

wan_mux_hw structure
: Output. Applies to the Multiplexed WAN driver and indicates how the workload is distributed.

  w_num_of_dsps
  : Total number of DSPs on the PMC

  w_chans_per_dsp
  : Number of channels processed per DSP

  w_num_of_proc_ports
  : Number of processing ports that are available.

    For the ARTIC 4-Port T1/E1 PMC, this is set to 1.

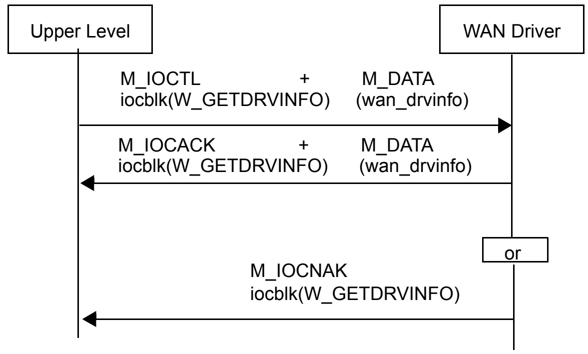
    For the ARTIC 4-Port T1/E1/J1 DSP PMC, this is set to 2.

Error codes

0
: The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

EINVAL
: The message size does not match.

Figure 5-8. Message flow for W_GETDRVINFO



W_GETHWTYPE — Get hardware type

This command is used to obtain information on the hardware (cable type and current mode) for a particular line.

ARTIC960

The following structure is associated with this command:

```
struct wan_gethwtype_ioc {
   uint8            w_type;
   uint8            w_spare[3];
   uint32           w_port_id;
   uint32           w_cable_type;
   uint32           w_current_mode;

};
```

ARTIC 1000/2000 Series

The following structure is associated with this command:

```
struct wan_gethwtype_ioc {
   uint8            w_type;
   uint8            w_spare[3];
   uint32           w_port_id;
   uint32           w_cable_type;
   uint32           w_rtm;
   uint32           w_pmc1;
   uint32           w_pmc2;
   uint32           w_cable_type1;
   uint32           w_cable_type2;
   uint32           w_current_mode;
};
```

Parameters

IOCTL_COMMAND
Input. The ioc_cmd field in struct iocblk should be W_GETHWTYPE.

w_type Input. This is always WAN_GETHWTYPE.

w_port_id
Input. The port number for which the information is requested.

w_cable_type
Output. The type of cable found on the card. The following types are defined:

WAN_V35_DTE
V.35 DTE interface

WAN_V35_DCE
V.35 DCE interface

WAN_RS449
RS-449 interface

WAN_RS422
RS-422 or EIA-530 interface

WAN_RS232
RS-232 or V.24 interface

WAN_X21
X.21 interface

The above cables are available in 2-port and 4-port configurations. 2-port cards support both the 2-port and 4-port cables.

WAN_2PORT_2TYPE
Port 0 = V.35 DTE
Port 1 = RS-232

WAN_RJ48
Generic RJ-48 connection

WAN_T1_RJ48_BAL
T1 interface, balanced RJ-48 jack connector

WAN_T1_TELCO_BAL
T1 interface, balanced telephone-jack connector

WAN_E1_RJ48_BAL
E1 interface, balanced RJ-48 jack connector

WAN_E1_BNC_UNBAL
E1 interface, unbalanced BNC connector

WAN_E1_BNC_BAL
E1 interface, balanced BNC connector

WAN_NO_WRAP_OR_CABLE
If no wrap plug or cable is attached

WAN_120_PIN_WRAP
If a 120-pin wrap plug is connected

WAN_T1E1_WRAP
If a T1/E1 wrap plug is connected

WAN_UNKNOWN
Cable ID is invalid.

w_current_mode
Output. The current operational mode for the RadiSys 4-Port T1/E1 PMC.

The following settings are defined:

W_T1 Configured for T1

W_E1 Configured for E1

W_J1 Configured for J1

w_rtm
: Output. Indicates whether the Rear IO module has been detected. The following types are defined:

  W_RTM_ATTACHED
  : The Rear I/O module has been detected by the software. If the RTM is connected, only Rear I/O is allowed; that is, the rear cable connections must be used.

  W_RTM_NOTATTACHED
  : The Rear IO module has not been detected. Normal cable connection will be enabled.

w_pmc1
: Output. Returns the following, depending on whether PMC #1 is attached.

  W_PMC1_NOTATTACHED
  : PMC #1 is not attached.

  W_PMC1_SERIAL
  : PMC #1 is attached and the adapter is an ARTIC 4-Port Serial PMC.

  W_PMC1_T1E1_DSP
  : PMC #1 is attached and the adapter is an ARTIC 4-Port T1/E1/J1 DSP PMC.

w_pmc2
: Output. Returns the following, depending on whether PMC #2 is attached.

  W_PMC2_NOTATTACHED
  : PMC #2 is not attached.

  W_PMC2_SERIAL
  : PMC #2 is attached and the adapter is an ARTIC 4-Port Serial PMC.

  W_PMC2_T1E1_PMC
  : PMC #2 is attached and the adapter is an ARTIC 4-Port T1/E1/J1 DSP PMC.

w_cabletype1
: Output. This describes the cable attached to PMC #1 for the ARTIC 4-Port T1/E1/J1 DSP PMC. For Serial PMCs on ARTIC 1000/2000 Series, see *w_cable_type* on page 74 for a list of the cable types defined.

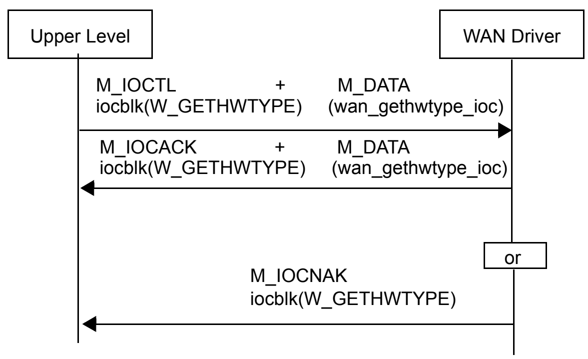
w_cabletype2
: Output. This describes the cable attached to PMC #2 for the ARTIC 4-Port T1/E1/J1 DSP PMC. For Serial PMCs on ARTIC 1000/2000 Series, see *w_cable_type* on page 74 for a list of the cable types defined.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
EINVAL	The message size does not match.
ERANGE	The port number supplied is out of range for the current hardware.

- The hardware type is determined by the PMC attached.
- The initial settings for the hardware are documented in the section *Initial line characteristics* on page *242*.
- The behavior of the Multiplexed WAN driver during initialization is described in *W_SETDI_PORT — Set attributes of a physical port* on page *165*.

Figure 5-9. Message flow for W_GETHWTYPE



W_GETSTATS — Get statistics

This command is not valid in ATM mode.

This command is used to read the statistics from the WAN driver. Statistics are maintained on a line or channel basis, and the required line or channel is selected using the w_snid field. The hdlc_stats field holds the returned statistics.

The following structure is associated with this command:

```
typedef struct  hstats {
        uint32        hc_txgood;
        uint32        hc_txurun;
        uint32        hc_rxgood;
        uint32        hc_rxorun;
        uint32        hc_rxcrc;
        uint32        hc_rxnobuf;
        uint32        hc_rxnflow;
        uint32        hc_rxoflow;
        uint32        hc_rxabort;
        uint32        hc_intframes;
} hdlcstats_t;

struct wan_stioc {
        uint8            w_type;
        uint8            w_state;
        uint8            w_spare[2];
        uint32           w_snid;
        hdlcstats_t      hdlc_stats;
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_GETSTATS.

w_type
: Input. This is always WAN_STATS.

w_state
: Output. This reflects the state of the hardware state machine. Reserved for internal use only.

w_snid
: Input. The subnetwork identifier. See the description of the wan_snid parameter on page *51*.

hdlcstats
: These are the statistics collected since the last time the counters were cleared. The following fields are defined for the structure:

  hc_txgood
  : Output. The number of good frames transmitted.

  hc_txurun
  : Output. The number of transmit underruns.

  hc_rxgood
  : Output. The number of good frames received.

hc_rxorun
: Output. The number of receive overruns.

hc_rxcrc
: Output. The number of receive CRC/Framing/short-frame errors.

hc_rxnobuf
: Output. The number of receive frames with no buffer.

  For BISYNC, the number of received messages with parity error.

hc_rxnflow
: Output. The number of receive frames with no flow control.

hc_rxoflow
: Output. The number of times the receive buffer overflowed.

hc_rxabort
: Output. The number of aborted frames.

  For BISYNC, the number of pad errors.

hc_intframes
: Output. The number of frames failed to be transferred due to loss of signals.

  This will be used to report the number of transmit CTS underruns.

Error codes

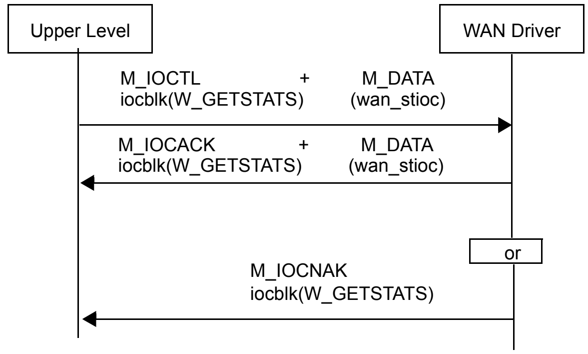
0
: The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

ENODEV
: Either the SNID cannot be found among the SNIDs, or the SNID format cannot be deciphered.

EINVAL
: The message size does not match.

ENXIO
: A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.

Figure 5-10. Message flow for W_GETSTATS



W_ZEROSTATS — Clear channel statistics

This command is not valid in ATM mode.

This command is used to reset the statistics maintained by the WAN driver. Statistics are maintained on a line or channel basis and the required line or channel is selected using the w_snid field. The hdlc_stats field holds the statistics before the counters in the driver are cleared.

The following structure is associated with this command:

```
struct wan_stioc {
        uint8           w_type;
        uint8           w_state;
        uint8           w_spare[2];
        uint32          w_snid;
        hdlcstats_t     hdlc_stats;
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_ZEROSTATS.

w_type
: Input. This is always WAN_STATS.

w_state
: Output. This reflects the state of the hardware state machine.

w_snid
: Input. The subnetwork identifier. See the description of the wan_snid parameter on page 51.

hdlc_stats
: These are values of the counters *before* they were cleared. The entity responsible for collecting must add these numbers to previously acquired ones. See *W_GETSTATS — Get statistics* on page 78 for a description of this field and its elements.

Error codes

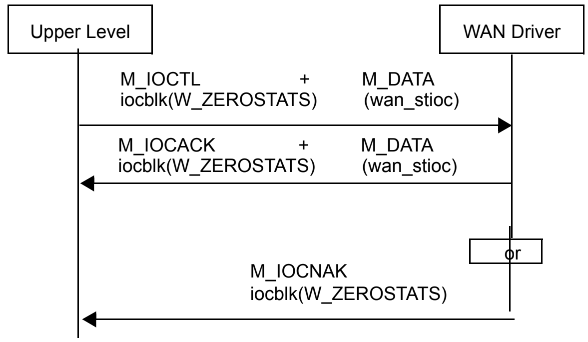
0
: The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

ENODEV
: Either the SNID cannot be found among the SNIDs, or the SNID format cannot be deciphered.

EINVAL
: The message size does not match, or the command is not valid for the ATM mode.

ENXIO
: A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.

Figure 5-11. Message flow for W_ZEROSTATS



W_SETTUNE — Set configuration

For the Serial WAN driver and the Multiplexed WAN driver, this command is used to set the following.

- Configurable parameters of the logical line
- Parameters associated with an ATM virtual channel

For Serial WAN, this command can be used in lieu of W_SETLINE.

To use the X.21 electrical interface in the synchronous Serial WAN driver, you must issue a W_SETLINE command. The W_SETTUNE command will not initialize the X.21 interface. The X.21 cable can be configured and used in non-X.21 mode.

The following structures are associated with this command:

```
struct WAN_atm_vcc {
        uint32      WAN_crpt_sdu_dlvr_len;
        uint32      WAN_cpcs_timer_value;
        uint32      WAN_event_disc;
        uint32      WAN_options;
};
struct WAN_mux {
        uint16      WAN_stat_port;
        uint16      WAN_bit_inv;
        uint32      WAN_event_disc;
};
struct  WAN_hddef {
        uint32    WAN_baud;
        uint16    WAN_maxframe;
        uint16    WAN_interface;
        union }
              uint16              WAN_cptype;
              struct WAN_mux      WAN_muxdef;
              struct WAN_atm_vcc  WAN_atmdef;
        } WAN_cpdef;
};
typedef struct  wan_tune {
      uint16           WAN_options;
      uint16           WAN_pad;
      struct WAN_hddef WAN_hd;
}wantune_t;
struct wan_tnioc {
      uint8     w_type;
      uint8     w_spare[3];
      uint32    w_snid;
      wantune_t wan_tune;
} ;
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_SETTUNE.

w_type
: Input. This is set to WAN_TUNE.

w_snid
: Input. The subnetwork identifier. See the description of the wan_snid parameter on page 51.

WAN_options (associated with the wan_tune structure)
: Input. Options for addressing.

  W_NO_TRANSLATE
  : No address translation.

  W_TRANSLATE
  : Address translation; not supported.

WAN_pad
: Input. This field must be set to zero.

WAN_baud
: Input. The number of bits per second (bps) at which the communication takes place.

  W_EXT_CLK_VERF_TXC
  : External clock from TXC with verification of rate. If the nominal rate is greater than the provided one (license rate bits 28–0), a WC_DISC is sent up the stream. This is a bit field (bit 31) that must be logical ORed with the internally generated baud rate. The hardware is acting as a DTE. Data received from DCE is sampled using the RXC from DCE. Rate comparison is done using the TXC from DCE. The comparison process is started when a WAN_REG is received. If this comparison fails, a WC_DISC is sent up the stream asynchronously. After any disconnect from either side, this process is repeated on entering the connect state. XTC is transmitted to the DCE and is taken from the TXC.

  W_DCE_INT_XTC_EXT_RXC
  : The hardware is acting as a DCE. The internal clock is generated on XTC based on the baud rates that are set (bits 28–0). This clocks DCE's TXD pin. The DCE's RXC receives the clock from the DTE's XTC, which is used to sample the DCE's RXD. This is a bit field (bit 30) that must be logical ORed with an internally generated baud rate.

W_DCE_INT_XTC_INT_RXC

The hardware is acting as a DCE. The internal clock is generated on the XTC based on the baud rates that are set (bits 28–0). This clocks DCE's TXD pin. DCE's RXC receives the clock from DCE's XTC internally, which is used to sample DCE's RXD. This is a bit field (bit 29) that must be logical ORed with an internally generated baud rate.

The following WAN_baud options have bit rate options that are used as bits 28–0 to specify a bit rate:

- W_EXT_CLK_VERF_TXC
- W_DCE_INT_XTC_EXT_RXC
- W_DCE_INT_XTC_INT_RXC

The bit rate options are as follows:

W_300_BPS	300 bits per second
W_600_BPS	600 bits per second
W_1200_BPS	1,200 bits per second
W_2400_BPS	2,400 bits per second
W_3600_BPS	3,600 bits per second
W_4800_BPS	4,800 bits per second
W_7200_BPS	7,200 bits per second
W_9600_BPS	9,600 bits per second
W_19200_BPS	19,200 bits per second
W_38400_BPS	38,400 bits per second
W_48500_BPS	48,500 bits per second
W_56000_BPS	56,000 bits per second
W_64000_BPS	64,000 bits per second
W_76800_BPS	76,800 bits per second
W_1544_BPS	1,544,000 bits per second
W_2048_BPS	2,048,000 bits per second

W_DTE_CLK_FROM_TXC

The hardware is acting as a DTE (clocking from external source). The data received from the DCE is sampled using the receive clock from the DCE. The receive clock is also used as the clock for DTE's transmitted data.

This is the only option supported X.21 port types.

This option has a value of zero and is the default.

`W_DTE_TX_FROM_TXC_RX_FROM_RXC`

The hardware is acting as a DTE (clocking from external source). The data received from the DCE is sampled using the receive clock from the DCE. The DCE transmit clock is used as the clock for DTE's transmit data. DTE transmit data is transmitted based on the external transmit clock. This is valid only for RS-232 electrical interface.

- The description of this parameter (WAN_baud) contains a list of all supported bit rates as bits per second (actual value). The bit rates supported depend on the type of application interface board, the type of adapter, the system requirements, and the I/O mode. These are used only as the transmit baud rate.
- The Multiplexed WAN driver supports W_56000_BPS and W_64000_BPS.
- When the hardware is acting as a DTE, the external clock provided by DCE can range from 0–2,048,000 bits per second.
- BISYNC supports only DTE (external) clock options.

`WAN_maxframe`

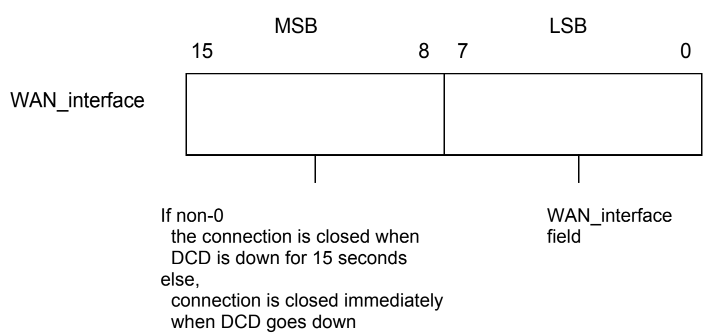
Input. WAN maximum frame size expressed in bytes. See *Multiplexed WAN driver for any of its channels — defaults* on page 242 for default values.

`WAN_interface`

Input. The hardware interface type and control of DCD modem signal. The LSB is the hardware interface.

For the Serial WAN driver, the MSB reflects the effect of DCD going down.

Figure 5-12. Format of WAN_interface of WAN_hddef structure



The following values are valid for this hardware interface parameter:

WAN_V35 Input selects V.35 interface.

WAN_V36 Input selects V.36 interface.

WAN_RS232 Input selects RS-232 interface.

WAN_RS422 Input selects RS-422 interface.

WAN_T1E1 Input selects T1/E1 Multiplexer interface. In this case, bits 8–15 are ignored. Also, the union WAN_cpdef is always interpreted as struct WAN_mux.

WAN_ATM Input selects ATM operation.

WAN_2PORT_2TYPE
Input selects:

- V35 DTE interface on Port 0
- RS-232 interface on Port 1

When the cable type is selected using the W_SETTUNE command's wan_interface parameter, the Serial WAN driver will test the cable type attached and select the interface based on the cable attached. Issue a W_GETTUNE command to verify the cable type attached.

WAN_X21 Input selects the X.21 electrical interface.

To use the X.21 electrical interface, a W_SETLINE command must be issued with the w_porttype parameter set to WAN_X21, with the X.21 cable attached. If this is not done, the X.21 cable can be used without the X.21 electrical interface. See *W_SETLINE — Define line characteristics* on page 209 for more information.

WAN_stat_port
This field is reserved for future use and must be set to 0.

WAN_bit_inv
Input.

W_NO_CHANGE
No change from previous setting.

W_INVERT Apply bit inversion to incoming and outgoing bit streams.

W_NO_INVERT
Normal mode; no inversion. (Default)

`WAN_event_disc`

Input. Defines which events will be reported as disconnect; that is, generates WC_DISC. This bit field takes bit combinations as defined in the wan_event field of the WAN_NOTIFDI message (see *WAN_NOTIFDI — Inform upper level of T1/E1 events* on page *129*). The default for HDLC and SS7 modes is the following (that is, generates disconnect if any of these events are detected):

- W_DI_FAR_RAI
- W_DI_FAR_AIS
- W_DI_LOS
- W_DI_FAR_LFA
- W_DI_FAR_LMFA

`WAN_crpt_sdu_dlvr_len`

Input. If WAN_DATA_DLVR_FLAG is TRUE, this field indicates the maximum number of octets of an assumed CPCS SDU that can be delivered to the CPCS user. This parameter corresponds to the Max_Corrupted_SDU_ Deliver_Length parameter, as described in the *ITU-T I.363.5* specifications. The default value is 4,120.

`WAN_cpcs_timer_value`

Input. This specifies the reassembly timer value. A nonzero value indicates the timer value in multiples of 125 microseconds, and its value should not exceed 65535. This provides a maximum time-out value of approximately eight seconds. The actions taken on timer expiration are described in Annex E of the *ITU-T 363.5* specification.

The default value is zero, indicating the reassembly timer is not to be used.

`WAN_options` (associated with the WAN_atm_vcc structure)

For the ATM protocol mode, the WAN driver performs some Operation and Maintenance Support (OAM) functions based on the *ITU-T I.610* specifications. OAM cells either will be processed by the WAN driver or passed up to the upper level for further processing (such as System Management and Performance Monitoring). This is a bit-wise OR field. The default value of the WAN_options field is zero. The following options are provided:

`WAN_AIS`

When this bit is reset (0), any OAM AIS cells that are received will be passed up to the upper level by the WAN driver.

When this bit is set (1), the WAN driver will enter the VC-AIS state when the OAM AIS cell is received. Once in the VC-AIS state, the WAN driver takes actions as defined by the *ITU-T I.610* specifications.

WAN_RDI

When this bit is reset (0), any OAM RDI cells that are received will be passed up to the upper level by the WAN driver.

When this bit is set (1), the WAN driver will enter the VC-RDI state when the OAM RDI cell is received. Once in the VC-RDI state, the WAN driver takes actions as defined by *ITU-T I.610* specifications.

WAN_RMT_CC_REQ

When this bit is reset (0), a request for activation of continuity check using an OAM cell will be denied by the WAN driver.

When this bit is set (1), a continuity check activation request will be confirmed by the WAN driver. The WAN driver will respond to an activation request for AB, BA or both directions. Actions taken are defined by the *ITU-T I.610* specifications (that is, generate CC cells if the direction is BA and monitor for CC cells in the AB direction).

WAN_RMT_CC_ACT

When this bit is reset (0), any received continuity check OAM cells will be passed up to the upper level by the WAN driver.

When this bit is set (1), the WAN driver will take action as defined by the *ITU-T I.610* specifications.

WAN_LPBK

When this bit is reset (0), the WAN driver will pass up loopback OAM cells to the upper level. When this bit is set (1), the WAN driver responds to loopback OAM cells initiated by the remote end.

WAN_RMT_PM_REQ

When this bit is reset (0), a request for activation of performance monitoring using an OAM cell will be denied by the WAN driver. Otherwise, the WAN driver will confirm the performance monitoring request as long as the direction bits indicate that the performance monitoring cells will be originated by the remote end. All other directions will be denied.

WAN_RMT_PM_ACT

When this bit is reset (0), the WAN driver passes performance monitoring OAM cells to the upper level for further processing; else it will respond to these cells.

WAN_SYS_MGMT

When this bit is reset (0), the WAN driver passes up system management OAM cells to the upper level for further processing. Otherwise, the WAN driver discards these cells.

`WAN_DATA_DLVR_FLAG`

This specifies corrupted data delivery option.

When enabled, as octets are received, the length is checked against the value specified by the WAN_crpt_sdu_dlvr_len field. If this is exceeded, ERR_F is set in the wan_aal5_rx_status field of the WAN_DAT message.

When disabled, as octets are received, the length is checked against the value specified by the WAN_maxframe field. If this is exceeded, received data is discarded, and the Segmentation and Reassembly (SAR) process is restarted. In addition, when this flag is set, the WAN_DAT message in the receive direction will contain appropriate values in the wan_aal5_rx_status field.

`WAN_CPCS`

When this bit is set, the WAN driver performs AAL5 CPCS/SAR and F5 OAM functions (based on the payload type) on the cells for this VCC. The WAN driver will calculate or check CRC-10 for F5 OAM cells. Likewise, the WAN driver will calculate or check CRC-32 for AAL5 CPCS data. This is the default.

`WAN_ATM_CELLS`

When this bit is set, the WAN driver passes up the raw ATM cells for this VCC.

`WAN_CRC10`

When this bit is set, the WAN driver performs the CRC-10 function on the cells for this VCC, which would be used for F4 OAM operations.

Error codes

0 — The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

ENODEV — Either the SNID cannot be found among the SNIDs, or the SNID format cannot be deciphered.

EINVAL — The message size does not match.

E2BIG — The host's maximum receive-buffer size is too small to hold the largest frame.

ENOMEM — Cannot allocate a single buffer for the requested frame size.

ENXIO — A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.

EIO — Command is being issued after WAN_REG.

EXDEV — Current operational mode does not match what is specified in WAN_interface.

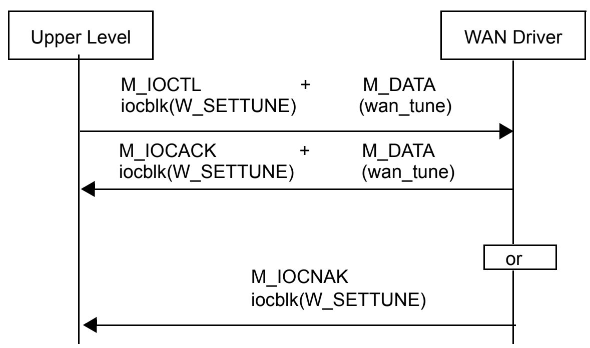
- When the cable type is selected using the W_SETTUNE command's WAN_interface parameter, the Serial WAN driver will test the cable type attached and select the interface based on the cable attached. Issue a W_GETTUNE command to verify the cable type attached.
- ATM Mode Notes:
  - When OAM cells are processed by the upper level, the WAN driver will still validate and generate a CRC-10.
  - The WAN_maxframe parameter of this command specifies the maximum size of the CPCS Service Data Unit (SDU) in octets that can be delivered to a CPCS user. If a received SDU's length exceeds this, the received data is discarded if WAN_DATA_DLVR_FLAG is reset; else, an error is flagged as ERR_E in the wan_aal5_rx_status field of the WAN_DAT message. The default value is 4100. This parameter corresponds to the MAX_SDU_Deliver_Length of the *ITU-T I.363.5* specifications.
  - WAN_CPCS, WAN_ATM_CELLS and WAN_CRC10 are mutually exclusive bits. That is, only one of them must be set.
  - *Table 5-3* on page 92 contains a summary of actions taken for various combinations of WAN_RMT_CC_REQ and WAN_RMT_CC_ACT bits.

Table 5-3. Actions taken for WAN_RMT_CC_REW and WAN_RMT_CC_ACT bits

WAN_RMT_CC_REQ	WAN_RMT_CC_ACT	Description
0	0	The WAN driver will deny a CC activation request and pass any received CC cells to the upper level.
0	1	The WAN driver will deny a CC activation request and discard any CC cells that are received.
1	0	The WAN driver will confirm a CC activation request and pass any received CC cells to the upper level.
1	1	The WAN driver will confirm a CC activation request and handle received CC cells as defined by the *ITU-T I.610* specifications.

Similar actions are taken for various settings of WAN_RMT_PM_REQ and WAN_RMT_PM_ACT bits.

Figure 5-13. Message flow for W_SETTUNE (SS7)



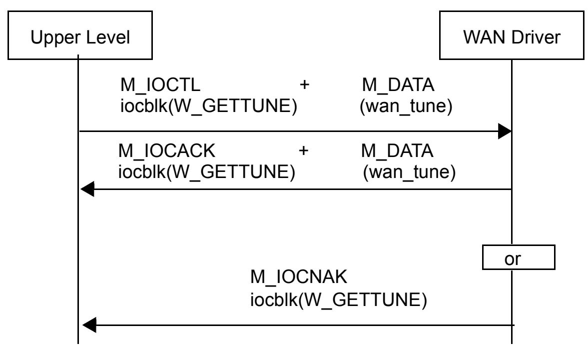
W_GETTUNE — Get configuration

This command returns the line parameters set by W_SETTUNE. The structure associated with this command is the same as W_SETTUNE. See *W_SETTUNE — Set configuration* on page *83* for details. The only exception is that the ioc_cmd field in struct iocblk should be set to W_GETTUNE.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
ENODEV	Either the SNID cannot be found among the SNIDs, or the SNID format cannot be deciphered.
EINVAL	The message size does not match.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.

Figure 5-14. Message flow for W_GETTUNE



W_SET_SNID — Allocate internal channel and associate SNID to it

This command associates a SNID to an internal channel.

The structures associated with this command are not compatible with the previous version of the Multiplexed WAN Driver (Specifications Version 1.1). A recompilation of the code is necessary for Multiplexed WAN driver-based protocol stacks.

- For the Serial WAN driver, this command attaches a SNID to a physical port.
- For the Multiplexed WAN driver, this command attaches a SNID to an internal channel that may or may not be connected to a physical-channel end port.

Usually this command is used in the clone open with SNID_DECODE=NO mode.

The following structures are associated with this command:

```
#if defined(INCLUDE_MUX) && (defined(INCLUDE_ATM)||defined(INCLUDE_SCB))
typedef union {
   uint32         chan;
   struct {
#if defined(BIG_ENDIAN_MEMORY)
     uint16        vci;
     uint16        vpi;
#else
      uint16       vpi;
      uint16       vci;
#endif
    } vpi_vci;
    struct {
#if defined(BIG_ENDIAN_MEMORY)
     uint16        tx;
     uint16        rx;
#else
      uint16       rx;
      uint16       tx;
#endif
    } rx_tx;
} vcc_chan;

#define      w_chnl_id     w_chan.chan
#define      w_vpi         w_chan.vpi_vci.vpi
#define      w_vci         w_chan.vpi_vci.vci
#define      w_chan_rx     w_chan.rx_tx.rx
#define      w_chan_tx     w_chan.rx_tx.tx
```

```
typedef union {
  uint32           port;
  uint32           pipe_id;
  struct {
#if defined(BIG_ENDIAN_MEMORY)
     uint16        tx;
     uint16        rx;
#else
     uint16        rx;
     uint16        tx;
#endif
  } rx_tx;
} vcc_port;

#define      w_pprt_id     w_port.port
#define      w_pipe_id     w_port.pipe_id
#define      w_port_rx     w_port.rx_tx.rx
#define      w_port_tx     w_port.rx_tx.tx
#endif /* INCLUDE_MUX */

struct wan_set_snid_ioc {
   uint8     w_type;
   uint8     w_spare[3];
   uint32    w_snid;
#if defined(INCLUDE_MUX) && (defined(INCLUDE_ATM) ||defined(INCLUDE_SCB))
   vcc_port  w_port;
   vcc_chan  w_chan;
#else
   uint32    w_port_id;
   uint32    w_chnl_id;
#define      w_pprt_id     w_port_id
#endif
   uint32    w_internal_id;
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_SET_SNID.

w_type
: Input. This is always WAN_SET_SNID.

w_snid
: Input. The subnetwork identifier to be assigned. Unlike other SNID inputs, this field is taken as a 32-bit quantity and is never interpreted by the driver.

`w_port_id`

Input on W_SET_SNID and Output on W_GET_SNID.

- For the Serial WAN driver, valid port numbers range from 1 to 4.
- For the Multiplexed WAN driver, valid port number ranges are described in *Figure 7-6* on page *147* and *Figure 7-7* on page *149*. If the port number is zero, the driver allocates an internal channel and the w_chnl_id field is ignored.

This can be used to perform chaining or mapping using the internal channel numbers by way of the W_SETCH_MAP command. Using this method, you can associate a SNID to a channel as follows:

- In a physical port
- In an SC-bus or CT-bus port
- Within a DSP explicitly

Also, see *Channelled mode using SC-bus and CT-bus channels for HDLC or SS7* on page *97* for a description of how this field is interpreted to accommodate SC-bus or CT-bus channels.

`w_chnl_id`

Input on W_SET_SNID and Output on W_GET_SNID.

- For the Serial WAN driver, this field is ignored.
- For the Multiplexed WAN driver, this is the channel number within the port. Valid channel number ranges are described in *Figure 7-6* on page *147* and *Figure 7-7* on page *149*. If the w_port_id specifies a DSP port, this indicates the channel on that DSP port.

Also, see *Channelled mode using SC-bus and CT-bus channels for HDLC or SS7* on page *97* for a description of how this field is interpreted to accommodate SC-bus channels.

Regarding the ARTIC960 4-Port T1/E1 Mezzanine Card, there are two IBM MWave DSP processors and each can process up to 16 channels. Therefore, if this field is set to 17, DSP number 1 (0 based) channel number 0 performs the work, and so on. Upper layers should use the GETDRVINFO command (see *W_GETDRVINFO — Get driver configuration information* on page *71*) to determine the workload of the underlying hardware (as viewed by the WAN driver).

`w_internal_id`

Input/Output. Must be set to 0 as input. On output, this is the internal channel number allocated for this SNID.

ATM, HDLC, or SS7 fat/fractional modes for the Multiplexed WAN driver

w_pprt_id
: Input. This is the value of the w_phy_pipe_id field returned by the W_SET_PHY_PIPE command.

w_vpi
: For ATM mode, this is the VPI value.

  For HDLC mode, this field must be set to zero.

w_vci
: For ATM mode, this is the VCI value.

  For HDLC mode, this field must be set to zero.

Channelled mode using SC-bus and CT-bus channels for HDLC or SS7

To accommodate the SC-bus and CT-bus channels, the Multiplexed WAN driver interprets the w_port_id and w_chnl_id fields as follows.

w_port_id
: This is broken into two fields, as follows.

  w_port_tx
  : Specify the bus wire in the transmit direction (from the WAN driver towards the bus). The following shows the possible values range, inclusive of both numbers.

    SC bus — 0x40 through 0x4f

    CT bus — 0x40 through 0x5f

  w_port_rx
  : Specify the SC-bus or CT-bus wire in the receive direction.

w_chnl_id
: This field is broken into two fields, as follows.

  w_chan_tx
  : Specify the channel within the transmit SC-bus or CT-bus wire (w_port_id field most significant 16 bits) on which the WAN driver will put the transmit data.

  w_chan_rx
  : Specify the channel within the receive SC-bus or CT-bus wire (w_port_id field least-significant 16 bits) from which the WAN driver will receive data.

Table 5-4 on page *98* summarizes all possible combinations for this command for the channelled mode for the Multiplexed WAN driver.

Table 5-4. SET_SNID command—combinations for channelled mode for Multiplexed WAN driver

w_port_id		w_chnl_id		Channels Allocated			
Tx	Rx	Tx	Rx	IC	DC	BC	Description
0	0	0	0	X			IC allocated, which is not connected to any resource. User connects this IC to IC or BC or PC using W_SETCH_MAP prior to doing a WAN_REG. If a WAN_REG is done prior to W_SETCH_MAP, error EIO is returned. DC is allocated at WAN_SID time, if it has not been allocated by the W_SETCH_MAP command.
0	0x80- 0xBF	0	0x01- 0x10	X			Connects to DC. User connects this DC to IC or BC or PC using W_SETCH_MAP prior to doing a WAN_REG; otherwise, error EIO is returned.
0	0x40- 0x4F	0	0x01- 0x20 or 0x01- 0x40	X	X		Allocates a DC and connects receive only to the BC specified by LSB of w_port_id and w_chnl_id.
0x40- 0x4F or 0x40- 0x5F	0	0x01- 0x20 or 0x01- 0x40	0	X	X		Allocates a DC and connects transmit only to the BC specified by MSB of w_port_id and w_chnl_id. See Figure 7-6 on page 147 and Figure 7-7 on page 149 for a description of valid channel number ranges.
0x40- 0x4F or 0x40- 0x5F	0x40- 0x4F or 0x40- 0x5F	0x01- 0x20 or 0x01- 0x40	0x01- 0x20 or 0x01- 0x40	X	X		Allocates a DC and connects receive to BC specified by LSB of w_port_id and w_chnl_id and connects transmit to BC specified by MSB of w_port_id and w_chnl_id. In this case, BCs within dedicated wires cannot be used. See Figure 7-6 on page 147 and Figure 7-7 on page 149 for a description of valid channel number ranges.
0	0x01- 0x04 or 0x01- 0x08	0	0x02- 0x20 or 0x01- 0x18	X	X	X	This is possible only if dedicated wires are defined during WAN driver load time. Allocates a DC, BC, and IC. See Figure 7-6 on page 147 and Figure 7-7 on page 149 for a description of valid channel number ranges.
0	0x100- 0x1FF	0x0001- 0xFFFF	0x0001- 0xFFFF	X			Allocates a VCC within this ATM pipe, with VPI and VCI specified in MSB and LSB of w_chnl_id field and the pipe ID specified in LSB of w_port_id. SNID is associated with this VCC.
0	0x100- 0x1FF	0	0	X			SNID is associated to this pipe. The pipe must be in HDLC mode.

The terms used in this table are defined as follows:

Tx	Transmit	DC	DSP Channel
Rx	Receive	BC	SC-bus or CT-bus Channel
IC	Internal Channel	PC	Physical Channel

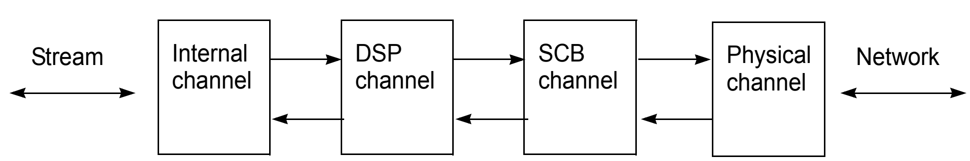
Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
EINVAL	The command size does not match.
ERANGE	Either the port number or the channel number supplied is out of range for the current hardware.
EEXIST	The SNID being assigned is not unique among those already defined.
EIO	Either the selected port is currently in remote or payload loop, or the channel is currently in use, or it is reserved (for example, channels 25 through 32 are in T1 mode).
EBUSY	The specified port or channel is currently being used.
ENOSR	All internal channels (4 or 32) have been allocated.

The following summarizes the internal workings of the Multiplexed WAN driver for this command.

- The internal channel number is allocated by the Multiplexed WAN driver and is used as a key in mapping to a DSP or SC bus, or to a physical/network port and channel.
- W_SET_SNID can be viewed as a shortcut to doing the W_SETCH_MAP command, where the allocated internal channel (w_internal_id) field is equivalent to the w_map field. The w_port_id and w_chnl_id fields are equivalent to the w_rec and w_xmt fields.
- STREAMS modules use the w_snid field to identify and operate on the data path (commands such as the WAN_REG and W_DISABLE), whereas the WAN driver uses the corresponding internal ID to map and connect various connection points so that data can flow to the ultimate destination. A data path consists of the connection points shown in *Figure 5-15*.

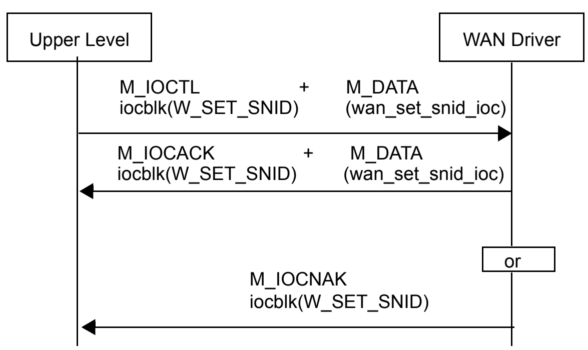
Figure 5-15. Connection points that make a data path



Each box represents a connection point.

- A mapping operation (W_SET_SNID or an entry in W_SETCH_MAP) specifies the two connection points of a data path. The w_map field in W_SETCH_MAP always specifies a full-duplex connection point.
- The Multiplexed WAN driver attempts to allocate the connection points along the data path if the mapping is of the following type:
  - Internal channel to/from SC-bus channel — An appropriate DSP channel is allocated.
  - Internal channel to/from Physical channel — A DSP channel and SC-bus or CT-bus channels are allocated if dedicated wires are defined. Otherwise, this would be an error. (The CT-bus does not require a dedicated wire.)
  - DSP channel to Physical channel — The SC-bus channel is allocated if dedicated wires are defined. Otherwise, this would be an error.
- To take complete control of this mapping, allocate only an internal channel using W_SET_SNID (w_port_id and w_chnl_id fields set to zero) and then perform mapping using the W_SETCH_MAP command.
- When in ATM cell stream mode, configure the time slots for the cell stream first (using the W_SET_PHY_PIPE command) and then issue this command to associate a SNID to a virtual channel over this cell stream.
- When in ATM cell stream mode, do not issue this command with the w_port_id and w_chnl_id fields set to zero.
- The maximum number of VCCs that can be set over an ATM pipe is limited to 8.

Figure 5-16. Message flow for W_SET_SNID



W_GET_SNID — Get the assigned internal channel ID

This command returns the internal channel assigned to a SNID, and it also returns the physical port and channel number associated with the SNID. For the Multiplexed WAN driver, this information can be used for chaining two logical channels. The structure associated with this command is the same as the one for the W_SET_SNID command, which is described in *W_SET_SNID — Allocate internal channel and associate SNID to it* on page 94.

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_GET_SNID.

w_type
: Input. This is always WAN_GET_SNID.

w_snid
: Input. The subnetwork identifier to be assigned. Unlike other SNID inputs, this field is taken as a 32-bit quantity and is never interpreted by the driver.

w_port_id, w_chnl_id, w_internal_id
: Output. The values are filled in by the driver.

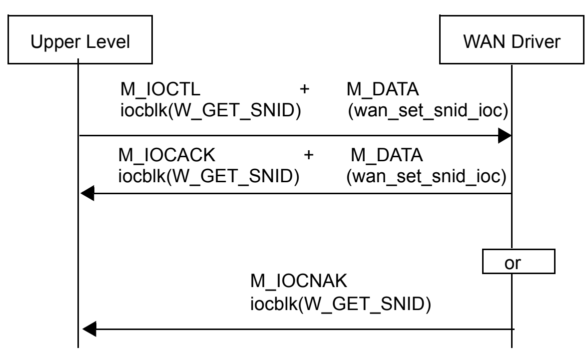
Error codes

0
: The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

EINVAL
: The command size does not match.

ENODEV
: The SNID is not found among those already defined.

Figure 5-17. Message flow for W_GET_SNID



W_REL_SNID — Release internal channel ID

This command releases the internal channel ID associated with a SNID. Usually this command is used in the clone close with SNID_DECODE=NO mode.

The following structure is associated with this command:

```
struct wan_rel_snid_ioc {
   uint8            w_type;
   uint8            w_spare[3];
   uint32           w_snid;
};
```

Parameters

IOCTL_COMMAND
Input. The ioc_cmd field in struct iocblk should be W_REL_SNID.

w_type Input. This is always WAN_REL_SNID.

w_snid Input. The subnetwork identifier to be released. Unlike other SNID inputs, this field is taken as a 32-bit quantity and is never interpreted by the Multiplexed WAN driver.

Error codes

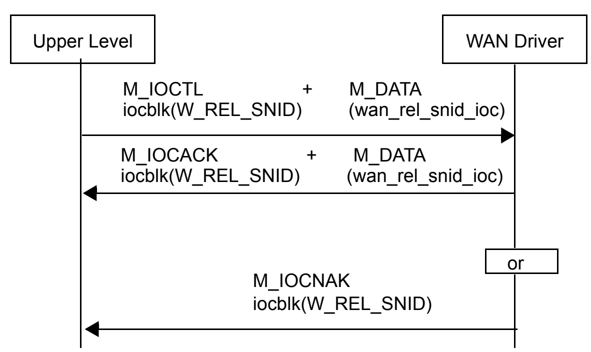
0 The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

EINVAL The command size does not match.

ENODEV The SNID being released is not found among those already defined.

EBUSY The specified port or channel is currently being used. That is, WAN_SID has been issued on this stream and a close has not been issued yet.

Figure 5-18. Message flow for W_REL_SNID



6 Signaling System Number 7 (SS7) (specific operations)

This chapter provides information related to operations specific to Signaling System Number 7 (SS7). The Serial synchronous and the Multiplexed WAN drivers support SS7 low-level processing protocol.

The SS7 functions require the creation of STREAMS service messages and management commands for both the Serial and Multiplexed WAN drivers. The format of these messages and commands is based on the existing ones.

Table 6-1. STREAMS service messages and management commands for SS7

Message	Use	Type	Direction	Page
WAN_ACTSS7	To control the SS7 features: activation/deactivation and ERM	Service Message (M_PROTO)	Down on any appropriate stream	105
WAN_NOTIFSS7	To inform the upper level of SS7 and ERM generated events	Service Message (M_PROTO)	Up on any appropriate stream	107
WAN_RESETSS7	To reset the FISU/LSSU filtering. See *SU filtering* on page *13* for a description of the reset operation.	Service Message (M_PROTO)	Down on any appropriate stream	109
W_SETSS7	To set the ITU-T/ANSI SS7 attributes of a logical channel ERM type, counter's threshold, ERM parameters	Management Command (M_IOCTL)	Down on any opened stream	112
W_GETSS7	To obtain the ITU-T/ANSI SS7 attributes of a logical channel	Management Command (M_IOCTL)	Down on any opened stream	115
W_SETSS7_JPN	To set the TTC SS7 attributes of a line or channel, ERM parameters, transmission time intervals	Management Command (M_IOCTL)	Down on any opened stream	117
W_GETSS7_JPN	To obtain the TTC SS7 attributes of a logical channel	Management Command (M_IOCTL)	Down on any opened stream	120

Relation between SS7 and HDLC modes

An SS7 link is operated on top of regular HDLC protocol. The SS7 functions on a logical channel are activated using the W_SS7_START action of the WAN_ACTSS7 service message. This message must precede all SS7-related messages on that stream and all SS7 management commands for that SNID. The SS7 mode is exited when the W_SS7_STOP action on the WAN_ACTSS7 service message is received.

STREAMS service messages for SS7

Message	Structure in M_PROTO	M_DATA?	Direction
WAN_ACTSS7	wan_actss7 (see *WAN_ACTSS7 — Control SS7 features* on page *105*)	No	To WAN driver
WAN_NOTIFSS7	wan_notifss7 (see *WAN_NOTIFSS7 — Notify SS7 status* on page *107*)	No	From WAN driver
WAN_RESETSS7	wan_resetss7 (see *WAN_RESETSS7 — Reset filtering operation* on page *109*)	Yes, Filter count + SU	To WAN driver

WAN_ACTSS7 — Control SS7 features

This message performs SS7-related actions on a stream. The possible actions are:

- Selecting ITU-T/ANSI SS7, Clear Channel Capability, or TTC SS7 mode or function
- Starting and stopping SUERM
- Starting and stopping AERM in normal or emergency mode
- Starting and stopping EIM in Clear Channel Capability mode
- Selecting the alignment mode (normal or emergency) when AERM is in *Idle* or *Monitoring* state.

Normally, a service message does not need a response. However, this message is returned as a confirmation after the appropriate action has been taken. Errors are reported in the w_error field.

The following structure is associated with this command:

```
struct wan_actss7 {
   uint8          w_type;
   uint8          w_spare[3];
   uint32         w_action;
   uint32         w_error;
};
```

Parameters

w_type Input. This is always WAN_ACTSS7.

w_action Input. The action requested on that SS7 link. The allowed values are:

W_SS7_START
Select or activate ITU-T/ANSI SS7 mode.

W_JSS7_START
Select or activate TTC SS7 mode.

W_CCC_START
Select or activate ITU-T/ANSI SS7 Clear Channel Capability mode.

W_SS7_STOP
Deselect or deactivate SS7 mode or TTC SS7 mode.

W_SS7_START_AERM
Start Alignment ERM.

W_SS7_SET_TIN
Put Alignment ERM in normal mode.

W_SS7_SET_TIE
Put Alignment ERM in emergency mode.

W_SS7_START_SUERM
Start Signal Unit ERM.

`W_SS7_STOP_AERM`
: Stop AERM.

`W_SS7_STOP_SUERM`
: Stop SUERM.

`W_JSS7_INC_SUERM_COUNTER`
: Increment SUERM counter.

`W_EIM_START`
: Start EIM ERM.

`W_EIM_STOP`
: Stop EIM ERM.

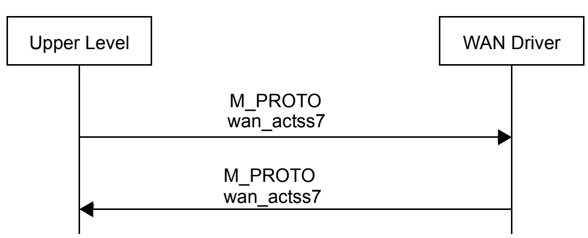
w_error Output. Contains the error codes defined in the following section.

Error codes

0	No error if command was successfully processed.
EINVAL	Either the message size does not match or the action is invalid.
EIO	The line or channel is in the wrong state.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.
EACCES	SS7 mode is not activated on this line or channel; that is, W_SS7_START was not issued, was started twice, or was stopped without starting.

- Switching between HDLC and SS7 modes is not allowed when a stream is opened (issuing W_SS7_START and W_SS7_STOP).
- When w_action is W_SS7_SET_TIN or W_SS7_SET_TIE, the logic resets the current error counter (Ca) to zero.

Figure 6-1. Message flow for WAN_ACTSS7



WAN_NOTIFSS7 — Notify SS7 status

This message notifies the upper level of events related to a SS7 link. Reported events are the following:

- Link failure due to SUERM or EIM threshold surpassed
- Abort proving due to AERM threshold surpassed
- Number of errored signal units at a periodic time interval

This message is sent on the stream that is associated with the SS7 link.

The following structure is associated with this M_PROTO message:

```
typedef struct wan_ss7_stats {
   uint32           wan_ss7_su_err_cnt ;
} wan_ss7_stats_t ;

struct wan_notifss7 {
   uint8            wan_type;
   uint8            wan_spare[3];
   uint32           wan_event;
   uint32           wan_diag;
   wan_ss7_stats_t  wan_ss7_info;
};
```

Parameters

wan_type Output. This is set to WAN_NOTIFSS7.

wan_event

Output. This indicates the events being reported. This is a bit-wise OR of the following values:

WAN_SS7_LINK_FAIL
A failure of the link due to the SUERM threshold being surpassed.

WAN_SS7_ABRT_PROV
An abort proving due to the AERM threshold being surpassed. After reporting this event, the AERM logic resets the Ca to zero and reenters the *monitoring* mode.

WAN_SS7_ERM_STATS
Number of signal units that were received in error during the past intervals. Field wan_ss7_su_err_cnt in wan_ss7_info is nonzero when this bit is set, that is, this event is reported only when the count of errored signal units is nonzero. This count is cleared once the event is reported.

wan_diag Output. For the Multiplexed WAN driver, this field reports additional information codes or reasons for failure. See the description for the wan_event field in *WAN_NOTIFDI — Inform upper level of T1/E1 events* on page *129* for status bits reported. Note that only the lower 8 bits of the wan_event field will be reported in wan_diag because it is only 8-bits wide.

`wan_ss7_info`

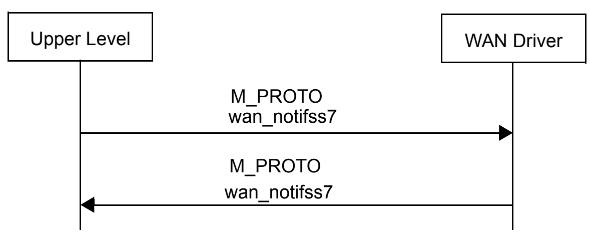
Output.

`wan_ss7_su_err_cnt`

Output. The driver maintains a count of the number of signal units that were received in error as defined by the Alignment and Signal Unit Error Rate Monitors. This count is cleared when the WAN_SS7_ERM_STATS event is reported.

- SUERM goes to the idle state when the event WAN_SS7_LINK_FAIL is reported. The upper layer must issue a W_SS7_START_SUERM to get it started again. In this case, if a W_GETSS7 is issued, w_erm_type is set to NO_ERM_RUNNING. Also, note that the OCM logic remains unaffected by these events, meaning WAN_SS7_ERM_STATS will be reported if errored SUs are detected, even though none of the Error Rate Monitors are active.
- The AERM resets the counter Ca to zero and reenters the *monitoring* state after WAN_SS7_ABRT_PROV is reported. A W_SS7_START_AERM need not be issued.

Figure 6-2. Message flow for WAN_NOTIFSS7



WAN_RESETSS7 — Reset filtering operation

This service message resets the FISU or LSSU filtering temporarily. In this way, the current FISU or LSSU is passed to the upper level for processing. The WAN driver acknowledges this message by updating the wan_error and wan_reset_status fields. An M_DATA message follows this acknowledgement if the reset operation was successful; that is, wan_reset_status is zero. This command forces the filtering process to restart. Hence, the filter count is reset to zero and the next incoming SU is sent up the stream and, from then on, the filtering process starts.

The following structure is associated with this M_PROTO message:

```
struct wan_resetss7 {
   uint8            wan_type;
   uint8            wan_spare[3];
   uint32           wan_reset_type;
   uint32           wan_error ;
   uint32           wan_reset_status ;
};
```

Parameters

wan_type Input. This is set to WAN_RESETSS7

wan_reset_type

Input. The type of filtering reset requested. The allowed values are:

WAN_RST_FISU
Reset FISU filtering.

WAN_RST_LSSU
Reset LSSU filtering.

wan_error

Output. Contains error codes, defined in the following section. In case of error, M_DATA message will not follow.

0	No error. Check wan_reset_status for more information.
EIO	The line or channel is in the wrong state.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.
EACCES	SS7 mode is not activated on this line or channel.
EINVAL	Either wan_reset_type contains an invalid value or the message size does not match.

`wan_reset_status`

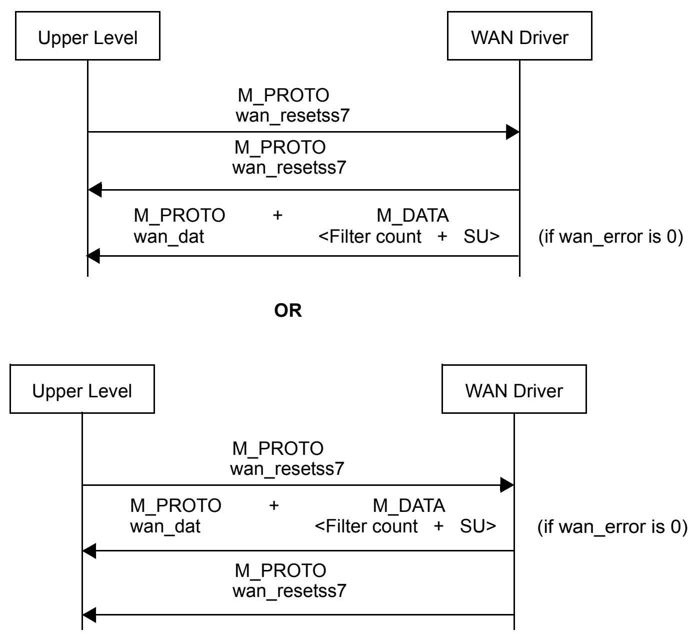
Output.

0 — Reset operation was completed without errors. M_DATA message containing the filter count and SU data will follow.

WAN_RESET_INVALID — Reset operation failed because the requested SU is not being filtered currently.

WAN_RESET_CNT_ZERO — So far, the requested SU has been seen only once by the WAN driver; hence, the current filter count is zero.

Figure 6-3. Message flow for WAN_RESETSS7



STREAMS management commands for SS7

ioc_cmd value of iocblk structure in M_IOCTL	Structure in M_DATA after M_IOCTL	M_DATA with M_IOCACK?
W_SETSS7	wan_setss7_ioc (see *W_SETSS7 — Set SS7 configuration parameters* on page *112*)	No
W_GETSS7	wan_getss7_ioc (see *W_GETSS7 — Get SS7 configuration parameters* on page *115*)	Yes, wan_getss7_ioc
W_SETSS7_JPN	wan_jpn_setss7_ioc (see *W_SETSS7_JPN — Set TTC SS7 configuration parameters* on page *117*)	No
W_GETSS7_JPN	wan_jpn_getss7_ioc (see *W_GETSS7_JPN — Get TTC SS7 configuration parameters* on page *120*)	Yes, wan_jpn_getss7_ioc
W_SETSS7_CCC	wan_ccc_setss7_ioc (see *W_SETSS7_CCC — Set SS7 Clear Channel Capability configuration parameters* on page *122*)	No
W_GETSS7_CCC	wan_ccc_getss7_ioc (see *W_GETSS7_CCC — Get SS7 Clear Channel Capability configuration parameters* on page *125*)	Yes, wan_ccc_getss7_ioc

W_SETSS7 — Set SS7 configuration parameters

This management command is used to configure different ITU/ANSI SS7 attributes of a line or channel. Configurable SS7 attributes are the following:

- SUERM counter threshold — T — defaults to 64
- Normal AERM counter threshold — Tin — defaults to 4
- Emergency AERM counter threshold — Tie — defaults to 1
- Number of good/erroneous SUs that needs to be received to decrement the SUERM counter — D — defaults to 256
- Number of octets needed in Octet Counting Mode before the *SU in Error* notification is generated — N — defaults to 16

The following structure is associated with this command:

```
typedef struct wan_ss7_info {
   uint16          w_erm_type ;
   uint16          w_suerm_cntr ;
   uint16          w_aerm_cntr ;
   uint16          w_ocm ;
   uint16          w_thres_T ;
   uint16          w_thres_Tin ;
   uint16          w_thres_Tie ;
   uint16          w_param_D ;
   uint16          w_param_N ;
   uint16          w_spare ;
} wan_ss7_info_t ;

struct wan_setss7_ioc {
   uint8              w_type;
   uint8              w_spare[3];
   uint32             w_snid;
   wan_ss7_info_t     w_ss7;
};
```

Parameters

IOCTL_COMMAND
Input. The ioc_cmd field in struct iocblk should be W_SETSS7.

w_type Input. This is always WAN_SETSS7.

w_snid Input. The subnetwork identifier. See the description of the wan_snid parameter on page 51.

w_ss7 Input. The structure describing the parameters to be set. The following fields are defined for the structure:

w_erm_type, w_suerm_cntr, w_aerm_cntr, w_ocm
Output. These fields are used only by the W_GETSS7 command. They are not used by the W_SETSS7 command.

`w_thresh_T`

Input. Threshold for SUERM counter. The default depends on how WAN_ACTSS7 was issued.

Issued with	Default
W_SS7_START	64
W_JSS7_START	285

`w_thresh_Tin`

Input. Threshold for AERM counter in normal alignment. The default value is 4.

`w_thresh_Tie`

Input. Threshold for AERM counter in emergency alignment. The default value is 1.

`w_param_D`

Input. The **D** parameter of SUERM. The default depends on how WAN_ACTSS7 was issued. To override the default value, issue the command again.

Issued with	**Description and Default**
W_SS7_START	Specifies the number of good or erroneous SUs that need to be received to decrement the SUERM counter. The default is 256.
W_JSS7_START	Specifies the number to be added to the SUERM counter when an erroneous SU is received. The default is 16.

`w_param_N`

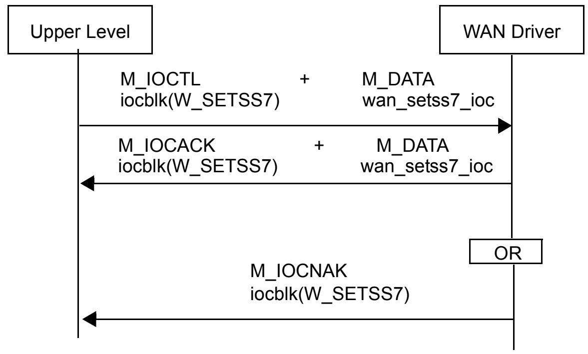
Input. The **N** parameter used when determining the *SU error* notification. When in Octet Counting Mode, the SU in error notification to ERM is generated every **N** octets. The default value is 16.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
ENODEV	Either the SNID cannot be found among the SNIDs, or the SNID format cannot be deciphered.
EINVAL	The message size does not match.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.
EACCES	SS7 has not been activated on the line or channel.
EIO	Command issued while in data transfer state.

The message flow is shown in *Figure 6-4*.

Figure 6-4. Message flow for W_SETSS7



W_GETSS7 — Get SS7 configuration parameters

This management command is used to obtain the different ITU-T/ANSI SS7 attributes of a logical channel. In addition to the attributes listed in *W_SETSS7 — Set SS7 configuration parameters* on page *112*, this command also obtains the following:

- Type of ERM
- Values of SUERM and AERM counters
- Octet Counting Mode status

The following structure is associated with this command:

```
struct wan_getss7_ioc {
   uint8              w_type;
   uint8              w_spare[3];
   uint32             w_snid;
   wan_ss7_info_t     w_ss7;
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_GETSS7.

w_type
: Input. This is always WAN_INFOSS7.

w_snid
: Input. The subnetwork identifier. See the description of the wan_snid parameter on page *51*.

w_ss7
: Output. The structure describing the attributes of the SS7 line or channel. Certain members of this structure are the same as the one described in *W_SETSS7 — Set SS7 configuration parameters* on page *112*. Additional structure members are described here.

  w_erm_type
  : Output. The type of ERM currently in operation on the signalling link. This can be either:

    - SU_ERM
    - A_NORM
    - A_EMERG
    - NO_ERM_RUNNING

  w_suerm_cntr
  : Output. Value of SUERM counter.

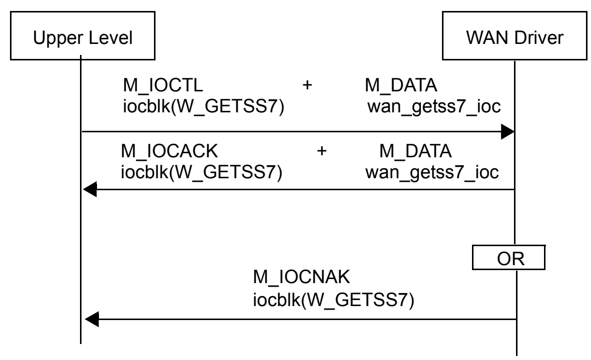
  w_aerm_cntr
  : Output. Value of AERM counter.

  w_ocm
  : Output. Status of Octet Counting Mode: OCM_ACTIVE (for active) and OCM_INACTIVE (for inactive).

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
ENODEV	Either the SNID cannot be found among the SNIDs or the SNID format cannot be deciphered.
EINVAL	The message size does not match.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.
EACCES	SS7 has not been activated on the line or channel.

Figure 6-5. Message flow for W_GETSS7



W_SETSS7_JPN — Set TTC SS7 configuration parameters

This management command is used to configure different TTC SS7 attributes of a line or channel. Configurable SS7 attributes are the following:

- SUERM counter threshold — T — defaults to 285
- Emergency AERM counter threshold — Tie — defaults to 1
- Number of good/erroneous SUs that needs to be received to decrement the SUERM counter — D — defaults to 256
- Number of octets needed in Octet Counting Mode before the *SU in Error* notification is generated — N — defaults to 16

The following structure is associated with this command:

```
typedef struct wan_jpn_ss7_info
{
   uint32             w_param_Ts ;
   uint32             w_param_Ps ;
   uint32             w_param_To ;
   uint32             w_param_Ta ;
   uint32             w_param_Tf ;
   uint32             w_param_Te ;
   uint32             w_OCM_flag ;
} wan_jpn_ss7_info_t ;

struct wan_jpn_setss7_ioc
{
   uint8              w_type ;
   uint8              w_spare[3] ;
   uint32             w_snid ;
   wan_jpn_ss7_info_t w_jpn_ss7 ;
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_SETSS7_JPN.

w_type
: Input. This is always WAN_SETSS7_JPN.

w_snid
: Input. The subnetwork identifier. See the description of the wan_snid parameter on page *51*.

w_jpn_ss7

Input. The structure describing the parameters to be set. The following fields are defined for the structure:

w_param_Ts

Input. This sets the time interval between transmission of an SIOS (Status Indicator Out of Service), which is a type of LSSU with Status Field set to 0x03. Idle flags (0x7e) are transmitted in between these SIOSs.

Default	Range	Precision
24 milliseconds	20–100 milliseconds	1 millisecond

w_param_Ps

Input. This defines the maximum period that SIOSs are transmitted.

Default	Range	Precision
3 milliseconds	1–100000 seconds	100 milliseconds

w_param_To

Input. This sets the time interval between transmission of an SIO (Service Information Out of Alignment), which is a type of LSSU with Status Field set to 0x00. Idle flags (0x7e) are transmitted in between these SIOs.

Default	Range	Precision
24 milliseconds	20–100 milliseconds	1 millisecond

w_param_Ta

Input. This sets the time interval between transmission of an SIE (Status Indicator Emergency), which is a type of LSSU with Status Field set to 0x02. Idle flags (0x7e) are transmitted in between these SIEs.

Default	Range	Precision
24 milliseconds	20–100 milliseconds	1 millisecond

w_param_Tf

Input. This sets the time interval between transmission of an FISU. Idle flags (0x7e) are transmitted in between these FISUs.

Default	Range	Precision
24 milliseconds	20–100 milliseconds	1 millisecond

w_param_Te

Input. This sets the time interval for checking the ERMs. A value of 0 (zero) indicates that the timer is not to be used.

Default	Range	Precision
24 milliseconds	20–100 milliseconds	1 millisecond
	0–Do not use timer.	

w_OCM_flag

Input.

0 Indicates normal OCM operation, where *SU in error* indications are generated upon receipt of every 16 octets (once OCM is triggered) until a good SU is received. This is the default.

1 Indicates that *SU in error* indications are not generated upon receipt of every 16 octets (once OCM is triggered).

Error codes

0 The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

ENODEV Either the SNID cannot be found among the SNIDs, or the SNID format cannot be deciphered.

EINVAL The message size does not match.

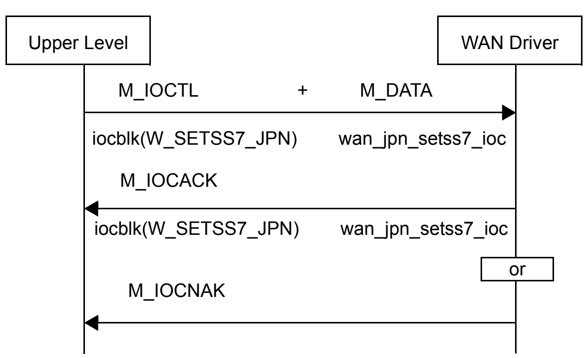
ENXIO A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.

EACCES SS7 has not been activated on the line or channel.

EIO Command issued while in data transfer state.

The message flow is shown in *Figure 6-6*.

Figure 6-6. Message flow for W_SETSS7_JPN



W_GETSS7_JPN — Get TTC SS7 configuration parameters

This management command is used to obtain the current settings of the TTC SS7-specific configuration parameters from the WAN driver.

The following structure is associated with this command:

```
struct wan_jpn_getss7_ioc

{
   uint8                      w_type;
   uint8                      w_spare[3];
   uint32                     w_snid;
   wan_jpn_ss7_info_t         w_jpn_ss7;
};
```

See *W_SETSS7_JPN — Set TTC SS7 configuration parameters* on page *117* for a description of the wan_jpn_ss7_info_t structure.

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_GETSS7_JPN.

w_type
: Input. This is always WAN_GETSS7_JPN.

w_snid
: Input. The subnetwork identifier. See the description of the wan_snid parameter on page *51*.

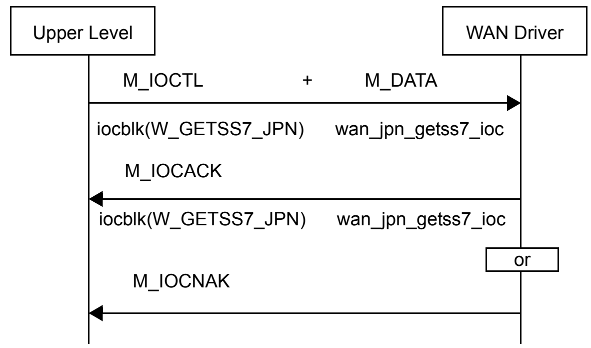
w_jpn_ss7
: Output. The structure describing the attributes of the TTC SS7 line or channel. See *W_SETSS7_JPN — Set TTC SS7 configuration parameters* on page *117* for a description of this structure. The current settings are returned.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
ENODEV	Either the SNID cannot be found among the SNIDs or the SNID format cannot be deciphered.
EINVAL	The message size does not match.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.
EACCES	SS7 has not been activated on the line or channel.
EIO	Command issued while in data transfer state.

The message flow is shown in *Figure 6-7*.

Figure 6-7. Message flow for W_GETSS7_JPN



W_SETSS7_CCC — Set SS7 Clear Channel Capability configuration parameters

To ensure this command is supported on your adapter, contact your RadiSys representative.

This management command is used to configure different ITU/ANSI SS7 attributes of a line or channel for Clear Channel Capability mode. Configurable SS7 attributes are the following:

- Normal AERM counter threshold — Tin — defaults to 4
- Emergency AERM counter threshold — Tie — defaults to 1
- Number of octets needed in Octet Counting Mode before the *SU in Error* notification is generated — N — defaults to 16
- EIM counter threshold — Te
- EIM upcount — Ue
- EIM downcount — De
- EIM monitoring interval — T8

The following structure is associated with this command:

```
typedef struct wan_ccc_ss7_info {
   uint32           w_eim_cntr ;
   uint32           w_ccc_Te ;
   uint32           w_ccc_U3 ;
   uint32           w_ccc_De ;
   uint32           w_ccc_T8 ;
   uint16           w_ccc_esnf ;
   uint16           w_ocm_enable ;
   uint16           w_erm_type ;
   uint16           w_aerm_cntr ;
   uint16           w_ocm ;
   uint16           w_thres_Tin ;
   uint16           w_thres_Tie ;
   uint16           w_param_N ;
} wan_ccc_ss7_info_t ;

struct wan_ccc_setss7_ioc {
   uint8               w_type;
   uint8               w_spare[3];
   uint32              w_snid;
   wan_ccc_ss7_info_t  w_ss7;
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_SETSS7_CCC.

w_type
: Input. This is always WAN_SETSS7_CCC.

w_snid
: Input. The subnetwork identifier. See the description of the wan_snid parameter on page *51*.

`w_ss7` Input. The structure describing the parameters to be set. The following fields are defined for the structure:

`w_eim_cntr`
Output. This field is used only by the W_GETSSU_CCC command. It is not used by the W_SETSS7_CCC command.

`w_ccc_Te` Input. Number of intervals where signal units have been received in error that will cause an error rate high indication to level 3. The default is as follows:

Default for T1 mode (1.5 MBit/s)	Default for E1 mode (2.0 MBit/s)
577169	793544

`w_ccc_U3` Input. Constant for incrementing the EIM counter. :

Default for T1 mode (1.5 MBit/s)	Default for E1 mode (2.0 MBit/s)
144292	198384

`w_ccc_De` Input. Constant for decrementing the EIM counter.:

Default for T1 mode (1.5 MBit/s)	Default for E1 mode (2.0 MBit/s)
930	11328

`w_ccc_T8` Input. Timer interval (in milliseconds) for monitoring errors.:

Default for T1 mode (1.5 MBit/s)	Default for E1 mode (2.0 MBit/s)
100 ms	100 ms

`w_ccc_esnf`
Input. Specifies whether extended sequence number format is used. This can be either:

W_ESNF_YES
Extended sequence number format is used.

W_ESNF_NO
Extended sequence number format is not used. (Default)

`w_ocm_enable`
Input. Specifies whether OCM logic is enabled.

W_OCM_ENABLED
Enables OCM logic and applies it to AERM only. (Default)

W_OCM_DISABLED
Disables OCM logic.

w_erm_type, w_aerm_cntr, w_ocm

Output. These fields are used only by the W_GETSS7_CCC command. They are not used by the W_SETSS7_CCC command.

w_thresh_Tin

Input. Threshold for AERM counter in normal alignment. The default value is 4.

w_thresh_Tie

Input. Threshold for AERM counter in emergency alignment. The default value is 1.

w_param_N

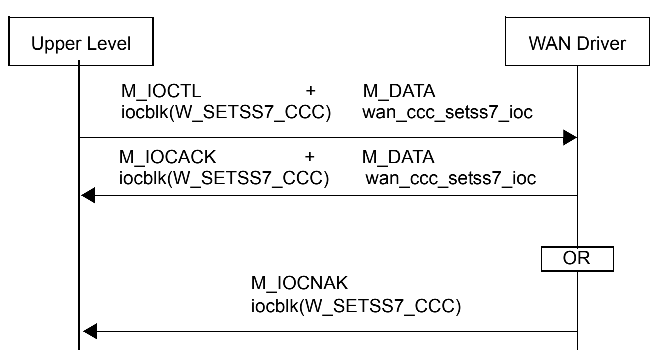
Input. The **N** parameter used when determining the SU error notification. When in Octet Counting Mode, the SU in error notification to ERM is generated every **N** octets. The default value is 16.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
ENODEV	Either the SNID cannot be found among the SNIDs, or the SNID format cannot be deciphered.
EINVAL	The message size does not match.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.
EACCES	SS7 has not been activated on the line or channel.
EIO	Command issued while in data transfer state.

The message flow is shown in *Figure 6-8*.

Figure 6-8. Message flow for W_SETSS7_CCC



W_GETSS7_CCC — Get SS7 Clear Channel Capability configuration parameters

To ensure this command is supported on your adapter, contact your RadiSys representative.

This management command is used to obtain the different ITU-T/ANSI SS7 attributes of a logical channel for Clear Channel Capability mode. In addition to the attributes listed in *W_SETSS7_CCC — Set SS7 Clear Channel Capability configuration parameters* on page *122*, this command also obtains the following:

- Type of ERM
- Values of EIM and AERM counters
- Octet Counting Mode status

The following structure is associated with this command:

```
struct wan_getss7_ccc_ioc {
   uint8             w_type;
   uint8             w_spare[3];
   uint32            w_snid;
   wan_ss7_info_t    w_ss7;
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_GETSS7_CCC.

w_type
: Input. This is always WAN_INFOSS7_CCC.

w_snid
: Input. The subnetwork identifier. See the description of the wan_snid parameter on page *51*.

w_ss7
: Output. The structure describing the attributes of the SS7 line or channel. Certain members of this structure are the same as the one described in *W_SETSS7_CCC — Set SS7 Clear Channel Capability configuration parameters* on page *122*. Additional structure members are described here.

  w_erm_type
  : Output. The type of ERM currently in operation on the signalling link. This can be either:

    - SU_ERM
    - A_NORM
    - A_EMERG
    - NO_ERM_RUNNING

w_eim_cntr
: Output. Value of EIM counter.

w_aerm_cntr
: Output. Value of AERM counter.

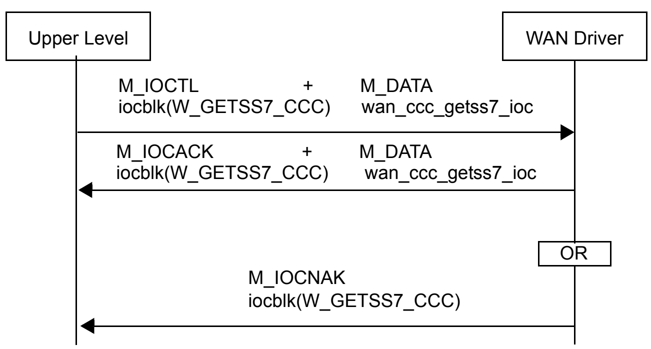
w_ocm
: Output. Status of Octet Counting Mode. This can be either:
- OCM_ACTIVE (for active)
- OCM_INACTIVE (for inactive).

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
ENODEV	Either the SNID cannot be found among the SNIDs or the SNID format cannot be deciphered.
EINVAL	The message size does not match.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.
EACCES	SS7 has not been activated on the line or channel.

The message flow is shown in *Figure 6-9*.

Figure 6-9. Message flow for W_GETSS7_CCC



7 T1/E1 interface (specific operations)

This chapter provides information related to operations specific to the T1/E1 interface (multiplexed mode). Complete T1/E1 control implies the creation of new STREAMS service messages and management commands for the Multiplex WAN driver. The format of these new messages and commands is based on the existing ones. In this chapter, *port*, *physical port*, or *digital interface* refers to a T1/E1 port.

Table 7-1. STREAMS service message and management commands for the T1/E1 interface

Message	Use	Page
Type: Service Message (M_PROTO) — Direction: up on all streams		
WAN_NOTIFDI	To inform the upper level of T1/E1 events from a particular port	129
WAN_NOTIFTIM	To send a timestamped notification	134
Type: Management Command (M_IOCTL) — Direction: down on any opened stream		
W_DI_TEST_CFG	To set up the hardware to generate certain test conditions, such as alarm simulation on a physical port	137
W_SET_PHY_PIPE	To define and undefine the time slots associated with a physical data pipe	140
W_GET_PHY_PIPE	To obtain time slot information associated with a physical data pipe	144
W_SETCH_MAP	To connect channels	146
W_GETCH_MAP	To obtain the current mapping of channels	156
W_SETDI	To set the attributes common to all digital interfaces	158
W_GETDI	To get the attributes common to all digital interfaces	164
W_SETDI_PORT	To set the attributes of one of the physical ports	165
W_GETDI_PORT	To get the attributes of a physical port	174
W_GETDI_STATS	To obtain the statistics from a physical port	176
W_ZERODI_STATS	To clear the statistics of a physical port	178
W_SETDI_LPBK	To place a port in loopback mode	179
W_SETDI_NOTIF	To enable/disable the notifications for T1/E1 events and alarms from a particular port	181
W_SET_TIMESTAMP	To set the current value of the timestamp	183

Identifying the T1/E1 components

Access to the T1/E1 components can be done at three levels:

- Global level — where all ports and channels are concerned. No identifier needed.
- Port level — to access the parameters/statistics of a single port. The port identifier is from 1 to 4.
- Channel level — to access one of the channels on a port. Channel numbers in this document are one-based, whereas time slots in various ITU-T publications are zero-based. Channel number 1 for T1 refers to time slot 0 and so on, whereas channel number 2 for E1 refers to time slot 1. (Time slot 0 for E1 is reserved for alignment, and so forth.)
  - For a T1 port, channels are from 1 to 24
  - For an E1 port, channels are from 2 to 32

To prevent duplication of messages, a group of *DI* (Digital Interface) messages are created that apply to both T1 and E1 carrier types.

STREAMS service messages for T1/E1

Message	Structure in M_PROTO	M_DATA?	Direction
WAN_NOTIFDI	wan_notifdi (see *WAN_NOTIFDI — Inform upper level of T1/E1 events* on page *129*)	No	From WAN driver
WAN_NOTIFTIM	wan_notiftim (see *WAN_NOTIFTIM — Send a timestamped notification* on page *134*)	No	From WAN driver

WAN_NOTIFDI — Inform upper level of T1/E1 events

This service message informs the upper level of events related to one of the digital interfaces. A message will be generated for each condition as it appears and disappears. The WAN driver will send the message on:

- All streams associated with the port; that is, all streams that received a WAN_SID command with the same port ID.
- All streams that have no association with a port; that is, all streams that did not receive a WAN_SID command (also called a *management stream*.

The following events are physical-port-specific, and the wan_port_id field is set to the physical port ID that is generating these events:

- W_DI_FAR_RAI
- W_DI_FAR_AIS
- W_DI_LOS
- W_DI_CLK_CHANGED
- W_DI_TX_SHORT
- W_DI_TX_OPEN
- W_DI_FAR_LMFA
- W_DI_FAR_LFA
- W_DI_SLN
- W_DI_SLP

For all other events, the wan_port_id field is set to GLOBAL_PORT.

The W_SETDI_NOTIF command uses a similar method to enable notification.

The following structure is associated with this M_PROTO message:

```
struct wan_notifdi  {
   uint8            wan_type;
   uint8            wan_spare[3];
   uint32           wan_port_id;
   uint32           wan_event;
   uint32           wan_other_event;
   uint32           wan_status;
   uint32           wan_curr_clk_src ;
};
```

Parameters

wan_type This is set to WAN_NOTIFDI.

wan_port_id
The number of the port (from 1 to 4) where the event occurred.

wan_event

This indicates the events being reported. This is a bit-wise OR of the following values:

W_DI_FAR_RAI
Far-end alarm (yellow alarm for T1 and distant alarm for E1)

W_DI_FAR_AIS
Alarm Indication Signal (AIS) failure.

W_DI_LOS Failure of Loss of Signal (LOS).

W_DI_CLK_CHANGED
The current clock source has failed, and the new source is the one defined in the wan_curr_clk_src field. This notification is issued for the port that loses the clock, as well as for the one that becomes the current master.

W_DI_TX_SHORT
For ternary-line interface. Indicates a short on transmit lines.

W_DI_TX_OPEN
Indicates an open on transmit lines.

W_DI_FAR_LMFA
Loss of multiframe alignment. This is applicable in:

- E1 mode, when a multiframe format is chosen, or
- T1 mode, when super-frame format is chosen.

W_DI_FAR_LFA
Loss of frame alignment.

W_DI_SLN Negative slip. The frequency of the receive route clock is greater than the provided one. A frame will be skipped.

W_DI_SLP Positive slip. The frequency of the receive route clock is less than the one provided. A frame will be repeated.

W_DI_XSLP Transmit slip. This event is reported when the communications chip detects a wandering in the clocks. This would occur when the source of the clock is changed (due to LOS or programming a different source). This event will occur in T1 mode.

WAN_OTHER This bit is set when any of the global events are set. When this bit is used in the w_event field of W_SETDI_NOTIF or in the WAN_event_disc field of W_SETTUNE, it either enables or disables all global events. If one needs to selectively access global events, this bit should be off. See the following `wan_other_event` field for details.

wan_other_event

These are global events associated with GLOBAL_PORT, and are reported when the WAN_OTHER bit is set in the wan_event field. This is a bit-wise OR of the following values:

W_DI_CABLE_MISMATCH

Cable mismatch on this port (unknown or a mismatch). The WAN driver will check the cable type every second to determine if the problem is corrected.

W_DI_DSP_ERROR

Unexpected interrupt or return codes were received from the DSP microcode. The WAN driver may or may not recover from this event.

W_DI_SCBUS_MASTER

Reported when the card gains or loses the mastership of the SC bus. For compatibility with the previous version of header files, this bit is equated to W_DI_SCBUS_CLK_FAIL. For a discussion about SC-bus mastership and how a failure of the SC-bus clock affects it, see *W_SETDI — Set attributes common to all digital interfaces* on page *158*.

W_DI_NET_ERROR

This bit is set when the network switch detects a clocking error while it was master on the SC bus. This may be a result of two masters on the bus. This event does not cause the data streams to get WC_DISC.

W_DI_NET_CLOCK

This bit is set when the network switch detects a long clock failure (more than four SC-bus clock periods). Normally, if an Armed Master is on the bus and the clock fails, the Armed Master should take over within four SC-bus clock periods, and none of the switches on the SC bus should detect the failure. This event does not cause the data streams to get WC_DISC.

W_DI_NET_CONFLICT

This bit is set when the network switch detects that one of its channels is transmitting to an SC-bus time slot while some other switch on the bus also is actively transmitting to the same time slot. The WAN driver immediately disconnects all channels from that SC-bus wire until the condition goes away. It then will try to reconnect all of the channels. If the user application takes no corrective action, the result could be many repeated notifications. This event does not cause the data streams to get WC_DISC.

The WAN driver prevents such a conflict from occurring for all channel assignments (for both network and processing switches).

W_DI_DATA_CLOCK

A loss of clock was detected on the processing switch. Comments similar to those for W_DI_NET_CLOCK apply. However, if this event is enabled in the WAN_event_disc field of W_SETTUNE, it will cause a WC_DISC for those streams, just as with other port-dependent events.

W_DI_DATA_CONFLICT

A conflict was detected on the processing switch. Comments similar to those for W_DI_NET_CONFLICT apply. Also, this event can cause a WC_DISC, if enabled.

wan_status

The status of the event. The allowed values are:

WAN_EVT_DETECTED

The following events or alarms have just been detected.

WAN_EVT_RELEASED

These events/alarms have gone away. For events W_DI_SLN and W_DI_SLP, a WAN_EVT_ RELEASED is not generated.

wan_curr_clk_src

This field always reflects the current source of the clock. Possible values are listed in the w_master_clk field of the W_SETDI command. See *W_SETDI — Set attributes common to all digital interfaces* on page *158* for more details.

- Multiple messages might be generated if some events go away and some get detected together. All detected events would be put together and all going-away events would be put together.
- For events W_DI_TX_OPEN, W_DI_FAR_AIS, and W_DI_FAR_LMFA, a WAN_EVT_RELEASED is generated on a polled basis; that is, within a second after the event has gone away.

Figure 7-1. Message flow for W_NOTIFDI

WAN_NOTIFTIM — Send a timestamped notification

This message type informs the upper level of events in relation to the timestamp. It is used only when the Multiplexed WAN driver is running in monitor mode. See the command-line parameter *W_MONITOR_MODE* on page *239* for information.

Reported events are as follows:

- WAN_TICK_EVENT

  This event is sent on all active data streams to indicate that the timestamp has crossed a 100 ms boundary. Frames sent after the tick event will have a timestamp greater than, but not equal to, that of the tick event.

- WAN_NOTIFDI

  This event is sent on the management stream to inform the upper level of events related to one of the digital interfaces.

- WAN_NOTIF_ATM

  This event is sent on the management stream to inform the upper level of events related to the ATM cell stream.

The following structure is associated with this M_PROTO message:

```
typedef union {
     struct wan_notifdi      di;
     struct wan_notif_atm    atm;
}    tim_event;
struct wan_notiftim   {
     uint8                   wan_type;
     uint8                   wan_spare[3];
     uint32                  wan_event;
     UINT64                  wan_timestamp;
     tim_event               wan_notif;
};
```

Parameters

`wan_type` Output. This is always WAN_NOTIFTIM.

`wan_event` Output. This indicates the event being reported, and will have one of the following values:

WAN_TICK_EVENT
: A tick event is generated when the timestamp has crossed a 100 ms boundary.

WAN_NOTIFDI
: Events related to one of the digital interfaces have occurred. Information about these events is in the structure wan_notifdi di field.

WAN_NOTIF_ATM
: Events related to the ATM cell stream status have occurred. Information about these events is in the structure wan_notif_atm atm field.

`wan_timestamp`
Output. This is the timestamp value in milliseconds.

`wan_notif` Output. This is the notification information. For definitions of the associated structures, see:

- *WAN_NOTIFDI* on page *127*
- *WAN_NOTIF_ATM* on page *185*.

This message is supported by the Multiplexed WAN driver only when using monitor mode. See *W_MONITOR_MODE* on page *239* for information.

This message is not currently supported by the Serial WAN driver.

Figure 7-2. Message flow for WAN_NOTIFTIM

STREAMS management commands for T1/E1

ioc_cmd value of iocblk structure in M_IOCTL	Structure in M_DATA after M_IOCTL	M_DATA with M_IOCACK?
W_DI_TEST_CFG	wan_ditestcfg_ioc (see *W_DI_TEST_CFG — Set test configuration for a physical port* on page *137*)	Yes
W_SET_PHY_PIPE	wan_setphypipe_ioc (see *W_SET_PHY_PIPE — Define and undefine time slots* on page *140*)	Yes
W_GET_PHY_PIPE	wan_getphypipe_ioc (see *W_GET_PHY_PIPE — Retrieve time-slot information* on page *144*)	Yes
W_SETCH_MAP	wan_setchmap_ioc (see *W_SETCH_MAP — Set up channel map table* on page *146*)	Yes
W_GETCH_MAP	wan_getchmap_ioc (see *W_GETCH_MAP — Get channel map table settings* on page *156*)	Yes
W_SETDI	wan_setdi_ioc (see *W_SETDI — Set attributes common to all digital interfaces* on page *158*)	No
W_GETDI	wan_getdi_ioc (see *W_GETDI — Get attributes common to all digital interfaces* on page *164*)	Yes
W_SETDI_PORT	wan_setdiprt_ioc (see *W_SETDI_PORT — Set attributes of a physical port* on page *165*)	No
W_GETDI_PORT	wan_getdiprt_ioc (see *W_GETDI_PORT — Get attributes of a physical port* on page *174*)	Yes
W_GETDI_STATS	wan_getdistats_ioc (see *W_GETDI_STATS — Get port statistics* on page *176*)	Yes
W_ZERODI_STATS	wan_zerodistats_ioc (see *W_ZERODI_STATS — Clear port statistics* on page *178*)	No
W_SETDI_LPBK	wan_setdilpbk_ioc (see *W_SETDI_LPBK — Put port in loopback* on page *179*)	No
W_SETDI_NOTIF	wan_setdinotif_ioc (see *W_SETDI_NOTIF — Set event filter for a physical port* on page *181*)	No
W_SET_TIMESTAMP	wan_time_ioc (see *W_SET_TIMESTAMP — Set timestamp* on page *183*)	Yes

W_DI_TEST_CFG — Set test configuration for a physical port

This command is useful in a test and development environment so that you can verify various programming paths. This command relies on the hardware chips to generate test conditions. Refer to *Siemens PEB2254* data sheets for detailed information on alarm simulation. PEB2254 registers that provide this support are as follows:

- Framer Mode Register 0 (SIM bit)
- Framer Receive Status Register 2 (ESC2-ESC0 bits)

For T1, the simulation is carried out in eight steps. It is the upper layer's responsibility to set up the appropriate simulation step. Once the test is complete, it should bring the hardware back to step zero and then turn simulation off.

For E1, the simulation is done in one step only.

Additionally, in ATM mode, this command can be used to generate ATM cells with HEC errors.

The following structure is associated with this command:

```
typedef union_test_data {
   uint32           w_esc;
   struct test_atm_data {
   uint8            parm0;
   uint8            parm1;
   uint16           parm2;
   } test;
}  test_data;
struct wan_ditestcfg_ioc {
   uint8            w_type;
   uint8            w_spare[3];
   uint32           w_port_id;
   uint32           w_action;
   test_data        w_test_data;
};

#define             w_esc w_test_data.w_esc
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_DI_TEST_CFG.

w_type
: Input. This is set to WAN_DI_TEST_CFG.

w_port_id
: Input. For ATM HEC testing, this specifies the pipe over which ATM cell traffic is taking place and on which this test will be carried out. Otherwise, this specifies the port number, a value between 1 and 4 (inclusive), on which the test is to be carried out.

w_action Input. A bit-wise OR field indicating parameters to set and tests that are to be run. The bits are as follows:

T1/E1 Alarm Simulation Testing

WAN_START_ALARM_SIM
To start alarm simulation and move to the next alarm-simulation step. Every time this command is issued, the simulation step is bumped to the next step.

WAN_STOP_ALARM_SIM
To end alarm simulation.

ATM HEC Testing

W_SET_NUM_BAD_HEC
When this bit is set, the w_parm0 field specifies the number of consecutive ATM cells with bad HECs (Header Error Checksums) that are to be generated. The default is zero.

W_SET_NUM_GOOD_HEC
When this bit is set, the w_parm1 field specifies the number of consecutive ATM cells with good HECs that are to be generated. The default is zero.

W_START_HEC_TEST
When this bit is set, the ATM framer enters error generation mode. When in this mode, the ATM framer transmits ATM cells according to the parameters set by W_SET_NUM_BAD_HEC and W_SET_NUM_GOOD_HEC, where cells with bad HECs are transmitted first (the number of such cells are defined by w_parm0) followed by cells with good HECs (the number of such cells are defined by w_parm1).

This pattern is repeated *n* times, where *n* is defined by the w_parm2 field. If *n* is set to zero, this pattern will be repeated indefinitely, until this command is issued with W_STOP_HEC_TEST.

W_STOP_HEC_TEST
When this bit is set, it ends the test (error-generation mode) and returns to normal operation, only if the test is in progress.

w_esc Output. For T1, Error Simulation Counter reflects the simulation step number the hardware is in after issuing the command.

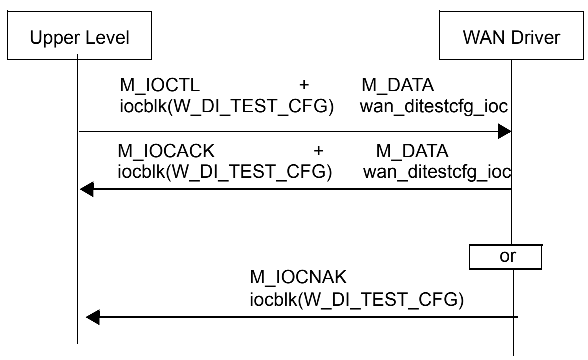
For E1, this value is not defined.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
EINVAL	Either there is an invalid option in w_action or the command size does not match.
ERANGE	The port number supplied is out of range for the current hardware.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.

- Currently this command is supported for T1 modes only. This command is under study for E1 modes.
- Bits W_SET_NUM_BAD_HEC, W_SET_NUM_GOOD_HEC, and W_START_HEC_TEST can be set at the same time, where parameters are set and then the test starts.
- When bit W_STOP_HEC_TEST is set, no other bits should be set.
- When the test ends, parameters (iteration count, cells with good and bad HECs) are reset to zero.

Figure 7-3. Message flow for W_DI_TEST_CFG



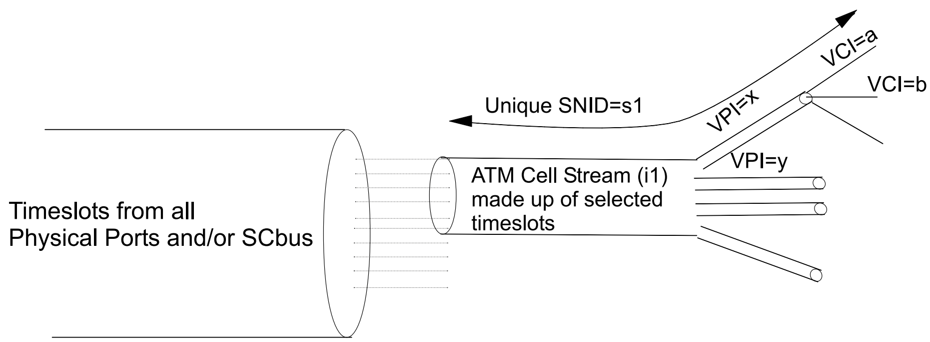
W_SET_PHY_PIPE — Define and undefine time slots

This management command is used for the following.

- Defining an ID for the physical stream
- Assigning time slots that make up the physical stream
- Configuring the physical stream for ATM, HDLC, or SS7
- Clear Channel Capability

When in ATM mode, this would be the first configuration command to be issued. Time slots can be obtained from the physical ports or SC bus. Data from multiple virtual channels can be carried over this physical stream by associating the identifier of the cell stream with the SNID and VCC using the W_SET_SNID command.

Relationship between timeslots, ATM cell streams, VPI/VCI and SNIDs



The following structure is associated with this command:

```
typedef struct {
   uint16     port_id;
   uint16     chan_id;
} time_slot;

typedef struct phy_pipe {
   uint32     w_phy_pipe_id;
   uint32     w_num_of_time_slots;
   time_slot  w_rx_ts[32];
   time_slot  w_tx_ts[32];
   uint32     w_disc_mask;
   uint32     w_options;

} phy_pipe;

struct wan_set_phy_pipe_ioc {
   uint8      w_type;
   uint8      w_spare[3];
   phy_pipe   w_phy_pipe;
};
```

Parameters

IOCTL_COMMAND
Input. The ioc_cmd field in struct iocblk should be W_SET_PHY_PIPE.

w_type Input. This should always be WAN_SET_PHY_PIPE.

w_phy_pipe
Input. The following fields are defined for the structure:

w_phy_pipe_id
Input/Output. This is a unique identifier associated with the combination of the time slots defined by this command.

The WAN driver returns a unique identifier when time slots are being defined for a physical pipe by the upper layer (that is, the w_num_of_time slots field is nonzero).

This field is input when the w_num_of_time slots field is set to zero by the upper layer, indicating that the upper layer wants to undefine or free up the time slots associated with the physical pipe.

w_num_of_time_slots
Input. This defines the total number of time slots that are to be combined. Therefore, fill in the many entries of w_rx_ts and w_tx_ts arrays starting from the first element of the arrays. The Maximum value is 32 (decimal). If this value is set to zero, it undefines or ungroups the time slots associated with this physical pipe.

w_rx_ts, w_tx_ts
Input. This identifies the time slots from which data is to be received and to which the data is to be transmitted. Data will be combined in the order the time slots are specified. That is, the time slot specified by array element 0 will be the first octet of data, array element 1 will be the second octet of data, and so on. See *Figure 7-6* on page *147* and *Figure 7-7* on page *149* for information on how the command defines the values for port and channel numbers.

w_disc_mask

Input. This is a bit-wise OR field, and the description of the bits is the same as the wan_event field of the WAN_NOTIFDI command. The default value is 0. This field applies only for ATM.

This field plays a role when all channels that make up the pipe are from the same physical port. In this case, and if one or more of these bits are set, then when the appropriate event is detected, the WAN driver will disable the ATM physical layer. This will result in a WAN_NOTIF_ATM with WAN_LOST_ATM_ CELL_ SYNCH. In addition, if the WAN_event_disc field (bit WAN_CELL_SYNC) of the W_SETTUNE command is set, WC_DISC will be generated on appropriate data streams associated with this physical pipe.

w_options

Input. This field consists of various options associated with this pipe. This is a bit-wise OR field. The default for this field is 0. The options are as follows:

W_BIT_INVERT

When enabled, this will perform 1's complement of the data before putting it on the line (transmit direction), and 1's complement of the data before processing it (receive direction).

W_PIPE_MODE, W_ATM_MODE

The ATM physical layer will be selected when either of these options are enabled.

W_HDLC_MODE

HDLC framing mode will be selected when this option is enabled.

W_SS7_MODE

SS7 mode will be selected when this option is enabled.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK. In case of an error, an M_IOCNAK message is sent back with an appropriate error code.
EINVAL	The message size does not match.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.
ERANGE	One or more parameters do not contain the proper value.
ENOSR	The specified configuration could not be set up because the underlying resources are not available.

ENODEV The specified w_phy_pipe_id does not exist.

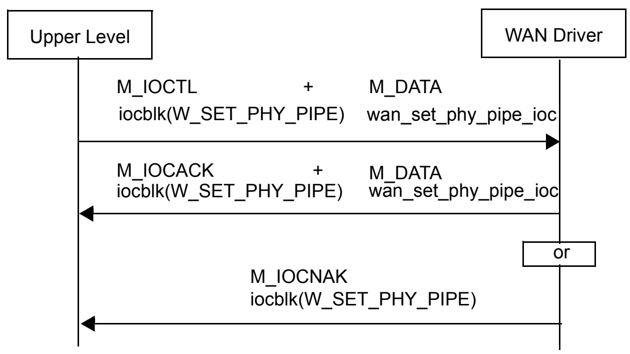
EBUSY One or more virtual connections were open when an attempt was made to ungroup the time slots associated with this physical pipe.

Other Errors

See the error codes listed under the EXDEV error for *W_SETCH_MAP — Set up channel map table* on page 146.

- The order of the specified time slots is important and should match with the other end.
- This command must be issued to obtain a pipe ID. Pipes are not created by default.
- Either W_HDLC_MODE, W_PIPE_MODE, or W_ATM_MODE must be set when using this command.

Figure 7-4. Message flow for W_SET_PHY_PIPE



W_GET_PHY_PIPE — Retrieve time-slot information

This management command is used to retrieve information associated with a physical pipe that was previously set by the W_SET_PHY_PIPE command.

The following structure is associated with this command:

```
struct wan_get_phy_pipe_ioc {
   uint8      w_type;
   uint8      w_spare[3];
   phy_pipe   w_phy_pipe;
};
```

Parameters

IOCTL_COMMAND
: The ioc_cmd field in struct iocblk should be W_GET_PHY_PIPE.

w_type
: Input. This should always be WAN_GET_PHY_PIPE.

w_phy_pipe
: Only the w_phy_pipe_id field is input. The remaining structure fields are output.

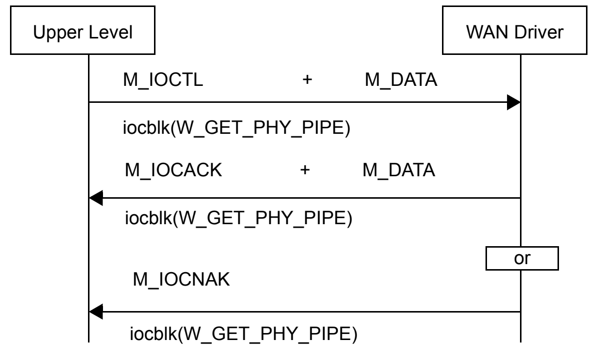
: If this parameter is set to -1 and x_rx_ts[0] and w_tx_ts[0] are set to a valid time slot of the pipe, then w_phy_pipe_id is returned with the valid pipe ID.

: See page *141* for a description of the fields for this structure.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK. In case of an error, an M_IOCNAK message is sent back with the appropriate error code.
EINVAL	The message size does not match.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.
ENODEV	The physical pipe defined by the w_phy_pipe_id field is not defined.

Figure 7-5. Message flow for W_GET_PHY_PIPE



W_SETCH_MAP — Set up channel map table

This command is used to configure how a particular channel is to be connected to another channel. This command also can be used for breaking the connection between channels. This is a fairly low-level command to implement various configurations. It exposes to the upper layer, the switching capability of the following:

- ARTIC960 4-Port T1/E1 Mezzanine Card (see *Figure 7-6. Port assignments for ARTIC960 4-Port T1/E1 Mezzanine Card* on page *147*)
- ARTIC 1000/2000 Series adapters (see *Figure 7-7. Port assignments for ARTIC 4-Port T1/E1/J1 DSP PMC and ARTIC 1000/2000 Series* on page *149*).

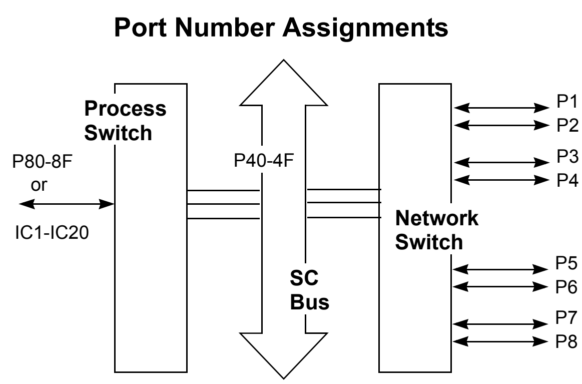
ARTIC960 4-Port T1/E1 Mezzanine Card (for ARTIC960)

The following explains *Figure 7-6. Port assignments for ARTIC960 4-Port T1/E1 Mezzanine Card* on page *147*.

- Network ports (0x01 to 0x08) carry a maximum of 32 full-duplex channels numbered 1 through 0x20.
- Processing ports (0x80 to 0x8F) carry a maximum of 16 full-duplex channels numbered 1 through 0x10, although the number of channels processed by the processing ports is dependent on the DSP code and/or the application.
- The SC-bus ports (0x40–0x4F) can carry a maximum of 32 or 64 half-duplex channels (1 to 0x20 or 1 to 0x40), depending on the SC-bus configuration.
- The P0 port is reserved and cannot be used in mapping.
- The port numbers P100 through P1FF are reserved for identifying the physical pipes or channel groups formed using the W-SET_PHY_PIPE command. These port numbers cannot be used in this command, but are used in W_SET_SNID to configure a VCC on the pipe for ATM operation.
- The port number PFF is reserved for identifying global events in W_SETDI_NOTIF and WAN_NOTIFDI. It also cannot be used in this command.

When the physical channels are mapped to the SC-bus time slots, the w_xmt field (see page *151*) refers to transmission to the SC bus, which is actually the receive on the physical port. For the internal and DSP channels, however, the sense of transmit and receive is the same as the fields suggest.

Figure 7-6. Port assignments for ARTIC960 4-Port T1/E1 Mezzanine Card



Legend

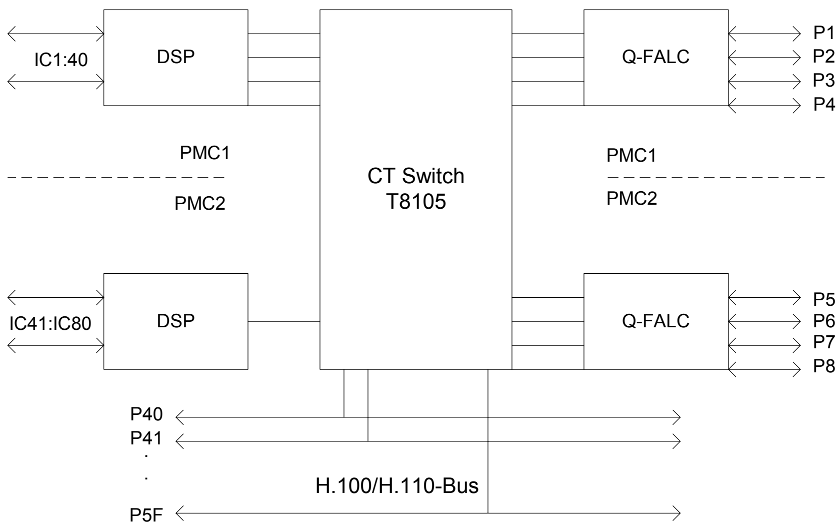
P0	Reserved.
PFF	Global port, for specifying and getting notifications on the global events.
P1...P8	Network/Physical Ports connected to the card. P1–P4 for the ARTIC960 4-Port T1/E1 Mezzanine Card.
P80...P8F	DSP/Processing Port where a channel can be processed. P80 and P81 for the ARTIC960 4-Port T1/E1 Mezzanine Card.
P100...P1FF	The ports identifying the physical pipes configured with W_SET_PHY_PIPE.
IC1...IC20	Internal channels 1–0x20.
P40...P4F	Represents the 16 wires of the SC bus.
SC bus	The 16-wire bus configured such that each wire carries either 32 or 64 channels (Half-Duplex).
Network Switch	The switch capable of switching any channel from Pi (1–8) to any channel on the SC bus, or to another Pi.
Processor Switch	The switch capable of switching any channel from Pi (80–8f) or internal channels (IC1–IC20) to any channel on the SC bus, or to another Pi or internal channel.

ARTIC 4-Port T1/E1/J1 DSP PMC (for ARTIC 1000/2000 Series)

The following explains *Figure 7-7. Port assignments for ARTIC 4-Port T1/E1/J1 DSP PMC and ARTIC 1000/2000 Series* on page *149*.

- The CT Switch (Computer Telephony Switch) of the ARTIC 1000 CompactPCI I/O Platform adapter allows any internal channel of the DSP to be routed to any channel within a network port (P1–P8) or any H.110-bus channel.
- Each of the network ports (0x01–0x8) can carry a maximum of 24 (for T1) or 30 (for E1) full-duplex channels numbered 1–0x1E for T1 or E1, respectively. However, each PMC will support a total of 64 channels with PMC #1 supporting channels
1–0x40 and PMC #2 supporting channels 0x41–0x80.
- The H.100/H.110 bus has 32 serial ports (0x40–0x5F) that can be programmed for three speeds in order to support 32, 64, or 128 half-duplex channels. The number of connections to the H.100/H.110 bus is limited to 512 half-duplex connections.
- The P0, PFF and P100–P1FF ports are treated as shown in *Figure 7-7* on page *149*.

Figure 7-7. Port assignments for ARTIC 4-Port T1/E1/J1 DSP PMC and ARTIC 1000/2000 Series



Legend

P0	Reserved.
PFF	Global port, for specifying and getting notifications on the global events
P1...P4	Network/Physical Ports connected to the first PMC
P5...P8	Network/Physical Ports connected to the second PMC
P100...P1FF	Ports identifying the physical pipes configured with W_SET_PHY_PIPE.
IC1...IC40	First PMC DSP channels 1–0x40
IC41...IC80	Second PMC DSP channels 0x41–0x80
P40...P5F	Represents 32 wires of the H.100/H.110-bus.
H.100/H.110 bus	The 32-wire bus configured so that each wire carries either 32, 64, or 128 channels (half-duplex).
CT Switch	The Computer Telephony Switch capable of routing channels from Pi (1–8) to: • any channel on the H.100/H.110 bus • any DSP internal channel • another Pi.

The following structure is associated with this command:

```
typedef union ch_type {
   struct          {
       uint16     port;
       uint16     chan;
   }              physical;
   uint32         internal;
} w_chan_t;

typedef union wan_chnl_map {

   struct          {
       uint32     w_use;
       w_chan_t   w_map;
       w_chan_t   w_rec;
       w_chan_t   w_xmt;
   } map;

   struct          {
       uint32     w_use;
       w_chan_t   w_map;
       uint32     w_mpe;
       uint32     w_rxe;
       uint32     w_txe;
   } resp;

} wan_chnl_map_t;

struct wan_setchmap_ioc {
   uint8              w_type;
   uint8              w_spare[3];
   wan_chnl_map_t     w_chnl_map [10];
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_SETCH_MAP.

w_type
: Input. This is always WAN_SETCHNL_MAP.

w_chnl_map
: Input/Output. The structure describing the channel mapping to be set. The following fields are defined for the structure:

  w_use — If nonzero, the w_chnl_map item from the array is used by the WAN driver. This is a bit-wise OR of the following values:

  - *W_USE* — Always use this option when the mapping entry in w_chnl_map is to be used.
  - *W_ERR* — In the response, this bit is turned on if the entry was in error.
  - *W_MAP_INTERNALS* — The w_map field specifies an internal channel number.

W_REC_INTERNAL
: The w_rec field specifies an internal channel number.

W_XMT_INTERNAL
: The w_xmt field specifies an internal channel number.

W_DIR_PROC_PORT
: When the WAN driver breaks a connection, the default direction is towards the network; for example, if the w_map field specifies a DSP channel, and w_xmt and w_rec are both set to zero, and if this option is not specified, the connection between the DSP channel and SC bus or physical channel is broken. Alternatively, if this option is specified, the connection between the DSP channel and the internal channel is broken. Likewise, this option can be used in W_GETCH_MAP to obtain the mapping information in the processing port direction.

w_map This is a full-duplex channel within a port (Network or DSP), or an internal channel is being mapped. Valid values for the port are shown in *Figure 7-7* on page *149*.

The internal channel numbers are specified as member *internal* of the union w_chan_t.

w_rec This is the port channel or internal channel from where w_map receives its data.

w_xmt This is the port channel or internal channel to where w_map transmits its data.

w_mpe The error code for why the w_map field was seen as bad.

w_rxe The error code for why w_map could not be connected to w_rec.

w_txe The error code for why w_map could not be connected to w_xmt.

Error codes

0 — The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. Unlike other commands (even in case of an error), an M_IOCACK message is sent upstream so that the response buffer is copied back to the user. The following error codes are for the command as a whole.

EINVAL — The command size does not match, or the internal channel number is incorrect.

EXDEV — The requested mapping between two channels cannot be performed. In this case, the response buffer has an error code for each wrong mapping requested. If the w_err bit in the response is set, the following errors may be set. Reasons are as follows:

- A map operation cannot specify channels on either side of the SC bus (for example, w_map is on the processing-switch side, while either w_xmt or w_rec is a physical channel), if:
  - w_rec and w_xmt channels are different, or
  - They are the same, but no dedicated wires have been specified on the command line. Therefore, the WAN driver cannot allocate SC-bus time slots to cross the SC bus.
- If w_map is a physical channel, w_rec and w_xmt cannot be different unless they are SC-bus channels.
- An SC-bus time slot is on one of the dedicated lines.

ERANGE — Either the port number or the channel number supplied is out of range for the current hardware.

EIO — The channel is in the wrong state. Possible reasons are:

- The channel being used is from a physical port that is in remote or payload loopback.
- The WAN driver breaks all current mappings of the w_map_channel field before attempting to connect the channels as given by the new mapping. If the w_xmt or w_rec channels still remain mapped after breaking current mappings, this error is given.
- If an internal channel is used in the mapping, and it does not yet have a DSP channel, one will be assigned before completing the mapping. While this channel is in use, it cannot be used in a new mapping command. When an internal channel is used to break the mapping, the DSP channel will be de-allocated.
- All channels used in W_SET_PHY_PIPE cannot be used in W_SETCH_MAP. They can be used *only* when W_SET_PHY_PIPE is used to undo the grouping.

EBUSY Reasons are as follows:

- A channel cannot be used here if it is in the data path to an internal channel, and the corresponding SNID has been processed by WAN_SNID *and* W_ENABLE. To change the mapping, issue a W_DISABLE command, change the mapping, and then enable the channel using W_ENABLE.
- An SC-bus time slot is being transmitted to by some other channel.

ENOSR There are not enough resources to perform all map operations together. Try splitting them into more than one command.

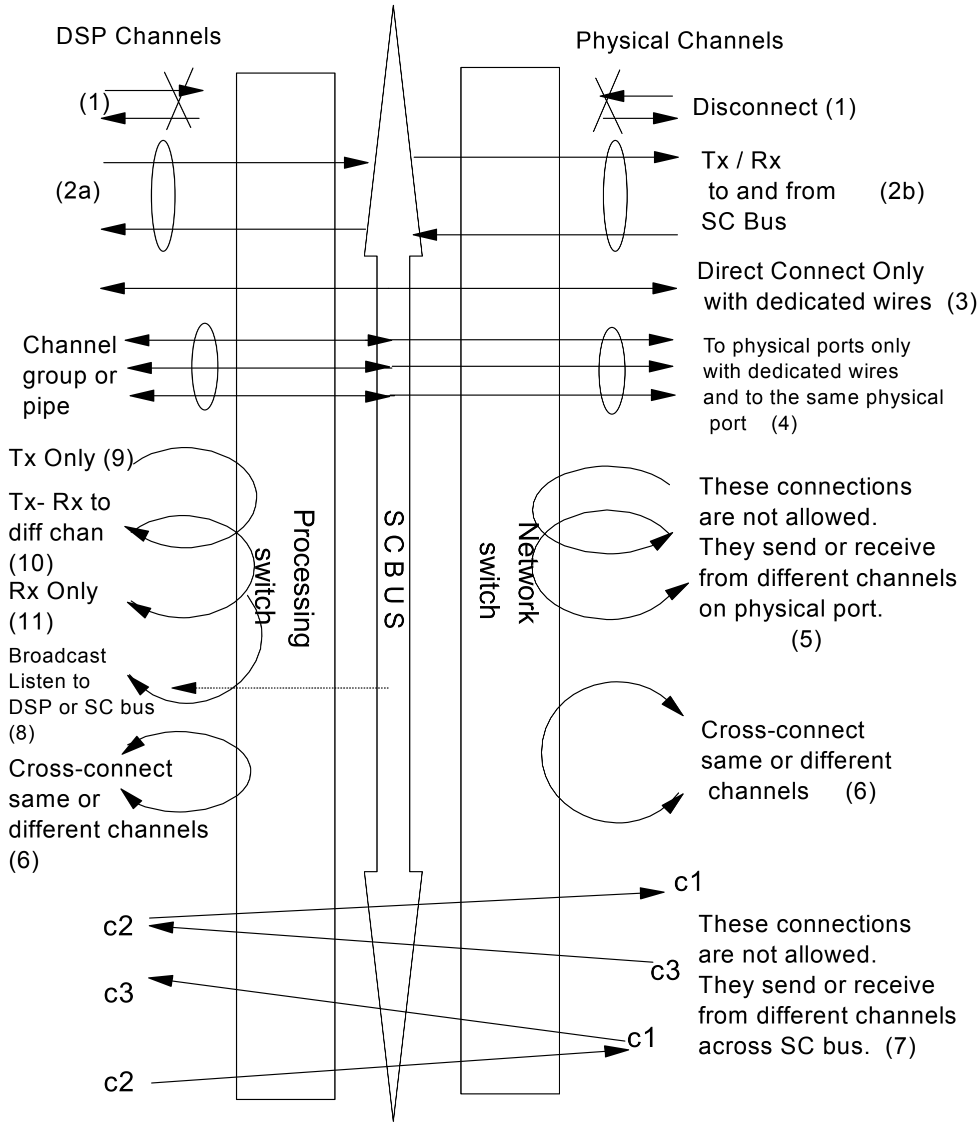
- If the port number is zero in the w_rec and/or w_xmt fields, the channel specified by the w_map field is not connected.
- If any of the w_chnl_map elements are in error, no partial mapping would be performed.
- When operating in SNID_DECODE=YES mode, this command can be used *only* to map physical channels that have not been associated with any logical channels through the WAN_SID command.
- The established link must be broken before mapping it again for any of the following conditions:
  - If a loopback is done through W_SETDI_LPBK
  - If a chaining is done through W_SETDI_PORT

The link can be broken through W_SETCH_MAP or W_SETDI_PORT (for port chaining) or W_SETDI_LPBK before a new mapping or loopback can be specified. This is true only for mapping between logical channels only or physical channels only.

While breaking the mapping, only one side need be specified as broken; the driver will implicitly declare the other end as disconnected, too. For example, if p1,c1 is mapped to p2,c4, to break that link you need only specify p1,c1 as mapped to nothing *or* p2,c4 as mapped to nothing. Both p1,c1 and p2,c4 do not need to be specified. The same applies for making the link; you need only specify that p1,c1 is mapped to p2,c4 to make the link. You do not have to specify also that p2,c4 is mapped to p1,c1.

The following figure illustrates which paths are possible, and the text describes how one can achieve this.

Figure 7-8. Possible paths



(1) The initial state; there are no default connections to and from the SC bus.

(2a) WAN transmits and receives data to and from the SC bus. This can be done by W_SET_SNID specifying the SC-bus port and channel numbers directly, or by first getting an internal channel using W_SET_SNID (with the w_port_id and w_chnl_id fields set to 0) and then performing a W_SETCH_MAP command to connect the internal channel to an SC-bus channel.

(2b) A channel within a physical port transmits and receives data to and from the SC bus. This is done by issuing a W_SETCH_MAP command.

(3) The WAN transmits and receives data to and from a channel within a physical port. For this case to work, dedicated wires must be defined (with command-line parameters w_scbus_xmit_wire and w_scbus_recv_wire). To achieve this connection, one can issue a W_SET_SNID command with appropriate physical-port and channel numbers, or first issue a W_SET_SNID command to obtain an internal channel ID, and then issue a W_SETCH_MAP command to map the internal channel to the physical channel.

(4) Multiple physical channels are combined to form a pipe; the WAN driver transmits and receives data to and from this pipe. This is achieved using the W_SET_PHY_PIPE command.

(5) A physical channel is transmitting and receiving data to and from different physical channels. This case is not possible.

(6) A physical channel transmits and receives data to and from another physical channel, or an internal channel transmits and receives data to and from another internal channel. This is done by issuing a W_SETCH_MAP command, or the entire physical port is chained to another physical port by the W_SETDI_PORT command.

(7) A physical channel transmits and receives data to and from different internal channels, or an internal channel transmits and receives data to and from different physical channels. This is not allowed.

(8) or (11)
An internal channel is simply receiving data from another internal channel or from an SC-bus channel, as if it were listening to a broadcast. This is achieved with either the W_SETCH_MAP or W_SET_SNID command. The transmit channel is set to 0 (zero) in the mapping command.

(9) An internal channel is simply transmitting data, as if it were to do a broadcast. This is done with the W_SETCH_MAP command.

(10) An internal channel transmits to and receives from different internal channels. This is acceptable, and can be achieved with the W_SETCH_MAP command.

W_GETCH_MAP — Get channel map table settings

This command is used to obtain the channel map settings. Using this command, you can also determine if the logical channel being worked with is mapped to any physical port and how. The mapping also reveals if the channel is looped or chained in some way.

The following structure is associated with this command:

```
struct wan_getchmap_ioc {
   uint8             w_type;
   uint8             w_spare[3];
   wan_chnl_map_t    w_chnl_rx  [5];
   wan_chnl_map_t    w_chnl_tx  [5];
};
```

Parameters

IOCTL_COMMAND
Input. The ioc_cmd field in struct iocblk should be W_GETCH_MAP.

w_type Input. This is always WAN_GETCHNL_MAP.

w_chnl_rx, *w_chnl_tx*
Input/Output. The structure describing the channel map that is currently set. See *Figure 7-6* on page 147 and *Figure 7-7* on page 149 for various fields.

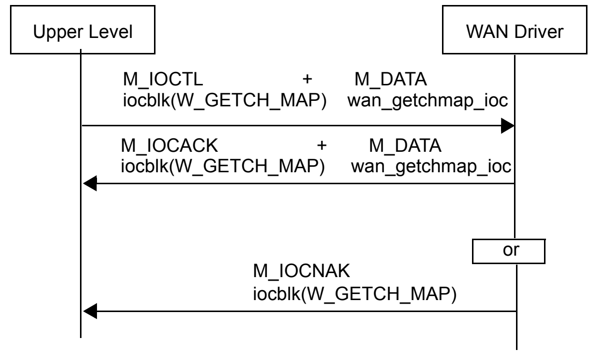
As input to this command, the user gives the channel number to query. Only the first item is used by the Multiplexed WAN driver. If the item is 0, the current stream's logical channel is assumed.

As a response to this command, the Multiplexed WAN driver fills the mapping information of the first connection in the first element.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with an M_IOCACK message in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
EINVAL	The command size does not match.
EIO	The w_chnl_rx(0) is 0, and the stream is a management stream.
ERANGE	Either the port number supplied is out of range for the current hardware, the channel number supplied is out of range for the current hardware, or the internal channel number is incorrect.

Figure 7-9. Message flow for W_GETCH_MAP



W_SETDI — Set attributes common to all digital interfaces

This command is used to configure the attributes that are common to all T1/E1 ports of the PMC. The configurable attributes are:

- Clock source common to all ports (internal or from one of the ports)
- First, second, and third backup clock sources
- Defines which wires can be dedicated to the ARTIC960 4-Port T1/E1 Mezzanine Card for transfer of data between physical channels and internal channels (channels processed by the ARTIC960 4-Port T1/E1 Mezzanine Card). The WAN driver is free to control the direction of these wires, as they will be defined as inputs for other adapters that are on the SC bus. The WAN driver needs two wires: one to receive data from the network switch, and another to transmit data to the network switch. This provides a way to avoid conflicts when multiple adapters are on the SC bus.

Dedicated wires are not required for the ARTIC 4-Port T1/E1/J1 DSP PMC.

The following structure is associated with this command:

```
typedef struct wan_di_info {
   uint32           w_master_clk ;
   uint32           w_bckup_clk_1;
   uint32           w_bckup_clk_2;
   uint32           w_bckup_clk_3;
   uint32           w_current_clk;
   uint32           w_net_switch_mode ;
} wan_di_info_t;

struct wan_setdi_ioc {
   uint8            w_type;
   uint8            w_spare[3];
   wan_di_info_t    w_di;
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_SETDI.

w_type
: Input. This is always WAN_SETDI.

w_di Input. The structure describing the parameters to be set, and general information for the digital interface ports. The following fields are defined for the structure:

w_master_clk
Indicates which port will provide the recovered clock from the received data that will drive the internal system highway. For the ARTIC 4-Port T1/E1/J1 DSP PMC, there can be up to eight ports if there are two PMCs installed.

W_NO_CHANGE	No change from the previous setting.
W_CLK_PORT_1	Clock from Port 1 (Default)
W_CLK_PORT_2	Clock from Port 2
W_CLK_PORT_3	Clock from Port 3
W_CLK_PORT_4	Clock from Port 4
W_CLK_PORT_5	Clock from Port 5
W_CLK_PORT_6	Clock from Port 6
W_CLK_PORT_7	Clock from Port 7
W_CLK_PORT_8	Clock from Port 8

w_bckup_clk_1
The first backup clock source. Same values as w_master_clk. The default is W_CLK_PORT_2.

w_bckup_clk_2
The second backup clock source. Same values as w_master_clk. The default is W_CLK_PORT_3.

w_bckup_clk_3
The third backup clock source. Same values as w_master_clk. The default is W_CLK_PORT_4.

w_current_clk
Not used by this command; it must be set to 0.

`w_net_switch_mode`

Specifies the operational mode of the network switch. It could be operating in one of the following modes:

`W_NO_CHANGE`

No change from the previous setting.

`SCBUS_MASTER`

The network switch is the master of the SC bus, and it drives the bus with appropriate clock control signals. In this mode, the network switch derives the clock from the source defined in the w_master_clk field. In the event of a loss of clock, the source clock is to be derived from sources defined in w_bckup_clk_1, w_bckup_clk_2, and w_bckup_clk_3, if the port number is different from the failed port number. This is the default.

In this mode, the network switch does not relinquish control of the bus, even if all ports specified in w_master_clk and w_bckup_clk lose their input signal. The driver assumes that no other Armed Master is on the bus that can supply clock signals to the SC bus. In the case where all ports lose signals, the SC bus is driven by a local oscillator. The only way to change this behavior is with W_SETDI and a different w_net_switch_mode.

`SCBUS_ARMED_MASTER`

The network switch is the armed master of the SC bus, and it will drive the SC bus when an SC-bus clock failure is detected. The clock source used is defined by the w_master_clk field.

When a clock failure is detected on the bus, the new mode of the switch will be SCBUS_MASTER. In this mode, the network switch will always provide a clock (either from one of the ports specified in w_master_clk or w_bckup_clk, or from the local oscillator). Also, it will remain in this mode until changed by another W_SETDI command.

`SCBUS_BACKED_MASTER`

In this mode, the network switch is the master on the SC bus as long as *any* of the ports specified in w_master_clk and w_bckup_clk have a signal. When all lose their signals, the WAN driver assumes that an Armed Master is configured on the SC bus and will relinquish control.

SCBUS_SLAVE
The network switch is put in slave mode, where it never drives the clock control signals of the SC bus.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
EINVAL	Either the clock source value for any of the fields is not defined, the command size does not match, or SC-bus related parameters are incorrect.
ERANGE	The values programmed in the w_scbus_ded_wires field are out of range.
ENXIO	A severe hardware error occurred in hardware. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.

Notes:

- On power-up, and in the absence of this command, Port 1 provides clocking for all other ports, and it derives this clock from its received data.
- When the port providing the master clock fails, the communications chip for that port will fall back to an internal clock source until the WAN driver can switch the master clock source from a different port.
- When a port clock fails or recovers, the WAN driver searches for a backup master clock source from the w_master_clk field through the w_bckup_clk_3 field, in order.
- When all ports are down, the internal highway is clocked from an internal crystal clock source.
- The command-line parameters (at driver load time), w_scbus_xmit_wire and w_scbus_recv_wire, indicate which wires are dedicated for the use of the Multiplexed WAN driver (note that dedicated wires are not required for the ARTIC 4-Port T1/E1/J1 DSP PMC).
  - w_scbus_xmit_wire refers to the wire that carries data from the processor in the network direction.
  - w_scbus_recv_wire refers to the wire that carries data from the network in the processor direction.

  For possible values, see *Command-line parameters* on page 235.

  The lowest value represents SD0 of the SC-bus and so forth. If both of these are set to 0 (zero), there are no dedicated wires in this SC-bus configuration, and you must use the W_SETCH_MAP command to set up the processing paths. Also, if you program one element to be a nonzero value, the other element also must be programmed to a different nonzero value, unless the SC bus is programmed for 4.096 Mbps rate.

- The command-line parameter (at driver load time), w_scbus_framing_mode, selects the framing mode of the SC bus. Possible values are:

  `W_SCBUS_AT_2048`
  : The SC bus is configured for 2.048 Mbps, 256 bits/frame, and 32 time slots/frame. This is the default.

  `W_SCBUS_AT_4096`
  : The SC bus is configured for 4.096 Mbps, 512 bits/frame, and 64 time slots/frame.

  `W_SCBUS_AT_8192`
  : The SC bus is configured for 8.192 Mbps, 1024 bits/frame, and 128 time slots/frame. This is not available on the ARTIC960 4-Port T1/E1 Mezzanine Card.

- In all network switch modes, the WAN driver will continue notification about *all* clock failures on the physical ports. Using this information, the user application should be able to determine if the system would be asynchronous if a particular PMC were requested to be a master on the SC bus.
- In SCBUS_SLAVE and SCBUS_ARMED_MASTER mode, the physical port timing towards the SC bus is always driven by the SC bus, regardless of the selection in w_master_clk and w_bckup_clk.
- This API selects only the timing for the SC bus (or the internal system highway) and the physical-ports interface timing to the system highway. The physical-port transmit timing is dependent on whether a particular port is in master mode or not (see *w_clk_mode* on page *169*). Normally, the transmit timing is derived from the receive bit stream (slave mode), but when a particular port is in master mode, or it loses its receive signals, it uses the local oscillator for transmit timing (which could be asynchronous to the SC-bus timing).
- The local oscillator also could drive the system highway if the current port selected for the system highway loses its signals, and it could do so indefinitely, if all choices were exhausted and the network switch remained the master of the SC bus.
- Care should be taken so that on a physical port connection, only one port is in master mode and the other is in slave mode.

The following figure shows the relationship between the recovered clocks from the communication chips (FALCs) and the network switch.

Figure 7-10. Relationship between recovered clocks

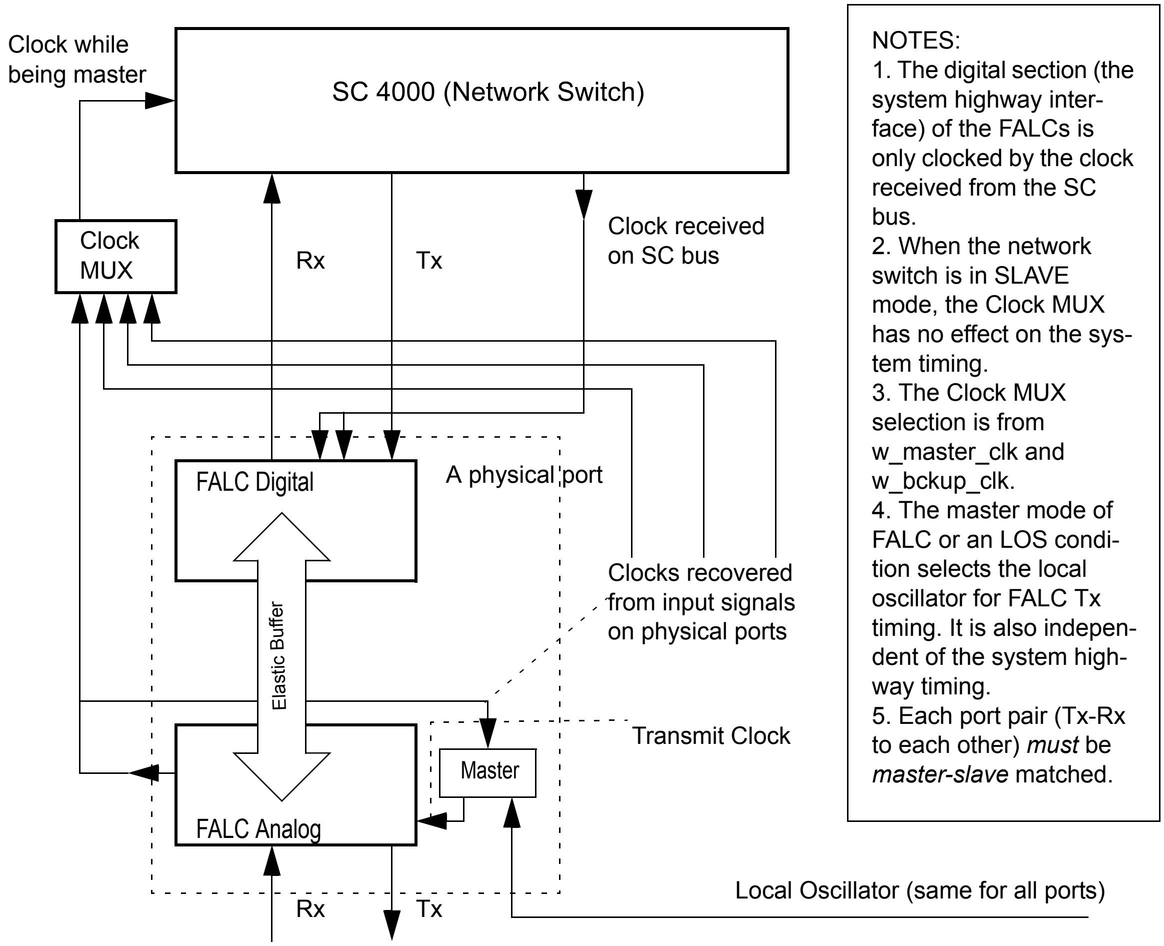
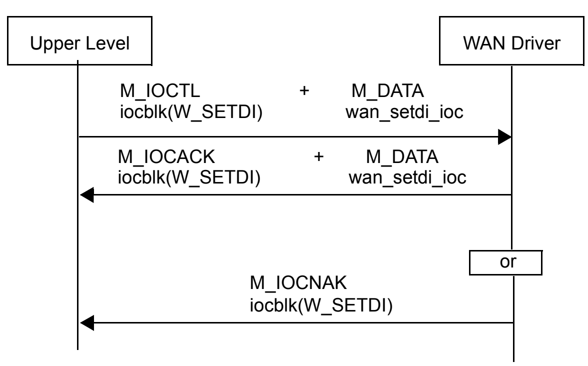


Figure 7-11. Message flow for W_SETDI



W_GETDI — Get attributes common to all digital interfaces

This command is used to obtain attributes that are common to all T1/E1 ports on the PMC. See *W_SETDI — Set attributes common to all digital interfaces* on page *158* for a list of the attributes.

The following structure is associated with this command:

```
struct wan_getdi_ioc {
   uint8           w_type;
   uint8           w_spare[3];
   wan_di_info_t   w_di;
};
```

Parameters

IOCTL_COMMAND
Input. The ioc_cmd field in struct iocblk should be W_GETDI.

w_type Input. This is always WAN_GETDI.

w_di Output. The structure describing the parameters currently set. See *W_SETDI — Set attributes common to all digital interfaces* on page *158* for more information on the structure. The w_current_clk field indicates the current master clock source.

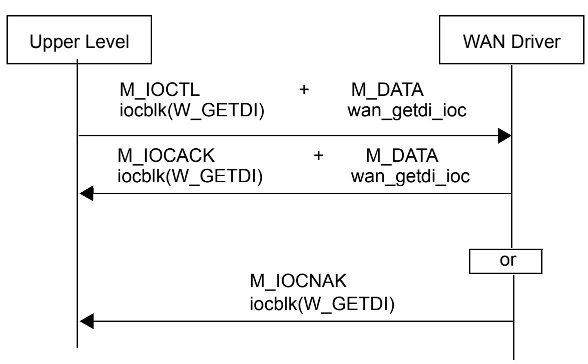
Error codes

0 The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

EINVAL The command size does not match.

ENXIO A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.

Figure 7-12. Message flow for W_GETDI



W_SETDI_PORT — Set attributes of a physical port

This command is used to configure the attributes of a particular port. The configurable attributes are:

- Line coding
- Frame format
- Control of blue alarm/AIS
- Chaining (all channels connected to equivalent channels on another port)
- Bit inversion for the entire port
- Alarm propagation

The following structure is associated with this command:

```
typedef struct wan_diprt_info {
   uint32           w_frame_format;
   uint32           w_crc_active;
   uint32           w_T1_LOF_err_bits;
   uint32           w_loop_back_conf;
   uint32           w_line_status_gen;
   uint32           w_line_status;
   uint32           w_line_coding;
   uint32           w_signal_mode;
   uint32           w_fdl_type;
   uint32           w_chain_flag;
   uint32           w_chain_port;
   uint32           w_bit_inv ;
   uint32           w_clk_mode ;
   uint32           w_port_fail_mask ;
} wan_diprt_info_t;

struct wan_setdiprt_ioc {
   uint8            w_type;
   uint8            w_spare[3];
   uint32           w_port_id;
   wan_diprt_info_t w_diprt;
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_SETDI_PORT.

w_type
: Input. This is always WAN_SETDI_PORT.

w_port_id
: Input. The port number; a value between 1 and 8 (inclusive).

w_diprt — Input. The structure describing the parameters to be set. The following fields are defined for the structure:

w_frame_format
: Indicates the variety of the digital interface line. Valid values are:

- *W_NO_CHANGE* — No change from previous setting.
- *W_DI_ESF* — Only T1. Extended Super Frame (ESF). This is the default.
- *W_DI_SF* — Only T1. Super Frame (SF) or AT&T D4 format.
- *W_DI_E1* — Only E1. Double Frame Default.
- *W_DI_E1_MF* — Only E1. Multiframe with CRC-4.
- *W_DI_E1_MF_G706_B* — Only E1. Multiframe with CRC-4 and allows for interworking between CRC-4 and non CRC-4 equipment.

w_crc_active
: Indicates if CRC-6 values for T1 (ESF framing format) are generated and accepted. Valid values are:

- *W_NO_CHANGE* — No change from previous setting.
- *W_CRC_OFF* — CRC inactive. This is the default.
- *W_CRC_ON* — CRC active.

w_T1_LOF_err_bits
: Applicable to T1 mode, this field determines how loss of frame alignment is declared.

- *W_NO_CHANGE* — No change from previous setting.
- *W_DI_T1_LOF_2_OUT_OF_4* — Two errors within four framing bits lead to loss of frame alignment.
- *W_DI_T1_LOF_2_OUT_OF_5* — Two errors within five framing bits lead to loss of frame alignment. This is the default.
- *W_DI_T1_LOF_2_OUT_OF_6* — Two errors within six framing bits lead to loss of frame alignment.

w_loop_back_conf
This field is not used by this command. It is used by the W_GETDI_PORT command.

w_line_status_gen
Controls automatic generation of RAI and AIS. This is a bit-wise OR of the following values:

W_DI_AUTO_RAI
The Remote Alarm bit is automatically set in the outgoing data stream if the receiver is in asynchronous state. The asynchronous state is reached when the receiver loses frame alignment. This is the default.

W_DI_NO_AUTO_RAI
The Remote Alarm bit is not automatically set in the outgoing data stream if the receiver is in the asynchronous state.

W_DI_TX_AIS
Send AIS toward the remote end.

W_DI_NO_TX_AIS
Stop sending AIS toward the remote end. This is the default.

W_DI_TX_RAI
Send RAI toward the remote end.

W_DI_NO_TX_RAI
Stop sending RAI toward the remote end. This is the default.

w_line_status
This field is not used by this command. It is used by the W_GETDI_PORT command.

w_line_coding
Describes the variety of Zero Code Suppression used on the link. Valid values are:

W_NO_CHANGE
No change from previous setting.

W_DI_B8ZS
Only T1. This is the default for T1.

W_DI_HDB3
Only E1. This is the default for E1.

W_DI_AMI For T1 or E1.

w_signal_mode

The signaling mode of the interface. Valid values are:

W_NO_CHANGE

No change from previous setting.

E1 applications—short haul:

W_SHORT_HAUL

E1 applications—long haul:

W_LONG_HAUL

T1 applications—short haul:

W_0_40M	0–40 meters
W_40_81M	40–81 meters
W_81_122M	81–122 meters
W_122_162M	122–162 meters
W_162_200M	162–200 meters
W_0_133ft	0–133 feet
W_133_266ft	133–266 feet
W_266_399ft	266–399 feet
W_399_533ft	399–533 feet
W_533_655ft	533–655 feet

T1 applications—long haul:

W_0dB	0 dB loss
W_7_5dB	–7.5 dB loss
W_15dB	–15 dB loss
W_22_5dB	–22.5 dB loss

w_fdl_type

Describes the use of the Facility Data Link (FDL) on the interface. Reserved. Value must be 0=W_NO_CHANGE.

w_chain_flag

Valid values are:

W_NO_CHANGE
: No change from previous setting.

W_CHAIN
: Chain to the port specified by the `w_chain_port` field. Alarms are not propagated.

W_NO_CHAIN
: Do not chain. This is the default.

W_CHAIN_AND_PROPAGATE_TO_CHAIN_PORT
: Chain and propagate alarms to the port specified by the `w_chain_port` field. See the notes on page 172 for more information.

w_chain_port

The port number (1 to 4) on which this port is chained.

w_bit_inv

Valid values are:

W_NO_CHANGE
: No change from previous setting.

W_INVERT
: Apply bit inversion to incoming and outgoing bit streams.

W_NO_INVERT
: Normal mode; no inversion. This is the default.

w_clk_mode

This field selects only the transmit clock source. The receive clock is always extracted from the received signal.

W_NO_CHANGE
: No change from the previous setting.

W_MASTER_CLK
: Port uses the internal clock generated by the PMC for transmission of data. The communications chip is put in *master* mode.

W_SLAVE_CLK
: The communications chip is put in *slave* mode. The clock for transmit data is the recovered clock from one of the ports that is acting as the master clock source for the internal highway. This is the default.

W_USE_RECOVERED_CLK
: The communications chip is put in slave mode. The clock for transmit data is the recovered clock from its own receive port.

w_port_fail_mask

Defines which events will result in a port disconnection, causing the port-specific green LED to turn off. This bit field takes bit combinations as defined in the wan_event field of the WAN_NOTIFDI message (see *WAN_NOTIFDI — Inform upper level of T1/E1 events* on page 129). The port-specific green LED will default to off if any of the following events are detected:

- W_DI_FAR_RAI
- W_DI_FAR_AIS
- W_DI_LOS
- W_DI_FAR_LFA
- W_DI_FAR_LMFA

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
ERANGE	The port number supplied is out of range for the current hardware.
EINVAL	Either the option value for any of the fields is not defined, or the command size does not match.
EXDEV	The current operational mode (T1/E1) of the port does not match the request. Check the cable ID and current operational mode for the port using the W_GETHWTYPE command.
EIO	Either a loopback is active, or the companion port is in the wrong state.
EBUSY	At least one channel of this port is being used.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.

How to propagate alarms from port 1 to port 2 — example

Assume the following:

- Port 1 is connected to the E1 network.
- Port 2 is connected to the back end.
- Port 1 and Port 2 are chained together.

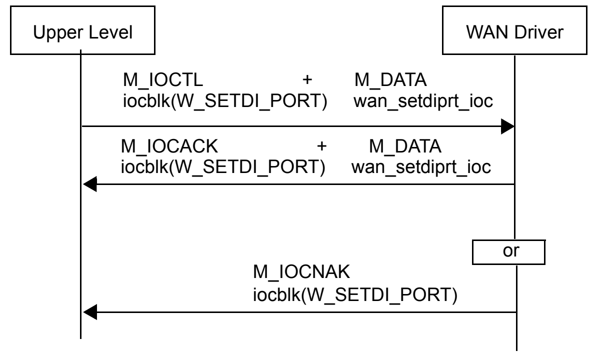
In order to chain the ports and have alarms from the E1 network propagate to the back end, do the following:

1. Issue a S_SETDI_PORT command for Port 1 with the following parameter settings:
   - Set the w_chain_flag field to W_CHAIN_AND_PROPAGATE_TO_CHAIN_PORT.
   - Set the w_chain_port field to 2.
2. Issue a W_SETDI_PORT command for Port 2 with the following parameter settings:
   - Set the w_chain_flag field to W_CHAIN.
   - Set the w_chain_port field to 1.

- On power-up, and in the absence of a SETDI_PORT command, the port would be programmed using the default values. A GETDI_PORT would return those defaults.
- At driver load time, the operational mode (T1 or E1) is determined as follows:
  - If a cable is attached and the cable ID is available, the type of cable determines the operational mode.
  - If the command-line parameter, W_INTERFACE_TYPE, was found, it determinesthe operational mode of the card.
  - If the command-line parameter also was not present, the operational mode defaults to E1.
- Once the operational mode is determined, it cannot be changed by plugging in a different type of cable. When a cable is disconnected, the Multiplexed WAN driver continues to look for a cable that is the same as the current operational mode. To change the current operational mode, the card must be reset.
- When the cable is disconnected, the Multiplexed WAN driver switches to balanced mode (high impedance). This is done to protect the circuitry when the old cable was a low-impedance cable and the new cable is of high impedance. When a new cable is plugged in, the Multiplexed WAN driver programs the impedance accordingly (balanced or unbalanced).
- In T1 Super Frame (12-frame multiframe) mode, there are two ways to indicate RAI (Remote Alarm Indication):
  - Set data bit 2 of all channels to 0 (zero).
  - Set the last bit of the multiframe alignment signal (bit 1 of frame 12) to 1 instead of 0. The WAN driver programs the hardware for this method to signal RAI.
- In T1 mode, the hardware is programmed in clear-channel mode; that is, the contents of the channel data are not overwritten by bit robbing and Zero Code Suppression information.
- The w_chain_flag *only* enables channel loopback to the same channel number on the port specified, until W_SETCH_MAP is issued to suggest a different connection.
- If the w_chain_flag is W_NO_CHAIN, *all* unused channels are lost on the card.
- If the w_chain_flag is W_CHAIN_AND_PROPAGATE_TO_CHAIN_PORT:
  - The value is valid only for E1 operation. If this value is specified for a port operating in T1 mode, an error code of EXDEV will be returned.
  - w_port_id must be a value between 1 and 4 (inclusive).

See the notes on page *153* for *W_SETCH_MAP — Set up channel map table* for more information.

Figure 7-13. Message flow for W_SETDI_PORT



W_GETDI_PORT — Get attributes of a physical port

This command is used to obtain the attributes of a particular port. In addition to the attributes listed in *W_SETDI_PORT — Set attributes of a physical port* on page *165*, this command also allows the following to be obtained:

- The presence of alarms from the far end (yellow alarm/RAI, blue alarm/AIS, red alarm/LOS)
- The generation of alarms at the near end
- The type of loopback (if any)

The following structure is associated with this command:

```
struct wan_getdiprt_ioc {
   uint8              w_type;
   uint8              w_spare[3];
   uint32             w_port_id;
   wan_diprt_info_t   w_diprt;
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_GETDI_PORT.

w_type
: Input. This is always WAN_GETDI_PORT.

w_port_id
: Input. The port number, a value between 1 and 4 (inclusive).

w_diprt
: Output. This structure is defined in *W_SETDI_PORT — Set attributes of a physical port* on page *165*. The fields that are unique to this command are described as follows:

w_loop_back_conf
: The loopback configuration of the interface. Valid values are:

W_DI_LOOP_NONE
: Not in loopback mode.

W_DI_LOOP_PAYLOAD
: Received signal is looped through the device (after the signal has passed through framing function).

W_DI_LOOP_REMOTE
: Entire signal is looped back out.

W_DI_LOOP_LOCAL
: Transmitted signal is looped back in.

w_line_status

The line status of the interface regardless of the notification mask. This is a bit-wise OR of the values listed in the wan_event field of the WAN_NOTIFDI service message. See *WAN_NOTIFDI — Inform upper level of T1/E1 events* on page *129* for more details. The following bits are defined in addition to the ones mentioned previously.

W_DI_NO_ALARM

No alarm present.

W_DI_NEAR_AIS

Near end sending AIS.

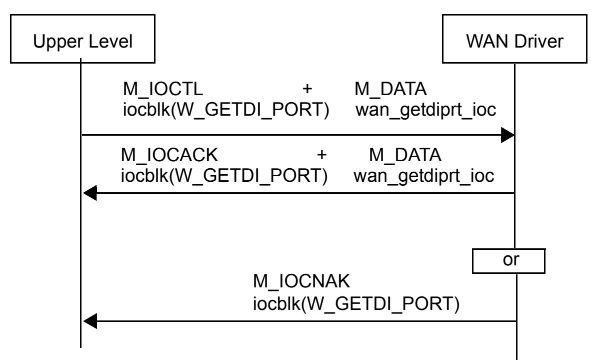
W_DI_NEAR_RAI

Near end sending RAI.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
EINVAL	The command size does not match.
ERANGE	The port number supplied is out of range for the current hardware.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.

Figure 7-14. Message flow for W_GETDI_PORT



W_GETDI_STATS — Get port statistics

This command is used to obtain the statistics for a particular port. The available statistics are:

- Number of Errored Seconds (ES)
- Number of framing errors
- Number of CRC errors (only for T1-ESF or E1-MF)
- Number of code violations
- Number of E-bit errors (only for E1-MF)

The following structure is associated with this command:

```
typedef struct wan_distats {
   uint32           w_err_secs;
   uint32           w_crc_err;
   uint32           w_framing_err;
   uint32           w_e_bit_err;
   uint32           w_code_violats;
} wan_distats_t;

struct wan_getdistats_ioc {
   uint8            w_type;
   uint8            w_spare[3];
   uint32           w_port_id;
   wan_distats_t    w_distats;
};
```

Parameters

IOCTL_COMMAND
Input. The ioc_cmd field in struct iocblk should be W_GETDI_STATS.

w_type Input. This is always WAN_GETDI_STATS.

w_port_id
Input. The port number, a value between 1 and 4 (inclusive).

w_distats
The statistics for that Digital Interface port. The following fields are defined for the structure:

w_err_secs
The number of Errored Seconds (ES) encountered by the interface.

w_crc_err
Only for T1-ESF and E1-MF. The number of CRC errors encountered by the interface.

w_framing_err
The number of framing errors encountered by the interface.

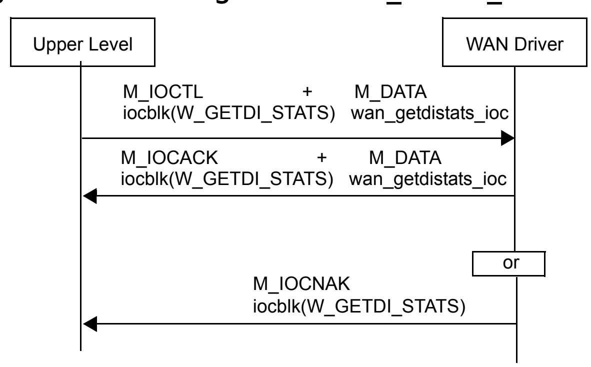
w_e_bit_err
: Only for E1-MF. The number of E-bit errors encountered by the interface.

w_code_violats
: The number of code violations encountered by the interface.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
EINVAL	The command size does not match.
ERANGE	The port number supplied is out of range for the current hardware.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.

Figure 7-15. Message flow for W_GETDI_STATS



W_ZERODI_STATS — Clear port statistics

This command is used to clear the statistics for a particular port. The w_distats field is filled with the current statistics just prior to clearing them so that the upper layer can obtain the statistics and then clear them in one operation.

The following structure is associated with this command:

```
struct wan_distats_ioc {
   uint8          w_type;
   uint8          w_spare[3];
   uint32         w_port_id;
   wan_distats_t  w_distats;
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_ZERODI_STATS.

w_type
: Input. This is always WAN_ZERODI_STATS.

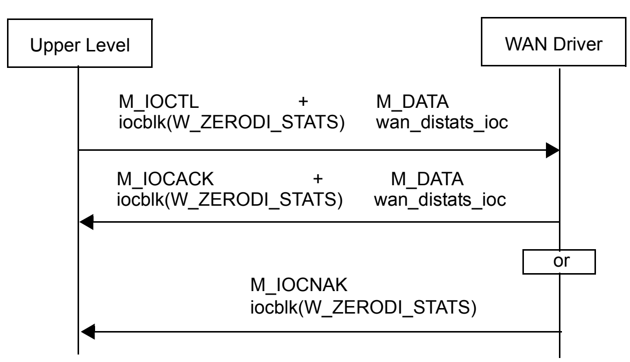
w_port_id
: Input. The port number, a value between 1 and 4 (inclusive).

w_distats
: Output. This field holds the statistic values before they were cleared. See page *176* for a description of this field and its elements.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
EINVAL	The command size does not match.
ERANGE	The port number supplied is out of range for the current hardware.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.

Figure 7-16. Message flow for W_ZERODI_STATS



W_SETDI_LPBK — Put port in loopback

This command is used to control the loopback on a port. Possible loopback modes are:

- Payload—Received data from the port is transmitted back with framing generated locally.
- Remote—Received data from the port is transmitted back (including framing).
- Local—Data to be transmitted is received back on the port.

The following structure is associated with this command:

```
struct wan_setdilpbk_ioc {

      uint8     w_type;
      uint8     w_spare[3];
      uint32    w_port_id;
      uint32    w_loopback_mode;
   };
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_SETDI_LPBK.

w_type
: This is set to WAN_SETDI_LPBK.

w_port_id
: The port number. A value between 1 and 8 (inclusive).

w_loopback_mode
: The loopback configuration of the port. Valid values are:

  W_DI_LOOP_NONE
  : Not in loopback mode. This is the default.

  W_DI_LOOP_PAYLOAD
  : Received signal is looped through the device (after the signal has passed through the framing function).

  W_DI_LOOP_REMOTE
  : Entire received signal is looped back out.

  W_DI_LOOP_LOCAL
  : Transmitted signal is looped back in.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
EINVAL	Either an invalid option is in w_loopback_mode, or the command size does not match.
ERANGE	The port number supplied is out of range for the current hardware.

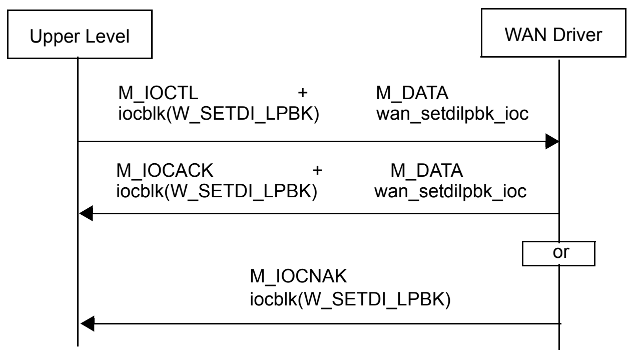
EBUSY — A stream is actively using a channel from this port. The stream *must* be closed before a loopback can be initiated.

EIO — The port is in the wrong state.

ENXIO — A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.

The port is programmed to the specified loopback configuration immediately.

Figure 7-17. Message flow for W_SETDI_LPBK



W_SETDI_NOTIF — Set event filter for a physical port

This command is used to control (enable or disable) the notification of events and alarms from the digital interfaces. The reporting of the following events is controlled by this message:

- Blue alarm/AIS
- Yellow alarm/RAI
- Red alarm/LOS
- Loss of clock
- Transmit line short
- Transmit line open

The following structure is associated with this command:

```
struct wan_setdinotif_ioc  {
   uint8            w_type;
   uint8            w_spare[3];
   uint32           w_port_id;
   uint32           w_action;
   uint32           w_event;
};
```

Parameters

IOCTL_COMMAND
Input. The ioc_cmd field in struct iocblk should be W_SETDI_NOTIF.

w_type Input. This is set to WAN_SETDI_NOTIF.

w_port_id
Input. The port number, a value between 1 and 4 (inclusive), where the event is to be detected.

w_action Input. The type of notification control to perform. The allowed values are:

WAN_EVT_ENABLE
To enable the event.

WAN_EVT_DISABLE
To disable the event.

w_event Input. The event/alarm being controlled for that port. These are bit-wise ORed and their values are the same as wan_event, defined on page 130.

Error codes

0 The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

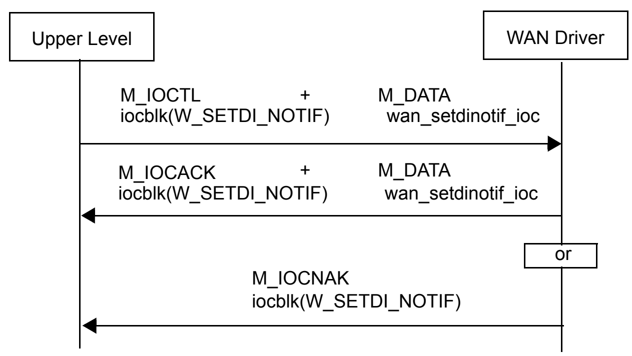
EINVAL Either there is an invalid option in w_action, an invalid condition in w_event, or the command size does not match.

ERANGE The port number supplied is out of range for the current hardware.

ENXIO A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.

When events are enabled or disabled, the previous events mask is overwritten. That is, the upper layer will need to track the events that it needs reported and perform logical OR or AND operations to get the proper mask.

Figure 7-18. Message flow for W_SETDI_NOTIF



W_SET_TIMESTAMP — Set timestamp

This command sets the current value of the timestamp, which is used when the Multiplex WAN driver is running in monitor mode. See *W_MONITOR_MODE* on page *239* for information.

The following structure is associated with this command.

```
struct wan_time_ioc {
   uint8            w_type;
   uint8            w_spare[7];
   UINT64           w_current_time;
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_SET_TIMESTAMP.

w_type
: Input. This is always WAN_SET_TIMESTAMP.

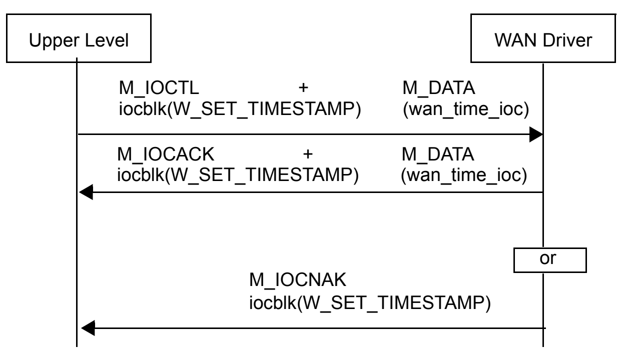
w_current_time
: Input. This is the current time in milliseconds. It is in MSG_TIME format.

Error codes

0
: The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

EINVAL
: The current time value specified differs from the current timestamp by more than the allowable skew amount (30 ms)..

Figure 7-19. Message flow for W_SET_TIMESTAMP



8 ATM (specific operations)

This chapter describes additional STREAMS service messages and management commands needed for configuration and operation of the Multiplexed WAN driver in an ATM environment. The Multiplexed WAN driver implements the following ATM functions.

- Streaming mode service for ATM Adaptation Layer 5 (AAL5)
- AAL0 (raw mode) support
- Network Node Interface (NNI) format for the ATM layer
- Operation and Maintenance (OAM) support

Table 8-1. STREAMS service messages and management commands for ATM (Part 1 of 2)

Message	Use	Type	Direction	Page
WAN_DAT	To send and receive data to and from a VCC	Service Message M_PROTO	Up or down on appropriate stream	61
WAN_NOTIF_ATM	To inform upper level of ATM layer-related events	Service Message M_PROTO	Up only on all management streams (that is, streams on which WAN_SID has not been issued)	187
W_SET_ATM	To set ATM layer parameters	Management Command M_IOCTL	Down on any opened stream	190
W_GET_ATM	To get ATM layer parameters and its current state	Management Command M_IOCTL	Down on any opened stream	193
W_SET_SNID	To associate a VCC with an ATM layer and obtain internal channel ID	Management Command M_IOCTL	Down on any opened stream	94
W_SETTUNE	To set the parameters related to VCC	Management Command M_IOCTL	Down on any opened stream	83
W_DI_TEST_CFG	To set test configuration for a physical port	Management Command M_IOCTL	Down on any opened stream	137

Table 8-1. STREAMS service messages and management commands for ATM (Part 2 of 2)

Message	Use	Type	Direction	Page
W_GET_VCC_STATS	To get statistics of a VCC	Management Command M_IOCTL	Down on any opened stream	*195*
W_ZERO_VCC_STATS	To get and clear statistics of a VCC	Management Command M_IOCTL	Down on any opened stream	*198*
W_GET_ATM_STATS	To get statistics of an ATM layer	Management Command M_IOCTL	Down on any opened stream	*200*
W_ZERO_ATM_STATS	To get and clear statistics of an ATM layer	Management Command M_IOCTL	Down on any opened stream	*203*

STREAMS service messages for ATM

Message	Structure in M_PROTO	M_DATA?	Direction
WAN_DAT	wan_msg (see *WAN_DAT — Data messages for transmission and reception* on page *61*)	Yes	To and from WAN driver
WAN_NOTIF_ATM	wan_notif_atm (see *WAN_NOTIF_ATM — Notify ATM cell stream status* on page *187*)	Yes	To and from WAN driver

WAN_NOTIF_ATM — Notify ATM cell stream status

This message notifies the upper level of events related to the ATM cell stream. Reported events are:

- When the cell delineation process at the physical layer loses cell synchronization (that is, the state machine changes the state from SYNCH to HUNT).
- When the cell delineation process at the physical layer gains cell synchronization (that is, the state machine enters the SYNCH state).

This message is sent up by the WAN driver only on all management streams (that is, streams on which WAN_SID has not been issued).

The following structure is associated with this M_PROTO message:

```
struct wan_notif_atm {
   uint8    w_type;
   uint8    w_spare[3];
   uint32   w_phy_pipe_id;
   uint32   wan_event;
} ;
```

Parameters

w_type Output. This is set to WAN_NOTIF_ATM.

w_phy_pipe_id
Output. An event has been detected on the pipe specified by this identifier.

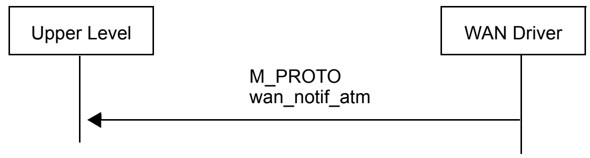
wan_event
Output. This indicates the events being reported. This is a bit-wise OR of the following values:

WAN_LOST_ATM_CELL_SYNCH
The ATM cell delineation process lost cell synchronization after ALPHA consecutive cells with incorrect HECs (Header Error Controls).

WAN_GAINED_ATM_CELL_SYNCH
The ATM cell delineation process has regained cell synchronization after DELTA consecutive cells with correct HECs.

When a WAN_LOST_ATM_CELL_SYNCH event is detected, a WC_DISC will be generated on a data stream (that is, WAN_SID has been issued) if the WAN_event_disc field in the W_SETTUNE command has the WAN_CELL_SYNC bit set. Likewise, a WC_CONNECT will be generated when a WAN_GAINED_ATM_CELL_SYNCH event is detected.

Figure 8-1. Message flow for WAN_NOTIF_ATM



STREAMS management commands for ATM.

ioc_cmd value of iocblk structure in M_IOCTL	Structure in M_DATA after M_IOCTL	M_DATA with M_IOCACK?
W_SET_ATM	wan_set_atm_ioc (see *W_SET_ATM — Define parameters for a physical layer of an ATM cell stream* on page *190*)	No
W_GET_ATM	wan_get_atm_ioc (see *W_GET_ATM — Obtain ATM physical-layer parameters and current state* on page *193*)	Yes, wan_get_atm_ioc
W_SET_SNID	wan_set_snid_ioc (see *W_SET_SNID — Allocate internal channel and associate SNID to it* on page *94*)	Yes, wan_set_snid_ioc
W_SETTUNE	wan_tnioc (see *W_SETTUNE — Set configuration* on page *83*)	Yes, wan_tnioc
W_DI_TEST_CFG	wan_ditestcfg_ioc (see *W_DI_TEST_CFG — Set test configuration for a physical port* on page *137*)	Yes, wan_ditestcfg_ioc
W_GET_VCC_STATS	wan_vcc_ioc (see *W_GET_VCC_STATS — Get statistics for a virtual channel* on page *195*)	Yes, wan_vcc_ioc
W_ZERO_VCC_STATS	wan_vcc_ioc (see *W_ZERO_VCC_STATS — Retrieve and clear statistics for a virtual channel* on page *198*)	Yes, wan_vcc_ioc
W_GET_ATM_STATS	wan_atm_ioc (see *W_GET_ATM_STATS — Get statistics for a physical ATM cell stream* on page *200*)	Yes, wan_atm_ioc
W_ZERO_ATM_STATS	wan_atm_ioc (see *W_ZERO_ATM_STATS — Retrieve and clear statistics for a physical ATM cell stream* on page *203*)	Yes, wan_atm_ioc

W_SET_ATM — Define parameters for a physical layer of an ATM cell stream

This management command is used for:

- Defining parameters related to cell delineation
- Performing single-bit error correction or not
- Scrambling E1 payload data

The following structure is associated with this command:

```
typedef struct _atm_parms_ {
   uint32     w_alpha;
   uint32     w_delta;
   uint32     w_scrambler_flag;
   uint32     w_error_correction_flag;
} atm_parms;
struct wan_set_atm_parms_ioc {
   uint8      w_type;
   uint8      w_spare[3];
   uint32     w_phy_pipe_id;
   atm_parms  w_atm_parms;
};
```

Parameters

IOCTL_COMMAND

The ioc_cmd field in struct iocblk should be W_SET_ATM.

w_type Input. This should always be WAN_SET_ATM.

w_phy_pipe_id

Input. This is a unique identifier associated with the combination of the time slots or ATM cell stream over which this ATM physical layer is operating. This was specified using the W_SET_PHY_PIPE command by the upper level.

w_atm_parms

Input. The following fields are defined for the structure:

w_alpha Input. This specifies the ALPHA value of the cell delineation state diagram, the number of consecutive HEC errors before going back to the HUNT state and reporting WAN_LOST_ATM_ CELL_ SYNCH using WAN_NOTIF_ATM. The default value is 7.

w_delta Input. This specifies the DELTA value of the cell delineation state diagram and the number of consecutive cells with correct HEC before entering the SYNCH state.

WAN_GAINED_ATM_CELL_SYNCH will be reported using WAN_NOTIF_ATM, if this is not the first time WAN enters SYNCH state. The default value is 8.

w_scrambler_flag

Input. This specifies whether the self-synchronizing scrambler is to be applied to the cell payload data. Use the `x**43 + 1` polynomial. This operation is defined in the *ITU-T I.432* and the *ITU-T G.804* specifications. A value of 1 activates the scrambler. The default value is 0, which does not perform scrambling.

w_error_correction_flag

Input. This specifies whether single-bit errors in the cell header are to be corrected. A value of 1 would make an attempt to correct a single-bit error. The default value is 0, which would not make an attempt to correct a single-bit error and would discard the cell. It is suggested that this value be set to 0, because DS1 or E1 lines incur multi-bit errors.

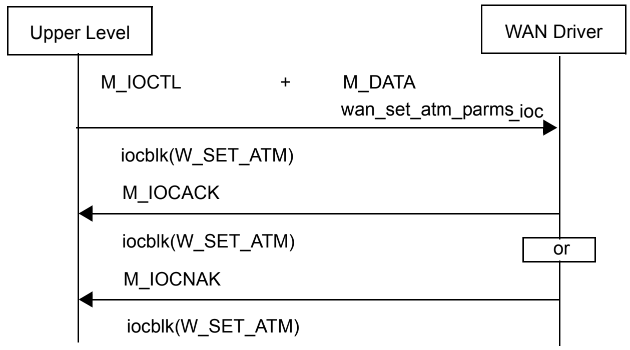
Currently, the WAN driver does not support this field.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK. In case of an error, an M_IOCNAK message is sent back with the appropriate error code.
EINVAL	The message size does not match.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.
ERANGE	One or more parameters do not contain the proper value.
ENODEV	The cell stream or physical pipe defined by the w_phy_pipe_id field is not defined.

- It is assumed that synchronization will be achieved on a byte boundary, and there is no need to perform a bit-by-bit synchronization.
- The ATM physical layer will be enabled when a WAN_REG is done for the first time on this pipe by any VCC (stream). Likewise, the ATM physical layer will be disabled when all VCCs (streams) associated with this pipe perform W_DISABLE, or all streams are closed.
- ATM physical-layer parameters can be altered only when the ATM physical layer is in the disabled state (that is, prior to the first WAN_REG, or after all W_DISABLEs are performed).

Figure 8-2. Message flow for W_SET_ATM



W_GET_ATM — Obtain ATM physical-layer parameters and current state

This management command is used to retrieve ATM physical-layer parameters that were set previously by the W_SET_ATM. It also returns the current state of the ATM physical layer.

The following structure is associated with this command:

```
typedef struct _atm_state_ {
   uint32      w_cell_delineation_state;
   uint32      w_cell_hdr_start_bit;
 atm_state;
}
struct wan_get_atm_ioc {
   uint8       w_type;
   uint8       w_spare[3];
   uint32      w_phy_pipe_id;
   atm_parms   w_atm_parms;
   atm_state   w_atm_state;
};
```

Parameters

IOCTL_COMMAND
: The ioc_cmd field in struct iocblk should be W_GET_ATM.

w_type
: Input. This should always be WAN_GET_ATM.

w_phy_pipe_id
: Input. This is a unique identifier associated with the combination of the time slots or ATM cell stream over which this ATM physical layer is operating.

w_atm_parms
: Output. See *W_SET_ATM — Define parameters for a physical layer of an ATM cell stream* on page 190 for a description of the fields for this structure.

w_atm_state
: Output. Informs the upper level of the state of the physical layer:

w_cell_delineation_state
: This represents the current state of the cell delineation state machine and it can be one of the following values: HUNT, PRESYNC, or SYNC.

w_cell_hdr_start_bit
: This represents the bit position where bit synchronization took place. Possible values range from 0–7, where 0 represents the
least-significant bit (rightmost).

Error codes

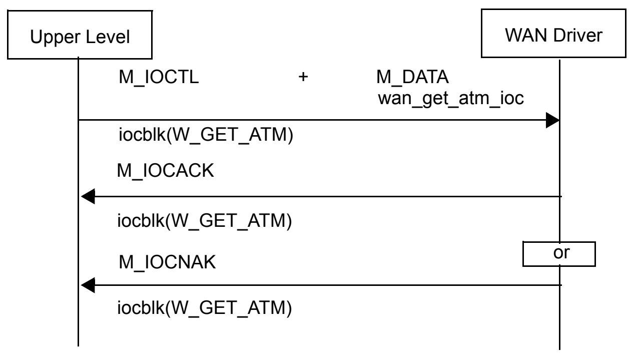
0 — The command was successfully processed. The IOCTL is acknowledged with M_IOCACK. In case of an error, an M_IOCNAK message is sent back with the appropriate error code.

EINVAL — The message size does not match.

ENXIO — A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.

ENODEV — The cell stream or physical pipe defined by the w_phy_pipe_id field is not defined.

Figure 8-3. Message flow for W_GET_ATM



W_GET_VCC_STATS — Get statistics for a virtual channel

This command is used for retrieving a virtual channel's accumulated statistics from the WAN Driver. These are maintained on a per-virtual-channel basis, and the virtual channel is selected by specifying the proper value in the w_snid field. The w_vcc_stats structure holds the returned statistics.

The following structure is associated with this command:

```
typedef struct _vcc_stats_ {
   uint32      tx_total;
   uint32      tx_oam_f5;
   uint32      rx_total_OK;
   uint32      rx_oam_f5;
   uint32      rx_Err_A;
   uint32      rx_Err_B;
   uint32      rx_Err_C;
   uint32      rx_Err_D;
   uint32      rx_Err_E;
   uint32      rx_Err_F;
   uint32      rx_Err_G;
   uint32      rx_nflow;
} vcc_stats;
struct wan_vcc_ioc {
   uint8       w_type;
   uint8       w_spare[3];
   uint32      w_snid;
   vcc_stats   w_vcc_stats;
}
```

Parameters

IOCTL_COMMAND
: The ioc_cmd field in struct iocblk should be W_GET_VCC_STATS.

w_type
: Input. This should always be WAN_GET_VCC_STATS.

w_snid
: Input. The subnetwork identifier associated with this stream or virtual channel.

w_vcc_stats
: Output. These are statistics collected since the last time the counters were cleared. The following fields are defined for the structure:

  tx_total
  : Output. If this stream is a CPCS stream, this represents the total number of CPCS SDUs that were transmitted. For all others, this represents the number of cells that were transmitted.

  tx_oam_f5
  : Output. Total number of F5 OAM cells that were transmitted on this VCC.

rx_total_OK
: Output. Total number of CPCS SDUs or cells that were received without any errors.

rx_oam_f5
: Output. Total number of F5 OAM cells that were received on this VCC.

rx_Err_A
: Output. Total number of CPCS SDUs that were received with Err_A or CRC-10 errors. See page 64 for an explanation of Err_A.

rx_Err_B
: Output. Total number of CPCS SDUs that were received with Err_B. See page 64 for an explanation of Err_B.

rx_Err_C
: Output. Total number of CPCS SDUs that were received with Err_C. See page 64 for an explanation of Err_C.

rx_Err_D
: Output. Total number of CPCS SDUs that were received with Err_D. See page 64 for an explanation of Err_D.

rx_Err_E
: Output. Total number of CPCS SDUs that were received with Err_E. See page 64 for an explanation of Err_E.

rx_Err_F
: Output. Total number of CPCS SDUs that were received with Err_F. See page 64 for an explanation of Err_F.

rx_Err_G
: Output. Total number of CPCS SDUs that were received with Err_G. See page 64 for an explanation of Err_G.

rx_nflow
: Output. Total number of CPCS SDUs or OAM cells that were dropped by the WAN driver. This happens when the upper level has flow control of the WAN driver.

Error codes

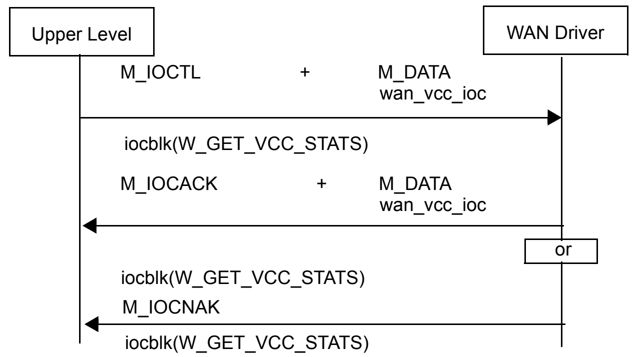
0
: The command was successfully processed. The IOCTL is acknowledged with M_IOCACK. In case of an error, an M_IOCNAK message is sent back with the appropriate error code.

EINVAL
: The message size does not match.

ENXIO
: A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.

ENODEV
: Either the SNID cannot be found among the SNIDs, or the SNID format cannot be deciphered.

Figure 8-4. Message flow for W_GET_VCC_STATS



W_ZERO_VCC_STATS — Retrieve and clear statistics for a virtual channel

This command is used for retrieving a virtual channel's accumulated statistics from the WAN Driver and then clearing them. The virtual channel is selected by specifying the proper value in the w_snid field. The w_vcc_stats structure holds the returned statistics. See *W_GET_VCC_STATS — Get statistics for a virtual channel* on page *195* for a description of the fields.

The following structure is associated with this command:

```
struct wan_vcc_ioc {
   uint8      w_type;
   uint8      w_spare[3];
   uint32     w_snid;
   vcc_stats  w_vcc_stats;
}
```

Parameters

IOCTL_COMMAND
: The ioc_cmd field in struct iocblk should be W_ZERO_VCC_STATS.

w_type
: Input. This should always be WAN_ZERO_VCC_STATS.

w_snid
: Input. The subnetwork identifier associated with this stream or virtual channel.

w_vcc_stats
: Output. These are statistics collected since the last time the counters were cleared. The upper level is responsible for adding these numbers with the previously acquired ones. See *W_GET_VCC_STATS — Get statistics for a virtual channel* on page *195* for a description of the fields for this structure.

Error codes

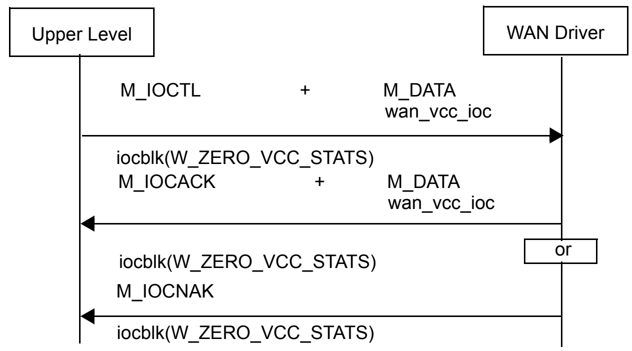
0
: The command was successfully processed. The IOCTL is acknowledged with M_IOCACK. In case of an error, an M_IOCNAK message is sent back with the appropriate error code.

EINVAL
: The message size does not match.

ENXIO
: A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset may remove the problem.

ENODEV
: Either the SNID cannot be found among the SNIDs, or the SNID format cannot be deciphered.

Figure 8-5. Message flow for W_ZERO_VCC_STATS



W_GET_ATM_STATS — Get statistics for a physical ATM cell stream

This command is used for retrieving physical ATM cell stream-related statistics from the WAN driver. These are maintained on a per-ATM-cell-stream basis, and the cell stream is selected by specifying the proper value in the w_phy_pipe_id field. The w_atm_stats structure holds the returned statistics.

The following structure is associated with this command:

```
typedef struct _atm_stats_ {
   uint32      tx_atm_cells;
   uint32      tx_oam_f5_cells;
   uint32      tx_cong_cells;
   uint32      tx_idles;
   uint32      rx_atm_cells;
   uint32      rx_oam_f5_cells;
   uint32      rx_cong_cells;
   uint32      rx_idles;
   uint32      rx_discard;
   uint32      rx_hec_errors;
} atm_stats;
struct wan_atm_ioc {
   uint8       w_type;
   uint8       w_spare[3];
   uint32      w_phy_pipe_id;
   atm_stats   w_atm_stats;
};
```

Parameters

IOCTL_COMMAND
The ioc_cmd field in struct iocblk should be W_GET_ATM_STATS.

w_type Input. This should always be WAN_GET_ATM_STATS.

w_phy_pipe_id
Input. Unique identifier associated with this cell stream as returned by the W_SET_PHY_PIPE command.

w_atm_stats
Output. These are statistics collected since the last time the counters were cleared. The following fields are defined for the structure:

tx_atm_cells
Output. Total number of ATM cells that were transmitted by the ATM layer.

tx_oam_f5_cells
Output. Total number of F5 OAM cells that were transmitted by the ATM layer.

tx_cong_cells
Output. Total number of congested cells that were transmitted by the ATM layer.

`tx_idles`
: Output. Total number of idle ATM cells that were transmitted by the physical layer.

`rx_atm_cells`
: Output. Total number of cells that were received for which a virtual channel was opened by the upper level.

`rx_oam_f5_cells`
: Output. Total number of F5 OAM cells that were received by the ATM layer.

`rx_cong_cells`
: Output. Total number of congested cells that were received by the ATM layer.

`rx_idles`
: Output. Total number of idle cells that were received and then discarded.

`rx_discard`
: Output. Total number of cells that were discarded (not including idles), because:

- A virtual channel (VCC is unknown) was not opened by the upper level on this pipe.
- No buffers are left to hold the incoming user cells.

`rx_hec_errors`
: Output. Total number of cells received with HEC errors.

Error codes

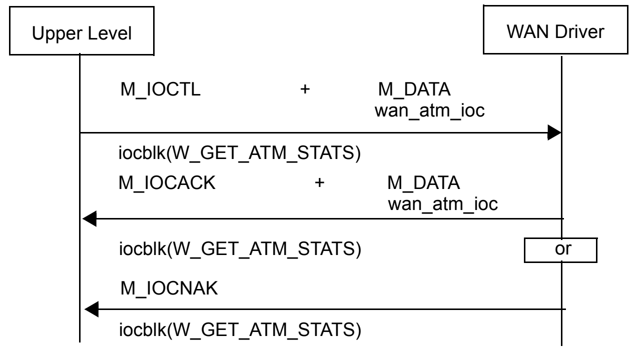
0 The command was successfully processed. The IOCTL is acknowledged with M_IOCACK. In case of an error, an M_IOCNAK message is sent back with the appropriate error code.

EINVAL The message size does not match.

ENXIO A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.

ENODEV The pipe ID cannot be found among the pipe IDs.

Figure 8-6. Message flow for W_GET_ATM_STATS



W_ZERO_ATM_STATS — Retrieve and clear statistics for a physical ATM cell stream

This command is used for retrieving a physical ATM cell stream's accumulated statistics from the WAN driver and then clearing them. The physical ATM cell stream is selected by specifying the proper value in the w_phy_pipe_id field. See *W_GET_ATM_STATS — Get statistics for a physical ATM cell stream* on page 200 for a description of the fields for the w_atm_stats structure.

The following structure is associated with this command:

```
struct wan_atm_ioc {
   uint8      w_type;
   uint8      w_spare[3];
   uint32     w_phy_pipe_id;
   atm_stats  w_atm_stats;
}
```

Parameters

IOCTL_COMMAND
: The ioc_cmd field in struct iocblk should be W_ZERO_ATM_STATS.

w_type
: Input. This should always be WAN_ZERO_ATM_STATS.

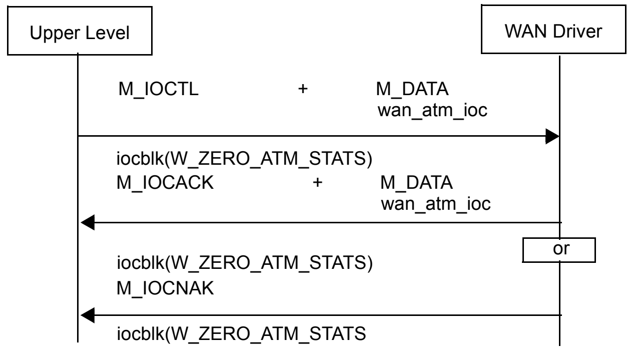
w_phy_pipe_id
: Input. The subnetwork identifier associated with this physical ATM cell stream.

w_atm_stats
: Output. These are statistics collected since the last time the counters were cleared. The upper level is responsible for adding these numbers with the previously acquired ones. See *W_GET_ATM_STATS — Get statistics for a physical ATM cell stream* on page 200 for a description of the fields for this structure.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK. In case of an error, an M_IOCNAK message is sent back with the appropriate error code.
EINVAL	The message size does not match.
ENXIO	A severe hardware error has occurred. Run diagnostics to find out more about the type of failure. A card reset might remove the problem.
ENODEV	The pipe ID cannot be found among the pipe IDs.

Figure 8-7. Message flow for W_ZERO_ATM_STATS



9 Extensions to Serial WAN driver provided by RadiSys

This chapter provides extensions to the Serial WAN driver provided by RadiSys. The Serial WAN driver is a combination of the existing Spider/Shiva X.25 WAN driver, r8.0 and RadiSys proprietary implementation. For a description of Shiva's implementation of the Serial WAN driver, refer to *SpiderX25 WAN Implementation Guide, r8.0* by Spider Systems. This chapter describes only the RadiSys extensions. The Serial WAN driver allows access to a device providing serial physical interfaces. Serial interfaces are of the type:

- RS-232
- RS-422
- V.35
- V.36
- X.21

The Serial WAN driver operates in synchronous, asynchronous, and/or bisynchronous modes, depending on the installed PMC or AIB. Not all interface types are supported on all PMCs and AIBs.

Interacting with the Serial WAN driver

A *non-clone open* on the Serial WAN driver creates a stream to the Serial WAN driver that can carry STREAMS messages for:

- Subnetwork identifier (SNID) assignment
- Upper-layer registration
- Data transfer control
- Transmission of frames
- Reception of frames

The SNID acts as the line identifier in all management commands. It allows management commands to be carried on any of the streams opened to the Serial WAN driver. The management commands include the following operations:

- Setting and obtaining configuration parameters for a line
- Clearing and obtaining statistics on the line
- Control of the interface address mapping

Serial WAN driver STREAMS interface

The STREAMS interface of the existing Serial WAN driver is composed of two types of messages:

Service messages

M_PROTO messages that control and provide the reception or transmission of frames for the line associated with the stream.

See the following messages and their descriptions for information.

- *WAN_SID — Set subnetwork ID* on page *51*
- *WAN_REG — Registration message — start hardware* on page *54*
- *WAN_CTL — Connection management* on page *56*
- *WAN_DAT — Data messages for transmission and reception* on page *61*

SS7-related service messages are described in *Chapter 6, Signaling System Number 7 (SS7) (specific operations)* on page *103*.

Management commands

M_IOCTL messages that allow management (parameters setting and statistics) of the different lines.

See the following commands and their descriptions for information.

- *W_GETSTATS — Get statistics* on page *78*
- *W_ZEROSTATS — Clear channel statistics* on page *81*

STREAMS service message for the Serial WAN driver

Message Type	Direction	Structure and Parameters	Use
WAN_NOTIFY	Up	wan_nty • Status being reported • Extra diagnostic information	To inform the upper level that an unsolicited status change has occurred in the hardware.

WAN_NOTIFY — Notification of errors

This message notifies the upper process of a hardware error or change in control signals. The following structure is associated with this M_PROTO message:

```
struct wan_nty {
   uint8    w_type;
   uint8    w_spare[3];
   uint32   w_snid;
   uint32   w_eventstat;
   uint32   w_reserved1;
   uint32   w_reserved2;
}
```

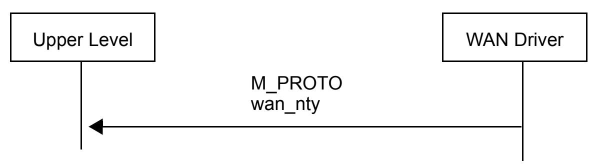
Parameters

w_type Output. This is set to WAN_NOTIFY.

w_snid Output. The subnetwork identifier. See *WAN_SID — Set subnetwork ID* on page *51* for a description of the wan_snid parameter.

w_eventstat
Output. When an event occurs, the appropriate bit is set to 1. One or more of the events are set, depending on which events were defined in the w_notifymask field of the W_SETLINE command. See page *211* for a description of the events.

Figure 9-1. Message flow for WAN_NOTIFY



STREAMS management commands for the Serial WAN driver

The management of the WAN driver is performed through the ioctl system call mechanism using the I_STR command of STREAMS. All ioctl system calls are replied to by the WAN driver by setting the ioctl message block type to M_IOCACK or M_IOCNAK for success or failure, and calling qreply to return the message to the user level.

ioctl Command	M_DATA content besides SNID	Use	See Page
W_SETLINE	wan_setlinef and table of tuning values	To set the configurable parameters for a line	209
W_GETLINE	wan_setlinef and table of tuning values	To obtain the configurable parameters for a line	220
W_SETSIG	wan_setsigf and Signals bit map	To control the modem control signals	221
W_GETSIG	wan_setsigf and Signals bit map	To obtain the modem control signals	223
W_RESET	wan_resetf	To reset the Serial Communication Controller	225
W_SENDBREAK	wan_sendbreakf	To send break signal in asynchronous communications	227
W_SETMODE	wan_setmodef	To set the operating mode in asynchronous communications	229
W_STATIONADDR	wan_stationaddrf and address of the station for receiving data frames	To control address filtering	231

W_SETLINE — Define line characteristics

This command is used to set the configurable parameters of the line characteristics.

The W_SETLINE and W_GETLINE extension commands are functionally equivalent to the W_SETTUNE and W_GETTUNE commands, except that W_SETLINE supports more configurable options. The use of only one of these two commands is supported if the defaults are not sufficient.

To change the supported configuration parameters using the W_SETLINE command, the port must be in an open state but not yet registered.

- To change configuration parameters for w_maxdatasize and w_bitratetrans on a port that is sending or receiving data, the port must be closed and reopened, and the W_SETLINE command must be issued before the registration (WAN_REG) command.
- For BISYNC, it is always necessary to issue a W_SETLINE command when specifying BISYNC protocol, because HDLC is the default protocol. If any W_SETLINE parameters need to be changed, the port must be closed and then opened before another W_SETLINE can be done.
- To use the X.21 interface, you must issue the W_SETLINE command with the w_porttype parameter equal to WAN_X21 and have the X.21 cable attached. You need to do this to initialize the hardware into X.21 mode.

The following structure is associated with this command:

```
 struct sync_setline {
      uint8    w_encoding;
      uint8    w_crc;
      uint8    w_shareflag;
      uint8    w_idlepat;
      uint8    w_options;
      uint8    w_cptype;
      uint8    w_extspeed;
      uint8    w_elementtiming;
  }

 struct async_setline {
      uint8    w_stopbits;
      uint8    w_parity;
      uint8    w_transmode;
      uint8    w_xonchar;
      uint8    w_xoffchar;
      uint8    w_spare[3];
  }

struct wan_setlinef {
    uint8      w_type;
    uint8      w_spare[3];
    uint32     w_snid;
    uint32     w_wanver;
    uint32     w_notifymask;
    uint16     w_maxdatasize;
    uint16     w_reserved;
    uint8      w_portmode;
    uint8      w_connmask;
    uint8      w_commtype;
    uint8      w_databits;
    uint8      w_porttype;
    uint8      w_maxtransmits;
    uint32     w_bitratercv;
    uint32     w_bitratetrans;
    uint8      w_spare2[8];
    union {
       struct sync_setline  s_params;
       struct async_setline a_params;
    }params;
  };
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_SETLINE.

w_type
: Input. This is set to WAN_SETLINE.

w_snid
: Input. The subnetwork identifier. See the description of the wan_snid parameter on page *51*.

w_wanver
: Input. This field is used to coordinate version numbers with the upper level. This command is not supported. Use the W_GETDRVINFO command to get the WAN version.

`w_notifymask`
: Input. A set of bits indicating which events cause notification to the upper level:

  `W_RECEIVE_BUFFER_OVFL`
  : The receive buffer is not big enough.

  `W_FRAMING_ERROR`
  : - CRC error for synchronous mode
    - Invalid frame type for bisynchronous mode

  `W_TIMEOUT`
  : Receive timeout

  `W_HD_OVERRUN`
  : Hardware overrun error

  `W_ATTACHED_DEV_INACT`
  : No clock from attached device. This function is supported only for the X.21 electrical interface. Some of the multiple causes of device inactive conditions are:
    - Cable clocking problems
    - Local attached device powered-off or malfunctioning
    - Cable disconnected

  `W_ATTACHED_DEV_ACTIVE`
  : Clock received from attached device. This function is supported only for the X.21 electrical interface.

  `W_FCS_ERR`
  : Error in FCS calculation

  `W_CTS_ON` CTS signal asserted

  `W_CTS_OFF`
  : CTS signal negated

  `W_DCD_ON` DCD signal asserted

  `W_DCD_OFF` DCD signal negated

  `W_DSR_ON` DSR signal asserted

  `W_DSR_OFF`
  : DSR signal negated

  `W_RI_ON` RI signal asserted

  `W_RI_OFF` RI signal negated

  `W_PARITY_ERROR`
  : Parity error

  `W_BREAK_DETECTED`
  : Break detected in asynchronous mode

W_SHORT_FRAME
SDLC short frame

W_TX_UNDERRUN
Transmit DMA underrun

W_ABORT SDLC abort frame

W_RCL_NOTZERO
SDLC RCL not zero. Last character of I frame did not have correct size.

W_BSC_PAD_ERR
BISYNC pad error

W_CTS_UNDERRUN
CTS signal negated during transmission

w_maxdatasize
Input. The maximum transmit or receive data size. The Serial WAN driver will add an additional byte to w_maxdatasize before allocating receive buffers. The default is 256 for synchronous and 2048 for asynchronous mode.

w_portmode
Input. Options for port mode:

W_NO_CHANGE
No change to previous selection of default.

W_ASYNC_3309_FRM
OSI 3309 asynchronous framing mode

W_ASYNC_AWP2224_FRM
AWP224 framing

W_SYNC Synchronous HDLC Framing

W_BSC BISYNC protocol. The WAN strips leading and ending control characters before giving received characters to upper level.

w_connmask

Input. A byte that defines the control-signal connections. The byte is a series of bit flags, one per control signal. The following provides the bit assignments for each control signal. The default is all control signal lines connected for synchronous mode. This option is not supported for X.21; the X.21 is leased-line only and defaults to the leased-line value.

W_RTS_CONNECT
Connect RTS

W_CTS_CONNECT
Connect CTS

W_DSR_CONNECT
Connect DSR

W_DTR_CONNECT
Connect DTR

W_DCD_CONNECT
Connect DCD

W_RI_CONNECT
Connect RI

w_commtype

Input. Indicates if the line is duplex. The following values are valid for this parameter:

W_NO_CHANGE
No change to previous selected value.

W_DUPLEX Duplex. This is the default.

W_H_DUPLEX
Half duplex. This is ignored for BISYNC, which is half-duplex only.

`w_databits`

Input. The number of data bits per character. The following values are valid for this parameter:

`W_NO_CHANGE`
No change to previous selected value.

`W_5_BPC` 5 bits per character

`W_6_BPC` 6 bits per character

`W_7_BPC` 7 bits per character

`W_8_BPC` 8 bits per character. This is the default.

- For BISYNC, w_databits is ignored.
- If w_encoding is:
  - W_BSC_ASCII, the w_databits value will be set to W_7_BPC
  - W_BSC_EBCDIC, the w_databits value will be set to W_8_BPC

`w_porttype`

Input. Except for the WAN_T1E1 field, the fields for w_porttype are the same as those for the WAN_interface field for W_SETTUNE. See *W_SETTUNE — Set configuration* on page 83 for a description of the fields.

- WAN_V35
- WAN_V36
- WAN_RS232
- WAN_RS422
- WAN_2PORT_2TYPE
- WAN_X21

- When the cable type is selected using the W_SETLINE command's w_porttype parameter, the Serial WAN driver will test the cable type attached and select the interface based on the cable attached. Issue a W_GETLINE command to verify the cable type attached.
- To use the X.21 electrical interface, a W_SETLINE command must be issued with the w_porttype parameter set to WAN_X21, with the X.21 cable attached. If this is not done, the X.21 cable can be used without the X.21 electrical interface.

w_maxtransmits
: Input. The number of outstanding incomplete transmit commands (not supported)

w_bitratercv
: Input. Not used for synchronous applications. The receive clock is always external. See the following w_bitratetrans field for the values for this parameter.

w_bitratetrans
: Input. The number of bits per second at which the communication chips should receive and transmit data.

For HDLC, the fields for w_bitratetrans are the same as those for the WAN_baud field for W_SETTUNE. See *W_SETTUNE — Set configuration* on page 83 for a description of the fields.

- W_EXT_CLK_VERF_TXC
- W_DCE_INT_XTC_EXT_RXC
- W_DCE_INT_XTC_INT_RXC
- W_DTE_CLK_FROM_TXC
- W_DTE_TX_FROM_TXC_RX_FROM_RXC

w_stopbits
: Input. The number of stop bits to be used by the asynchronous transmitter and receiver circuits for character encoding and decoding. This parameter is asynchronous-specific.

W_NO_CHANGE
: No change to default or selected value.

W_1_STOP_BIT
: 1 stop bit.

W_15_STOP_BIT
: 1.5 stop bits.

W_2_STOP_BIT
: 2 stop bits.

w_parity Input. Parity is the parity used for each character encoding and decoding. This parameter is asynchronous-specific. The following values are valid for this parameter:

W_NO_CHANGE
: No change to default or selected value.

W_NO_PARITY
: No parity.

W_EVEN_PARITY
: Even parity.

W_ODD_PARITY
: Odd parity.

w_transmode

Input. This parameter defines the transparency mode for the port. This parameter is asynchronous-specific. The following values are valid for this parameter:

W_NO_CHANGE
No change to default or selected value.

W_FLOW_CNTL_TRANS
Flow-control transparency.

W_FULL_TRANS
Full transparency.

w_xonchar

Input. This byte contains the XON character (not supported). This parameter is asynchronous-specific.

w_xoffchar

Input. This byte contains the XOFF character (not supported). This parameter is asynchronous-specific.

w_encoding

Input. The data encoding method to be used. The following values are valid for this parameter:

W_NO_CHANGE
No change to default or selected value

W_NRZ NRZ

W_NRZI NRZI

W_FM1 FM1

W_FM0 FM0

If w_portmode is W_BSC, the following values are valid for this parameter:

W_BSC_ASCII
ASCII BISYNC. This is the default.

W_BSC_EBCDIC
EBCDIC BISYNC

w_crc Input. The CRC indicates which CRC calculation and which preset values are to be used. The following values are valid for this parameter:

W_NO_CHANGE
No change to default or selected value.

W_CRC_CCITT_0
CRC-CCITT, preset to 0's.

W_CRC_CCITT_1
CRC-CCITT, preset to 1's. This is the default.

W_CRC16_0
CRC-16, preset to 0's.

W_CRC16_1
CRC-16, preset to 1's.

W_LRC8_0 LRC-8, preset to 0's.

- For BISYNC, w_crc is ignored.
- If w_encoding is:
  - W_BSC_ASCII, the w_crc value will be set to W_LRC8_0
  - W_BSC_EBCDIC, the w_crc value will be set to W_CRC16_0

w_shareflag
Input. Indicates if the frame synchronization flags should be shared, if possible, when processing WRITE commands. Sharing the flags means that the closing flag of the last transmitted frame becomes the starting flag of the new outgoing frame.

W_NO_SHARE_FLAG
Do not share flags.

W_SHARE_FLAG
Share flags. This is the default.

w_idlepat
Input. The pattern sent when the link connection is in an idle state.

W_NO_CHANGE
Do not change chosen pattern.

W_MARK Mark pattern. This is the default.

W_FLAG Flag pattern.

For BISYNC, w_idlepat is ignored because BISYNC will idle marks, except after transmission of an ITB message, in which case BISYNC will idle SYNs.

`w_options`
: Input. Options for addressing.

  `W_NO_TRANSLATE`
  : No address translation.

  `W_TRANSLATE`
  : Address translation. (This option is not supported.)

`w_cptype`
: Input. Call control procedure. The following value is valid for this parameter:

  `W_NO_CP`
  : No call control procedure supported.

`w_extspeed`
: Output. Returns the external line speed. (This option is not supported.)

`w_elementtiming`
: Input. Identifies if DCE signal element timing is supported for X.21. This option supports the following parameters:

  `DCE_SET_SUPPORTED`
  : Set this parameter if the DCE supports signal element timing. This causes the WAN to set hardware parameters that synchronize data with the clock signal. This parameter is recommended for high speeds.

  `DCE_SET_NOT_SUPPORTED`
  : Set this parameter if the DCE does not support signal element timing. The WAN will not program the hardware to output the clock back to the DCE.

Error codes

0
: The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

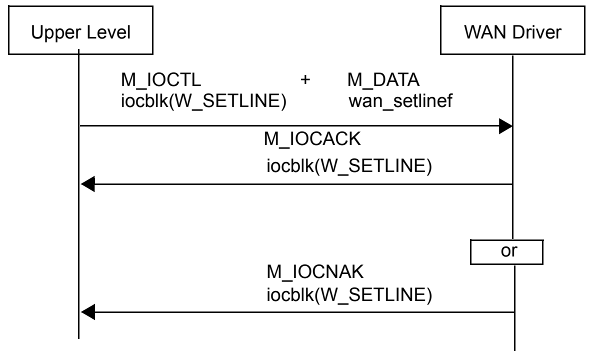
EINVAL
: The message size does not match.

ENODEV
: Either the SNID cannot be found among the port SNIDs, or the SNID format cannot be deciphered.

ECONNREFUSED
: Incorrect version of the WAN driver.

E2BIG
: The host’s maximum receive-buffer size is too small to hold the largest frame.

Figure 9-2. Message flow for W_SETLINE



W_GETLINE — Get line characteristics

This is an IOCTL command with the ioc_cmd field in struct iocblk set to W_GETLINE.

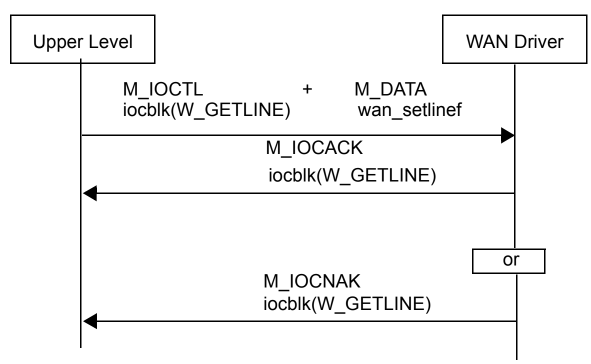
This command returns the line configuration values to the upper level. The parameters were previously configured by a W_SETLINE command, or the default values were used if the upper level has not configured the line.

There is no structure associated with this command. It is passed in the wan_setlinef structure with the w_type field set to WAN_GETLINE. See *W_SETLINE — Define line characteristics* on page *209* for more details on the structure.

Error codes

0	The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.
EINVAL	The message size does not match.
ENODEV	Either the SNID cannot be found among the port SNIDs, or the SNID format cannot be deciphered.

Figure 9-3. Message flow for W_GETLINE



W_SETSIG — Output signal control

This command allows the process to control the state of the output control signals.

This command is not recommended for X.21 protocol as the WAN is setting and timing signals.

The following structure is associated with this command:

```
typedef struct  wan_setsig {
       uint8        w_ctrlsignal;
       uint8        w_reserved1[3];
}wan_setsig_t;

struct wan_setsigf {
   uint8              w_type;
   uint8              w_spare[3];
   uint32             w_snid;
   wan_setsig_t       wan_setsig;
};
```

Parameters

IOCTL_COMMAND
Input. The ioc_cmd field in struct iocblk should be W_SETSIG.

w_type Input. This is set to WAN_SETSIG.

w_snid Input. The subnetwork identifier. See *WAN_SID — Set subnetwork ID* on page *51* for a description of the wan_snid parameter.

w_ctrlsignal
Input. Indicates the state of the output control signals and which input signals are required for transmission. CTRLSIGNAL is used for all electrical interfaces.

The signals are controlled by the process setting the appropriate bits.

W_RTS_HIGH
RTS is set high.

W_DTR_HIGH
DTR is set high.

W_RTS_LOW RTS is set low.

W_DTR_LOW DTR is set low.

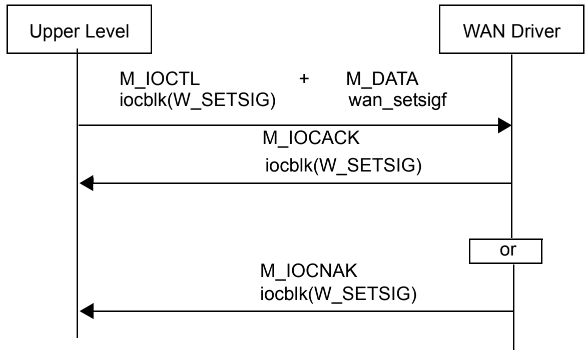
Error codes

0 The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

EINVAL The message size does not match.

ENODEV Either the SNID cannot be found among the port SNIDs, or the SNID format cannot be deciphered.

Figure 9-4. Message flow for W_SETSIG



W_GETSIG — Return the states of control

This command allows the WAN driver to return the current control signals.

The structure is the same as for the W_SETSIG command. See *W_SETSIG — Output signal control* on page 221 for more details.

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_GETSIG.

w_type
: Input. This is set to WAN_GETSIG.

w_snid
: Input. The subnetwork identifier. See the description of the wan_snid parameter on page 51.

w_ctrlsignal
: Output. Indicates the state of the input and output control signals.

  W_RTS_HIGH
  : RTS is high. (RTS denotes C signal in X.21.)

  W_DTR_HIGH
  : DTR is high.

  W_DCD_HIGH
  : DCD is high.

  W_DSR_HIGH
  : DSR is high.

  W_CTS_HIGH
  : CTS is high. (CTS denotes I signal for X.21.)

  W_RI_HIGH
  : RI is high.

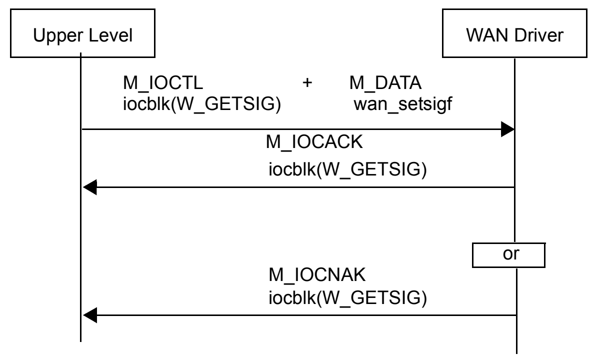
Error codes

0 — The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

EINVAL — The message size does not match.

ENODEV — Either the SNID cannot be found among the port SNIDs, or the SNID format cannot be deciphered.

Figure 9-5. Message flow for W_GETSIG



W_RESET — Reset communications chip

This command resets the Serial Communication Controller interface hardware for a port.

The following structure is associated with this command:

```
typedef struct  wan_reset {
        uint8       w_resettype;
        uint8       w_reserved1[3];
}wan_reset_t;

struct wan_resetf {
   uint8            w_type;
   uint8            w_spare[3];
   uint32           w_snid;
   wan_reset_t      wan_reset;
};
```

Parameters

IOCTL_COMMAND
Input. The ioc_cmd field in struct iocblk should be W_RESET.

w_type Input. This is set to WAN_RESET.

w_snid Input. The subnetwork identifier. See the description of the wan_snid parameter on page 51.

w_resettype
Input. This identifies the type of reset requested. A soft reset clears the read and write queues, but continues to use the line configuration parameters from W_SETTUNE and W_SETLINE. A hard reset returns the port to default values and clears both queues.

W_SOFT_RESET
Soft reset.

W_HARD_RESET
Hard reset.

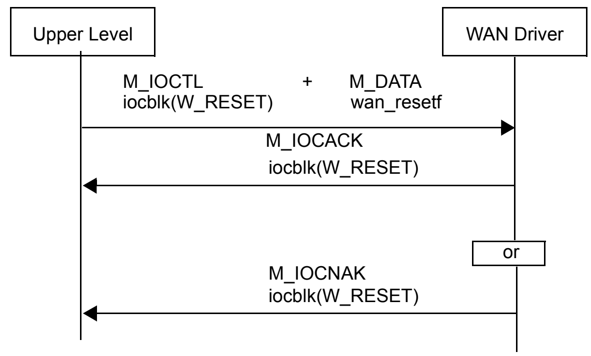
Error codes

0 The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

EINVAL The message size does not match.

ENODEV Either the SNID cannot be found among the port SNIDs, or the SNID format cannot be deciphered.

Figure 9-6. Message flow for W_RESET



W_SENDBREAK — Send break character

This is an asynchronous-only command. It causes the WAN driver to transmit a break signal for one character time plus 50 times the duration value. The maximum break duration is 500 ms or a w_duration value of 0xa.

The following structure is associated with this command:

```
typedef struct  wan_sendbreak {
       uint16       w_duration;
       uint16       reserved1;
 }wan_sendbreak_t;

struct wan_sendbreakf {
   uint8            w_type;
   uint8            w_spare[3];
   uint32           w_snid;
   wan_sendbreak_t  wan_sendbreak;
};
```

Parameters

IOCTL_COMMAND
Input. The ioc_cmd field in struct iocblk should be W_SENDBREAK.

w_type Input. This is set to WAN_SENDBREAK.

w_snid Input. The subnetwork identifier. See the description of the wan_snid parameter on page *51*.

w_duration
Input. This is the time in 50-ms units for the break signal.

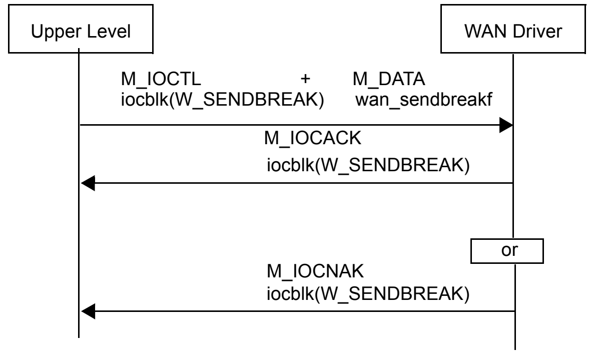
Error codes

0 The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

EINVAL The message size does not match.

ENODEV Either the SNID cannot be found among the port SNIDs, or the SNID format cannot be deciphered.

Figure 9-7. Message flow for W_SENDBREAK



W_SETMODE — Set port mode

This command sets the operating mode for this port.

The following structure is associated with this command:

```
typedef struct  wan_setmode {
       uint16       w_mode;
       uint16       reserved1;
 }wan_setmode_t;

struct wan_setmodef {
   uint8            w_type;
   uint8            w_spare[3];
   uint32           w_snid;
   wan_setmode_t    wan_setmode;
};
```

Parameters

IOCTL_COMMAND
: Input. The ioc_cmd field in struct iocblk should be W_SETMODE.

w_type Input. This is set to WAN_SETMODE.

w_snid Input. The subnetwork identifier. See the description of the wan_snid parameter on page 51.

w_mode Input. The port mode parameter must be:

W_NDIS NDIS protocol.

W_CONN_MANGT
: Connection management. This is the default.

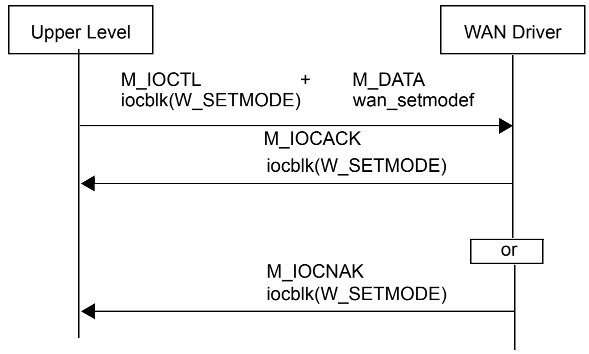
Error codes

0 The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

EINVAL The message size does not match.

ENODEV Either the SNID cannot be found among the port SNIDs or the SNID format cannot be deciphered.

Figure 9-8. Message flow for W_SETMODE



W_STATIONADDR — Set filtering address

This command sets the filtering address for HDLC protocol, if the default station address (0xff) is not desired.

The following structure is associated with this command:

```
typedef struct  wan_stationaddr {
        uint16        w_stationaddr;
        uint8         w_sasize;
        uint8         w_reserved3[16];
}wan_stationaddr_t;

struct wan_stationaddrf {
   uint8              w_type;
   uint8              w_spare[3];
   uint32             w_snid;
   wan_stationaddr_t  wan_stationaddr;
};
```

Parameters

IOCTL_COMMAND
Input. The ioc_cmd field in struct iocblk should be W_STATIONADDR.

w_type Input. This is set to WAN_STATIONADDR.

w_snid Input. The subnetwork identifier. See the description of the wan_snid parameter on page 51.

w_stationaddr
Input.

- The address of the station for receiving data frames. The address contained in the data frame must match this address or the broadcast address (`FFh` or `FFFFh`) for the frame's acceptance.
- A value of 0 turns off address filtering. This is the default. If the station address is one byte, the application must ensure that the high byte of the parameter is 0. To disable address filtering, set the w_stationaddr parameter to 0 and the w_sasize parameter to 1 or 2.
- For BISYNC, the poll or select address of the station for receiving data messages. The message containing the address must be of the format D ENQ, where D is a 1- or 2-byte address. The address in the ENQ message must match the poll or select address, or the broadcast address (`FFh` or `FFFFh`) for the message's acceptance. All further messages will be accepted until an EOT is received. To begin receiving messages again, an ENQ with the correct address must be received.

  Setting w_stationaddr to `0000h` turns off address filtering. This is the default.

w_sasize Input. Indicates the size of the station address. The default size is one byte.

W_NO_CHANGE
This option is not supported.

W_1BYTE_ADDR
1-byte station address.

W_2BYTE_ADDR
2-byte station address.

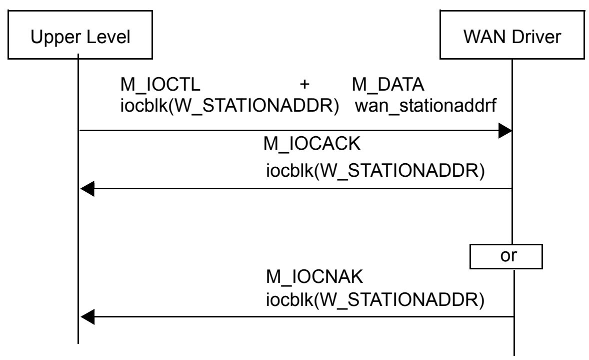
Error codes

0 The command was successfully processed. The IOCTL is acknowledged with M_IOCACK in the reverse direction. In case of an error, an M_IOCNAK message is sent upstream with the appropriate error code.

EINVAL The message size does not match.

ENODEV Either the SNID cannot be found among the port SNIDs, or the SNID format cannot be deciphered.

Figure 9-9. Message flow for W_STATIONADDR



10 Configuration and program development

This chapter provides WAN driver load-time configuration, initial port characteristics, and program development information.

Executable files required for ARTIC environments

Because the WAN driver is an On-Card STREAMS Subsystem (OSS) driver, after resetting the card, depending on the environment (ARTIC960 or ARTIC 1000/2000 Series), you must load the ARTIC executable files *in the order listed* before loading WAN drivers on the card.

These executable files are supplied with the RadiSys ARTIC runtime support. A basic load script or command file is also supplied. The order for loading these executable files is the same order as that shown in the following sections, unless overridden by the supplied load script or specified otherwise in the guide and reference publications for the ARTIC960 or ARTIC 1000/2000 Series environment.

ARTIC960 environment

1. ric_kern.rel

   The on-card base operating system. See *7. ric_skrn.rel* on page 234, which can be used instead of ric_kern.rel.

2. ric_pci.rel

   The local PCI-bus configuration device driver that identifies PCI resources located on the PMC.

3. ric_oss.rel

   The on-card STREAMS emulation system. See *7. ric_skrn.rel* on page 234, which can be used instead of ric_oss.rel.

4. ric_mcio.rel

   The peer-to-peer and system-unit access subsystem. This is required only if ric_scb.rel is to be loaded.

5. ric_scb.rel

   The SCB (subsystem control block) protocol-handler subsystem. This is required only if the cross-bus driver requires it. Check the cross-bus driver documentation for its requirements. For the SCB to work correctly, the system-unit utility riccnfg needs to be run on the host system. The riccnfg utility sets up the communication areas for SCB between the ARTIC cards and the system units.

6. ric_ess.rel

   The cross-bus driver for all RadiSys solutions based on the STREAMS environment. This requires that ric_scb.rel be loaded.

7. ric_skrn.rel

   This combines the functions ric_kern.rel and ric_oss.rel to improve performance. If you load ric_skrn.rel, you must load it first, to replace *1. ric_kern.rel* on page 233 and *3. ric_oss.rel* on page 233.

ARTIC 1000/2000 Series environment

1. rpq_skrn.rel

   The on-card STREAMS-based operating system.

2. rpq_cxb.rel

   The cross-bus driver that provides STREAMS message transfers to and from the host environment.

Load WAN drivers

After successful loading of the basic support, the WAN driver can be loaded. Choose the correct executable file based on the protocol needed and the hardware installed on the card. For the required hardware type and the .rel file, see *Table 1-3, "Summary of supported hardware with ARTIC adapters,"* on page 4 for the ARTIC960 and ARTIC 1000/2000 Series environments.

Unless the driver is loaded, the mismatch between the driver and the hardware cannot be known. The base support may not detect and inform the type of mezzanine hardware installed. However, the WAN driver will query the hardware during its initialization and abort if a mismatch is detected.

After being loaded, the WAN driver analyzes the command-line parameters and proceeds to do the static initialization. During this process, it queries the hardware and gets static resources based on the hardware configuration and the command-line options. A success or failure at this step results in the driver process being unloaded from the card. You can learn if there is a failure to load the driver successfully by setting the wait option on the load command for the driver.

For details on the Application Loader utility, refer to the guide and reference publications for the ARTIC960 or ARTIC 1000/2000 Series environment. See *Reference publications* on page *xiv* for a list of RadiSys reference books.

See *Table 10-2* on page 241 for a list of the initialization errors.

Command-line parameters

The WAN drivers take command-line parameters for the various load-time options that are mentioned throughout this book. These parameters can be specified using the Application Loader utility. Refer to the appropriate RadiSys reference book for your adapter for information on how to specify the Application Loader parameters. See *Reference publications* on page *xiv* for a list of RadiSys reference books.

The parameters are specified as follows:

- <Name>=<Value> with no intervening space
- A space separates two parameter specifications.
- In a parameter file, a new line also separates parameter specifications.
- Character strings are not case-sensitive.
- Numbers can be specified in C style; for example, decimal 20 could be specified as 0x14.

MAX_NON_CLONE

Integer. This defines the maximum number of non-clone opens for the WAN driver. The value can be 0 or a positive number, but cannot be more than MAX_OPENS.

For the default, see *Table 1-3* on page *4* for the maximum number of logical channels for the hardware.

MAX_OPENS

Integer. This indicates the total number of streams to be opened to the WAN driver. The number is based on the logical channels for the hardware, the desired number of *clone opens* for normal protocol operation, and those for activities such as configuration, monitoring, and control.

This parameter *must* be nonzero and greater than, or equal to, the MAX_NON_CLONE parameter.

The default is the maximum number of data streams the hardware can support plus 1.

SNID_DECODE

This parameter controls whether the SNID in commands and messages is to be decoded and, if yes, then how. See *WAN_SID — Set subnetwork ID* on page *51* for more details on the decoding methods. Possible values are:

- YES — SNID decoding is performed. Not supported in ATM mode or with pipes.
- NO — No SNID decoding is performed. This is the default.

`SNID_KEY` Integer. See *WAN_SID — Set subnetwork ID* on page *51* for details.

The default is the ASCII code for character ‘c’ (0x63).

`DATA_MSG_ONLY`

This parameter controls whether the WAN driver uses the WAN_DAT interface, where an M_PROTO header is attached to every data block in either direction. Setting it to YES eliminates the overhead incurred during the normal data path. Setting DATA_MSG_ONLY to YES is not supported for BISYNC. Possible values and the default values are as follows:

YES — Defaults to YES if ONE_DATA_MSG_ONLY is set to YES.

NO — Defaults to NO if ONE_DATA_MSG_ONLY is set to NO.

`ONE_DATA_MSG_ONLY`

Possible values are:

- YES — There will not be an M_PROTO header for the WAN_DAT interface, and the data will be contained in one block in either direction.
- NO (Default)

`TEST_INTERFACE`

Possible values are:

- YES — Tests the hardware by performing an internal loopback at driver initialization time. The driver will not load if errors are found.
- NO (Default)

`TX_BLKS` Integer. This is a performance-tuning parameter, and it defines the total number of DMA blocks available for transmit processing.

When running HSL, this parameter should be in the range 50–70.

- For the Multiplexed WAN driver, the default is 20 per channel.
- For the Serial WAN driver, the default is 40 per port.

`RX_BLKS` Integer. This is a performance-tuning parameter, and it defines the total number of DMA blocks available for receive processing.

When running HSL, this parameter should be in the range 50–70.

- For the Multiplexed WAN driver, the default is 20 per channel.
- For the Serial WAN driver, the default is 17 per port.

`RX_HDR_SPACE`

Integer. This parameter governs how much space the WAN driver should leave in the first M_DATA block. This is useful for the upper layers to put information regarding this block.

The default is 0.

`W_SCBUS_XMIT_WIRE`

Integer. This parameter specifies which SC-bus wire will be used for transferring data from the processor to the network. This is not used for the Serial WAN driver or the WAN driver for the ARTIC 1000/2000 Series adapters. For possible values, see *Figure 7-6* on page *147* and *Figure 7-7* on page *149*.

The default is 0x40.

`W_SCBUS_RECV_WIRE`

Integer. This parameter specifies which SC-bus wire is used for transferring data from the network to the processor. This is not used for the Serial WAN driver or the WAN driver for the ARTIC 1000/2000 Series adapters. For possible values, see *Figure 7-6* on page *147* and *Figure 7-7* on page *149*.

The default is 0x41.

`W_SCBUS_FRAMING_MODE`

This parameter specifies the speed of the SC bus. See page *162* for more details. This is not used for the Serial WAN driver or the WAN driver for the ARTIC 1000/2000 Series adapters. Possible values are:

- W_SCBUS_AT_2048
- W_SCBUS_AT_4096 (Default)
- W_SCBUS_AT_8192

`W_NET_SWITCH_MODE`

This parameter specifies the operational mode of the network switch. See page *160* for more details. This is not used for the Serial WAN driver. Possible values are:

- SCBUS_MASTER (Default)
- SCBUS_ARMED_MASTER
- SCBUS_BACKED_MASTER
- SCBUS_SLAVE

`W_INTERFACE_TYPE`

This parameter specifies the operational mode of the WAN driver. See *W_SETDI_PORT — Set attributes of a physical port* on page *165* and the notes on page *172* for more details. Possible values are:

- W_E1 (Default)
- W_T1
- W_J1

BSN_FLAG This parameter applies only to the Multiplexed WAN driver. Possible values are:

- YES — Reduces the acknowledgement response time by applying the BSN and BIB values to all SUs that are waiting to be transmitted.
- NO (Default)

LOGICAL_ PORT_BASE
Unsupported.

PMC_SELECT
This parameter selects the PMCs this WAN driver will process. Possible values are:

0 Specifies that the first WAN driver loaded should attempt to own all PMCs. An error will result if no PMC is installed on the adapter. This is the default.

1 Specifies that the WAN driver should attempt to own the first PMC. If the PMC is already owned, the WAN driver will exit initialization in error and the owning WAN driver will remain. If the PMC is not installed, the WAN driver load will exit in error.

2 Specifies that the WAN driver should attempt to own the second PMC. If the PMC is already owned, the WAN driver will exit initialization in error and the owning WAN driver will remain. If the PMC is not installed, the WAN driver load will exit in error.

RX_CRC_SELECT
This parameter causes the CRC bytes of each received HDLC frame to be passed upstream in the first two bytes of RX_HDR_SPACE. They are placed only in the first M_DATA block of the received messages. Use this parameter for further data integrity tests upstream. Possible values are:

- YES
- NO (Default)

SS7_FILTER_COUNT
Integer. This parameter specifies the number of duplicate FISUs (or LSSUs) that will not be filtered out and will be passed upstream.

Assume that 20 identical, consecutive FISUs are received by the WAN driver:

- If this parameter is set to 1, then two FISUs are sent (the first FISU and one duplicate). The remaining 18 FISUs are filtered out.
- If this parameter is set to 2, then three FISUs are sent (the first FISU and two duplicates). The remaining 17 FISUs are filtered out.

The default is 0.

`W_MONITOR_MODE`

This parameter applies to the Serial and Multiplexed WAN drivers for the ARTIC 1000/2000 Series adapters.

The following table shows the possible values and their descriptions.

Table 10-1. W_MONITOR_MODE — possible values

Value	Monitor Enabled – Attenuated Note 1	Monitor Enabled – Not Attenuated Note 2	Monitor SS7 traffic Note 3	ATM mode Note 4	Timestamp Note 5	Tick event generation Note 6	PMC Timestamp Note 7
YES	X		X				
NO (Default)							
ATM_QSAAL_TIME_TICK	X			X	X	X	
ATM_CPCS_TIME_TICK	X			X	X	X	
ATM_QSAAL_TIME	X			X	X		
ATM_CPCS_TIME	X			X	X		
YES_NOATTEN		X	X				
ATM_QSAAL_TIME_TICK_NOATTEN		X		X	X	X	
ATM_CPCS_TIME_TICK_NOATTEN		X		X	X	X	
ATM_QSAAL_TIME_NOATTEN		X		X	X		
ATM_CPCS_TIME_NOATTEN		X		X	X		
PMC_TIMESTAMP	X		X				X

Notes:

1. This value puts all T1/E1/J1 ports in monitor mode where the transmitter is tri-stated and the receiver's sensitivity is increased to detect an incoming signal of –20 dB resistive attenuation.
2. If the user's equipment does not need –20 db resistive attenuation, this value loads the rpq_wanm.rel in monitor mode without resistive attenuation.
3. This value is used for monitoring SS7 traffic.
4. This value is effective only when running in ATM mode.
5. This value enables the timestamp facility to put a timestamp on the front of all data messages and send a WAN_NOTIFTIM message to the upper level whenever an event occurs that is related to:
   - Digital interfaces (WAN_NOTIFDI)
   - ATM cell stream status (WAN_NOTIF_ATM).
6. This value enables the tick event facility to send a WAN_NOTIFTIM message (WAN_TICK_EVENT) to the upper level when the timestamp crosses a 100 ms boundary. This message will be sent on every active data stream.
7. This value causes a 32-bit timestamp to be placed as the first field in every message. This timestamp is applied as messages are received by the hardware. The timestamp value will be in big endian format. The granularity of the timestamp will be as follows:
   - Serial WAN driver — 60.606 nanoseconds
   - Multiplexed WAN driver (T1/E1/J1) — 16 nanoseconds.

For more information on notes 5 and 6, see *WAN_NOTIFTIM — Send a timestamped notification* on page *134* and *W_SET_TIMESTAMP — Set timestamp* on page *183*.

W_TDM_CLOCK_RATE

This parameter represents the rate at which the TDM clock will run. It will be kept in the field wan_params.w_tdm_clock_rate. Possible values are:

4 A 4 MB clock rate (Default).

8 The 8 MB clock rate is meant to be used by special applications that require only one PMC, and which must be in slot 1. The maximum number of channels i 72.

Table 10-2. Initialization error codes

Error name	Hexadecimal Value	Description
INIT_NO_HARDWARE	0xff000001	The WAN driver did not find the correct type of mezzanine card. Either use a different driver or different hardware.
INIT_PARAM_ERROR	0xff000002	The parameters passed are not correct or are inconsistent.
INIT_ALLOC_ERROR	0xff000003	A general allocation error. Unless specified by a specific error code, this error code indicates that some resource required for operation could not be obtained.
INIT_NO_MEM	0xff000004	A general memory allocation error. Unless a specific error code, this error code indicates that some resource required for operation could not be obtained.
INIT_POD_ERROR	0xff000005	Basic diagnostics failure. For the Multiplexed WAN driver, core diagnostics failed. For the Serial WAN driver, the DMA and SCC chips could not be initialized.
INIT_PORT1_FAIL	0xff000011	Port 1 loopback failed.
INIT_PORT2_FAIL	0xff000012	Port 2 loopback failed.
INIT_PORT3_FAIL	0xff000014	Port 3 loopback failed.
INIT_PORT4_FAIL	0xff000018	Port 4 loopback failed.
INIT_PORT5_FAIL	0xff000022	Port 5 loopback failed.
INIT_PORT6_FAIL	0xff000023	Port 6 loopback failed.
INIT_PORT7_FAIL	0xff000024	Port 7 loopback failed.
INIT_PORT8_FAIL	0xff000025	Port 8 loopback failed.
INIT_PORT9_FAIL	0xff000026	Port 9 loopback failed.
INIT_PORT10_FAIL	0xff000027	Port 10 loopback failed.
INIT_PORT11_FAIL	0xff000028	Port 11 loopback failed.
INIT_PORT12_FAIL	0xff000029	Port 12 loopback failed.
INIT_PORT13_FAIL	0xff00002A	Port 13 loopback failed.
INIT_PORT14_FAIL	0xff00002B	Port 14 loopback failed.
INIT_PORT15_FAIL	0xff00002C	Port 15 loopback failed.
INIT_PORT16_FAIL	0xff00002D	Port 16 loopback failed.
Note: INIT_PORTx_FAIL can be logically ORed to signify failures on multiple ports. When any of these errors are encountered, the driver will not load.		

Initial line characteristics

The WAN drivers provide an extensive set of commands to alter various port and channel parameters for handling various line conditions. The WAN drivers program the hardware to some default value. If these defaults meet your needs, you will not need to go through the time-consuming configuration process.

Serial WAN driver in synchronous mode — defaults

The following list is for the Serial WAN driver in synchronous mode:

- HDLC framing
- For AIB, V.36 electrical Interface
- For PMC, the electrical interface depends on the cable ID
- External clocking
- 8 bits per character
- Flag idle (0x7E)
- Duplex
- CRC-CCITT (preset to 1's)
- NRZ
- No address filtering
- No timeouts
- Maximum Frame Size = 256 bytes
- Shared frame synchronization flags

Multiplexed WAN driver for any of its ports — defaults

The following list is for the Multiplexed WAN driver for any of its ports:

- For T1 operational mode, the port is programmed for T1 ESF (Extended Super Frame)
- For E1 operational mode, the port is programmed for E1 double-frame format.
- No frame CRC
- Line Coding:
  - T1 B8ZS
  - E1 HDB3
  - J1 B8ZS
- No Alarm notifications.
- All channels on a port are not chained or looped back, causing data to be lost.

Multiplexed WAN driver for any of its channels — defaults

The following are for the Multiplexed WAN driver for any of its channels:

- The protocol mode is HDLC.
- The data rate is 64 Kbps.
- Maximum frame size is 280 bytes.
- For ATM mode, the default value is 4100.

Interfacing with the WAN driver

The user of the WAN driver will need to write some programs to configure the hardware, and write the protocol drivers to handle specific protocols on the port or channel. For this purpose, the following header files are provided for programming those components in C. For details on how to develop those programs, refer to the programmer's reference for the environment in which the program executes and the applicable development environment.

The header files are organized so that only those extensions required for a particular application can be included or, if desired, all extensions can be included for handling all cases of the hardware and software options with the WAN driver. For this purpose, *Table 10-3* defines how the extensions are included in the basic include file ric_wan.h.

Table 10-3. Header file organization

File name	#define name	Description
ric_wan.h		Base header file. Define all needed extensions *before* including this file. It has statements to correctly include those extensions for you. Alternatively, include those individual files. This also sets up the most common definitions, such as mapping of uint32 to a 32-bit quantity, including ric.h. This file also includes the common command definitions.
wandefs.h		Basic #defines for common WAN support. The commands and messages in *Chapter 5, Serial and Multiplexed WAN drivers (common operations)* on page *49* will use the definitions here.
wan_prot.h		The common message definitions.
wan_cont.h		The common command definitions.
wan_ss7.h	INCLUDE_SS7	The SS7 command and message definitions.
wan_ibm.h	INCLUDE_IBM	The RadiSys command extensions and message definitions for the Serial WAN driver.
wan_mux.h	INCLUDE_MUX	The extensions to command and message definitions for the Multiplexed WAN driver operations.
wan.atm.h	INCLUDE_ATM INCLUDE_SCBUS	Adds support for ATM and SC bus.

LED use

The PMCs have a varying number of LEDs that can be used to indicate hardware status. *Table 10-4* summarizes the usage.

Table 10-4. LED usage summary

PMC	Type and Number of LEDs	State on power-up	State after WAN load
ARTIC960 4-Port Selectable PMC	Amber/Green (1)	OFF	Green, if tests during WAN load are successful; otherwise, Amber.
ARTIC960 4-Port T1/E1 Mezzanine Card	Main Amber/Green (1) and Port Green (4)	OFF	If tests during the WAN driver load are not successful, or if one or more ports have a hardware error during normal operations, the main Amber/Green LED is set to Amber; otherwise, it is set to Green. The port-specific Green LED is OFF if the cable type and the current operational mode do not match, or if an alarm is present on that port; otherwise, it is set to Green.
ARTIC 1000/2000 Series	Amber/Green (4)	OFF	The port-specific Green LED is OFF if the cable type and the current operational mode do not match, or if an alarm is present on that port; otherwise, it is set to Green.

Glossary

A

AAL	ATM Adaptation Layer — Enhances the services provided by the ATM Layer to support functions required by the next higher level.
AAL5	ATM Adaptation Layer 5 — Consists of the CP and SSCS.
AERM	Alignment Error Rate Monitor
AIB	Application Interface Boards
AIS	Alarm Indication Signal
ATM	Asynchronous Transfer Mode — Packet-oriented transfer mode that uses the asynchronous time division multiplexing technique to multiplex information flow in fixed blocks called *cells*.
AWP	Architecture Working Standard

B

BIB	Backward Indicator Bit
BISYNC	Binary Synchronous Communications
BSN	Backward Sequence Number

C

Ca	The ERM gets indications from frame processing on the occurrence of erroneous and valid SUs. It does not need to look into the SU data. Each type of ERM keeps a counter: Ca for AERM.
CAS	Channel Associated Signaling
CCS	Common Channel Signaling
cell	In ATM, a fixed-size block containing multiplexed information.
channel	For the Multiplexed WAN driver, the terms *line* or *channel* are used to refer to one of the multiplexed signals on a port (or one of the time slot).
clone device	In the UNIX file system, the system configuration process defines a wild card special file, called a *clone device*.
connection management mode	Sends standard asynchronous data frames without transparency. This mode can be used to communicate with a modem.

CP Common Part — Part of the AAL5 and consists of the SAR and CPCS.

cPCI CompactPCI† environment

CPCS Common Part Convergence Sublayer — A layer of the CP.

CRC Cyclic Redundancy Check

Cs The ERM gets indications from frame processing on the occurrence of erroneous and valid SUs. It does not need to look into the SU data. Each type of ERM keeps a counter: Cs for SUERM

CS Convergence Sublayer

CSF Current Status Field

CT Computer Telephony

D

DCD data carrier detect

DAEDR Delimitation, Alignment, Error Detection for receive

DAEDT Delimitation, Alignment, Error Detection for transmit

device special files The system configuration process defines special files called *device special files* in the UNIX file system. They usually represent a fixed profile to users.

DF Double Frame

DI Digital Interface

DL Data Link

DLE Data Link Escape

DMA Direct Memory Access

E

ECTF Enterprise Computer Telephony — Standard bus for interoperable computer telephony (CT) systems.

EIM Errored Interval Monitor

ERM Error Rate Monitor

ESF Extended Super Frame

F

FALC Recovered clocks from the communication chips

FCS Frame Check Sequence

FDL Facility Data Link

FIB Forward Indicator Bit

FISU	Fill-In Signal Unit
Flow Control Trans-parency	Transparency mode supported by the Asynchronous HDLC WAN. Allows ASCII X-ON (0x11) and X-OFF (0x13) characters to be sent transparently over the link connection.
FSN	Forward Sequence Number
Full Trans-parency	Transparency mode supported by the Asynchronous HDLC WAN. Allows all ASCII control characters to be sent transparently over the link.

H

HEC	Header Error Checksum
HDLC	High-Level Data Link Control governed by the *ISO 3309* specifications.
HSL	High-Speed Signaling Link. A term used for implementing higher speeds for signaling.

I

IAC	Initial Alignment Control
ISO	International Standards Organization
ITU	International Telecommunication Union

L

LED	Light Emitting Diode
LI	Length Indicator
line	One of the physical ports controlled by the Serial WAN driver. For the Multiplexed WAN driver the terms *line* or *channel* are used to refer to one of the multiplexed signals on a port (or one of the time slots).
LOF	Loss of Frame
long haul	Cable length is more than 200 meters
LOS	Loss of Signal
LSC	Link State Control
LSSU	Link Status Signal Unit
LSSURT	LSSU retransmission flag
LSUH	Last SU Header

M

Message Mode A mode of service defined by the AAL5.

MF Multiframe

minor numbers The system configuration process defines special files called *device special files* in the UNIX file system. They usually represent a fixed profile to users. The system configuration process assigns fixed numbers, called *minor numbers*, which are passed to the driver when the device special file is opened.

MSU Message Signal Unit

MTP1 Level 1 of the Message Transfer Part.

MTP2 Level 2 of the Message Transfer Part.

MTP3 Level 3 of the Message Transfer Part.

Multiplexed WAN Driver A WAN driver that provides access to a physical interface over which multiplexing of data as separate logical channels (or time slots) is possible (for example, T1, E1 or J1).

N

NDIS protocol mode Transparency mode wherein the asynchronous HDLC driver provides the data transparency, FCS calculation, and framing necessary to conform to the *ISO 3309* standard.

NNI Network Node Interface — A type of SSCF.

non-clone open The system configuration process defines special files called *device special files* in the UNIX file system. They usually represent a fixed profile to users. The system configuration process assigns fixed numbers, called *minor numbers*, which are passed to the driver when the device special file is opened. The process of opening such a special file is called *specific open* or *non-clone open* in this book

O

OAM Operation and Maintenance

OCM Octet Counting Mode

OSI Open System Interconnect

OSS On-card STREAMS Subsystem

P

PCI Peripheral Component Interconnect

PMC PCI Mezzanine Card

PM Physical Medium

R

RAI	Remote Alarm Indication
RC	Reception Control
RTM	Rear Transition Module

S

SAAL	Signaling ATM Adaptation Layer
SAL	STREAMS Access Library
SAP	Service Access Points
SAR	Segmentation and Reassembly — A layer of the CP.
SCB	subsystem control block
SDU	Service Data Unit
Serial WAN driver	A WAN driver that provides access to a physical interface capable of serial communications over which multiplexing of data is not possible or available (for example, 56-Kbps leased line).
SF	**1:** Status Field; **2:** Super Frame (T1 interface)
short haul	Cable length is less than 200 meters
SIB	*Busy* LSSU
SIE	Status Indicator Emergency
SIF	Signaling Information Field
SIO	**1:** Status Indicator Out of Alignment (a type of LSSU); **2:** Service Information Octet
SIOS	Status Indicator Out of Service
SNMP	Simple network management protocol
specific open	The system configuration process defines special files called *device special files* in the UNIX file system. They usually represent a fixed profile to users. The system configuration process assigns fixed numbers, called *minor numbers*, which are passed to the driver when the device special file is opened. The process of opening such a special file is called *specific open* or *non-clone open* in this book.
SSCF	Service Specific Coordination Function — Part of the SSCS.
SSCOP	Service Specific Connection Oriented Protocol — Part of the SSCS.
SSCS	Service Specific Convergence Sublayer — Part of AAL5.
SS7	Signaling System Number 7 — A dedicated digital network to perform call control.
Streaming Mode	A mode of service defined by the AAL5.
SU	Signal Unit or a Frame

sub-networked mode Also referred to as the multiplexed mode.

SUH SU Header

SUERM Signal Unit Error Rate Monitor

SUH SU Header. All MTP2 frames start with the BSN, BIB, FSN and FIB. This group of fields is called the SUH.

T

TC Transmission Control — SS7 mode

TC Transmission Convergence — ATM mode

TDM Time Division Multiplexed data bus

trans-parency mode See NDIS protocol mode.

U

UNI User-to-Network Interface — A type of SSCF.

V

VC Virtual Channel

VPI Virtual Path Identifier — Field used to label ATM cells.

VCI Virtual Channel Identifier — Field used to label ATM cells.

W

WAN driver Wide Area Network Device driver

Index

A

AAL5 (ATM Adaptation Layer 5)
  definition *30*
  protocol reference model *27*
  See also ATM (Asynchronous Transfer Mode)
aborted frame *7*
AERM (Alignment Error Rate Monitor) *17*
AIBs (Application Interface Boards) *4*
alarm conditions *22*
align signalling link *17*
alignment loss *12*
ARTIC 1000/2000 Series WAN driver *xii*, *1*, *74*
ARTIC960 WAN driver *xii*, *1*, *74*
asynchronous mode *49*
Asynchronous Transfer Mode (ATM)
  See ATM (Asynchronous Transfer Mode)
ATM (Asynchronous Transfer Mode)
  command sequence *46*
  definition *27*, *29*
  events, notify *187*
  management commands
    list *185*
    W_GET_ATM *193*
    W_GET_ATM_STATS *200*
    W_GET_VCC_STATS *195*
    W_SET_ATM *190*
    W_ZERO_ATM_STATS *203*
    W_ZERO_VCC_STATS *198*
  operations *185*
  physical layer
    define parameters *190*
    description *28*
    retrieve parameters *193*
  rates for transporting ATM cells *28*
  service messages
    list *185*
    WAN_NOTIF_ATM *187*
  statistics
    clear for a physical ATM cell *203*
    clear for a virtual channel *198*
    retrieve for a physical ATM cell *200*
    retrieve for a virtual channel *195*
  virtual channel, parameters *83*
ATM Adaptation Layer 5 (AAL5)
  See AAL5 (ATM Adaptation Layer 5)

B

BIB (Backward Indicator Bit) *14*
bisynchronous mode
  command sequence *42*
  definition *5*
  receive messages types *62*
  selecting *49*
bisynchronous protocol *8*
blue alarm *23*
BSN (Backward Sequence Number) *14*
BSN_FLAG, command line parameter *238*
busy LSSU *15*

C

cable types *74*
channel ID, internal *101*
channel, configure (T1/E1) *146*
channel, Multiplexed WAN driver *xiii*
channelled mode *97*
channels, logical *115*
Clear Channel Capability *19*, *45*, *140*
clock rate *240*
clone device *33*
clone open *33*, *34*
clone stream *34*
command line parameters
  BSN_FLAG *238*
  DATA_MSG_ONLY *236*
  MAX_NON_CLONE *235*
  MAX_OPENS *235*
  ONE_DATA_MSG_ONLY *236*
  PMC_SELECT *238*
  RX_BLKS *236*
  RX_CRC_SELECT *238*
  RX_HDR_SPACE *236*
  SNID_DECODE *235*
  SNID_KEY *236*
  SS7_FILTER_COUNT *238*
  TEST_INTERFACE *236*
  TX_BLKS *236*
  W_INTERFACE_TYPE *237*
  W_MONITOR_MODE *239*

W_NET_SWITCH_MODE *237*
W_SCBUS_FRAMING_MODE *237*
W_SCBUS_RECV_WIRE *237*
W_SCBUS_XMIT_WIRE *237*
W_TDM_CLOCK_RATE *240*
command sequences
ATM mode *46*
bisynchronous mode *42*
Clear Channel Capability *45*
HDLC framing mode *40*, *44*
SS7 protocol *37*, *49*, *103*
commands, summary
ATM *185*
common to Serial and Multiplexed WAN drivers *68*
RadiSys-specific *208*
SS7 *103*
T1/E1 interface *127*
Common Part (CP), part of AAL5 See CP (Common Part)
Common Part Convergence Sublayer (CPCS), part of Common Part
See CPCS (Common Part Convergence Sublayer)
communication speeds *84*
communications support *1*
configuration parameters
logical line *83*
TTC SS7 *120*
configuration, default values
Multiplexed WAN driver (any channels) *242*
Multiplexed WAN driver (any ports) *242*
Serial WAN driver in synchronous mode *242*
configuration, executable files *233*
configure WAN driver *33*
connection management mode *56*
contiguous 1 bits *7*
control messages, exchange *56*
conventions, notational *xiii*
CP (Common Part), part of AAL5 *30*
CPCS (Common Part Convergence Sublayer) *30*, *88*
create STREAMS *34*
CSF (Current Status Field) *15*
CT bus
channelled mode *97*
connection scenarios *47*
H.100/H.110 variants *25*
implementation *25*

D

DAEDR (Delimitation, Alignment, Error Detection for receive) *12*
DAEDT (Delimitation, Alignment, Error Detection for Transmit) *10*, *14*
data link protocol *11*, *11*
data transfer state *50*
DATA_MSG_ONLY, command line parameter *236*
default values, configuration
Multiplexed WAN driver (any channels) *242*
Multiplexed WAN driver (any ports) *242*
Serial WAN driver in synchronous mode *242*
define subdevices *33*
device special files *33*
DI (Digital Interface) messages *128*
digital network *9*
disable reception/transmission *57*, *69*
disconnects *22*
DMA blocks
receive, maximum *236*
transmit, maximum *236*
driver configuration information *71*

E

ECTF (Enterprise Computer Telephony Forum *25*
electrical interfaces, supported *4*
enable reception/transmission *69*
encoding SNID *49*
ERM (Error Rate Monitor)
definition *17*
SS7 SU reception *12*
erroneous frames *12*
Error Rate Monitor (ERM)
See ERM (Error Rate Monitor
events
ATM *187*
SS7 *107*
T1/E1 *129*, *181*
exchange control messages *56*
executable files *233*
extensions to the Serial WAN driver *205*

F

FCS (Frame Check Sequence)
definition *7*
SU categories *11*, *11*
SU format requirements *12*
FIB (Forward Indicator Bit) *14*
files
clone device *33*
device special files *33*
executable, configuration *233*
header, list of *243*
wild card special file *33*
Fill In Signal Unit (FISU)
See FISU (Fill In Signal Unit)
filtering address for HDLC *231*

filtering operation, reset and start 109
filtering, SU 13
FISU (Fill In Signal Unit)
  reset 109
  SU category 11, 11
  SU filtering, SS7 13
flag value 7
Forward Indicator Bit (FIB)
  See FIB (Forward Indicator Bit)
Forward Sequence Number (FSN)
  See FSN (Forward Sequence Number)
Frame Check Sequence (FCS)
  See FCS (Frame Check Sequence)
frame, aborted 7
framing structure 7
FSN (Forward Sequence Number) 14

H

H.100/H.110, variants of CT bus 25
hardware
  information 74
  interface type 86
  interface, reset 225
  start 54
  status, LEDs 244
  supported 4
  switching support 24, 26
  test 236
HDLC (High-level Data Link Control)
  aborted frame 7
  channelled mode 97
  commands, Multiplexed WAN driver 44
  commands, Serial Synchronous WAN driver 40
  definition 5
  framing 7
  framing plus SS7 mode 49
  Serial Synchronous WAN driver 40
  statistics collected 78, 81
  SU (Signal Unit) 11, 11
header files, list of 243
High-level Data Link Control (HDLC)
  See HDLC (High-level Data Link Control)
HSL (High-speed Signaling Link) 27

I

IAC (Initial Alignment Control) 10
idle flags, transmit 57
initialization error codes 241
interface type 86
interface, T1/E1 22
internal channel
  allocate 94
  get ID 101
  release ID 102
ISO (International Standards Organization) 9
ITU (International Telecommunication Union) 9

L

LEDs 244
levels 9
line characteristics 220
line, Multiplexed WAN driver *xiii*
link alignment 17
Link Status Signal Unit (LSSU)
  See LSSU (Link Status Signal Unit)
load time options 235
logic, transmission 16
logical channels 115
logical line 83
LSC (Link State Control) 10
LSSU (Link Status Signal Unit)
  reset 109
  retransmission flag 15
  SU category 11, 11
  SU filtering, SS7 13
LSSURT (LSSU retransmission flag) 15
LSUH (Last SU Header) 15, 20

M

M_DATA block, formatted 63, 236
management commands
  definition 36
  RadiSys-specific
    W_GETLINE 220
    W_GETSIG 223
    W_RESET 225
    W_SENDBREAK 227
    W_SETLINE 209
    W_SETMODE 229
    W_SETSIG 221
    W_STATIONADDR 231
  STREAMS interface 206
  summary
    ATM 185
    common to Serial and Multiplexed WAN drivers 68
    SS7 103
    T1/E1 interface 127
  W_DI_TEST_CFG 137, 185
  W_DISABLE 69
  W_ENABLE 69
  W_GET_ATM 193
  W_GET_ATM_STATS 200
  W_GET_PHY_PIPE 144

W_GET_SNID 101
W_GET_VCC_STATS 195
W_GETCH_MAP 156
W_GETDI 164
W_GETDI_PORT 174
W_GETDI_STATS 176
W_GETDRVINFO 71
W_GETHWTYPE 74
W_GETSS7 115
W_GETSS7_CCC 125
W_GETSS7_JPN 120
W_GETSTATS 78
W_GETTUNE 93
W_REL_SNID 102
W_SET_ATM 190
W_SET_PHY_PIPE 140
W_SET_SNID 94, 185
W_SET_TIMESTAMP 183
W_SETCH_MAP 146
W_SETDI 158
W_SETDI_LPBK 179
W_SETDI_NOTIF 181
W_SETDI_PORT 165
W_SETSS7 112
W_SETSS7_CCC 122
W_SETSS7_JPN 117
W_SETTUNE 83, 185
W_ZERO_ATM_STATS 203
W_ZERO_VCC_STATS 198
W_ZERODI_STATS 178
W_ZEROSTATS 81
management stream 129
MAX_NON_CLONE, command line parameters 235
MAX_OPENS, command line parameter 235
message flow for WAN_CTL 60
Message Signal Unit (MSU)
See MSU (Message Signal Unit)
Message Transfer Part level 2 (MTP2)
See MTP2 (Message Transfer Part level 2)
messages, DI 128
minor numbers 33
modem signals, down 58
monitor mode 239
MSU (Message Signal Unit)
SU category 11
transmission logic 16
MTP2 (Message Transfer Part level 2)
definition 11
SS7 protocol 9
Multiplexed WAN driver
defaults 242
definition xiii

N

Network Node Interface (NNI) 27
network switch 237
non-clone open 33, 34, 235
notational conventions xiii

O

OAM (Operation and Maintenance)
purpose 31
rx_nflow 196
WAN_options parameter 88
OCM (Octet Counting Mode) 12
ONE_DATA_MSG_ONLY, command line parameter 236
Operation and Maintenance (OAM)
See OAM (Operation and Maintenance)
operational mode 237
OSI (Open System Interconnect) 9

P

PMC_SELECT, command line parameter 238
PMCs (PCI Mezzanine Cards) 4, 238
protocol layer processing 1
protocols, supported 4

R

RadiSys-specific commands for the Serial WAN driver
W_GETLINE 220
W_GETSIG 223
W_RESET 225
W_SENDBREAK 227
W_SETLINE 209
W_SETMODE 229
W_SETSIG 221
W_STATIONADDR 231
rates for transporting ATM cells 28
RC (Reception Control) 10
red alarm 23
reset filtering operation, SS7 14, 109
reset hardware interface 225
ric_ess.rel, executable file 234
ric_kern.rel, executable file 233
ric_mcio.rel, executable file 233
ric_oss.rel, executable file 233
ric_pci.rel, executable file 233
ric_scb.rel, executable file 233
ric_skrn.rel, executable file 234
ric_wan.h, header file 243
rpq_cxb.rel, executable file 234
rpq_skrn.rel, executable file 234
run-time support 233

RX_BLKS, command line parameter 236
RX_CRC_SELECT, command line parameter 238
RX_HDR_SPACE, command line parameter 236

S

SC bus
- channelled mode 97
- connection scenarios 47
- connector 24
- implementation 24
- speed 237
- W_SCBUS_FRAMING_MODE, command line parameter 237
- W_SCBUS_RECV_WIRE, command line parameter 237
- W_SCBUS_XMIT_WIRE, command line parameter 237

SCB protocol handler subsystem 233

Serial WAN Driver
- definition *xiii*

Serial WAN driver
- break character, send 227
- control signals 223
- defaults 242
- filtering address for HDLC 231
- hardware interface, reset 225
- interface 206
- line characteristics 209, 220
- management commands 67
- management commands, RadiSys-specific
  - list 208
  - W_GETLINE 220
  - W_GETSIG 223
  - W_RESET 225
  - W_SENDBREAK 227
  - W_SETLINE 209
  - W_SETMODE 229
  - W_SETSIG 221
  - W_STATIONADDR 231
- management commands, STREAMS interface 206
- port, operating mode 229
- service messages 206
- stream, create 205
- WAN_NOTIFY, service message 207

service messages
- ATM, summary 185
- classifications 50
- common to Serial and Multiplexed WAN drivers, summary 50
- definition 36
- SS7, summary 103
- STREAMS interface 206
- T1/E1, summary 128
- WAN_ACTSS7 105
- WAN_CTL 56
- WAN_NOTIF_ATM 187
- WAN_NOTIFDI 129
- WAN_NOTIFSS7 107
- WAN_NOTIFTIM 134
- WAN_NOTIFY, for Serial WAN driver 207
- WAN_REG 54
- WAN_RESETSS7 109
- WAN_SID 51

Service Specific Connection Oriented Protocol (SSCOP), part of SSCS
- *See* SSCOP (Service Specific Connection Oriented Protocol)

Service Specific Convergence Sublayer (SSCS)
- *See* SSCS (Service Specific Convergence Sublayer)

Service Specific Coordination Function (SSCF), part of SSCS
- *See* SSCF (Service Specific Coordination Function)

Signal Unit (SU)
- *See* SU (Signal Unit)

Signalling System Number 7 (SS7)
- *See* SS7 (Signalling System Number 7)

signals, control 60

SNID_DECODE, command line parameter 235

SNID_KEY, command line parameter 236

specific minor number open 33

specific open 33

speed, communications 84

SS7 (Signalling System Number 7)
- channels, logical 115
- Clear Channel Capability 122, 125
- command sequences, Multiplexed WAN driver 44
- command sequences, Serial Synchronous WAN driver 37
- definition 5
- filtering, reset and start 109
- line, configure 112
- low-level processing 9
- management commands
  - W_GETSS7 115
  - W_GETSS7_CCC 125
  - W_SETSS7 112
  - W_SETSS7_CCC 122
- relationship with HDLC modes 103
- select 105
- service messages
  - WAN_ACTSS7 105
  - WAN_NOTIFSS7 107

WAN_RESETSS7 109
speed, achieving higher 27
status 107
SU format requirements 5
TTC SS7 (Japan version) 11
wan_ss7.h, header file 243
SS7_FILTER_COUNT, command line parameter 238
SSCF (Service Specific Coordination Function) 30
SSCOP (Service Specific Connection Oriented Protocol) 30
SSCS (Service Specific Convergence Sublayer) 30
standards 22
statistics
port, T1/E1 176
read 78
reset 81
status
hardware, LEDs 244
SS7 107
stream, clone/non-clone 34
STREAMS
calls 37
commands 36
create 34
environment emulation 1
interface 206
management commands for ATM 185
management commands for SS7 111
messages 36
service messages, T1/E1 128
STREAMS Access Library (SAL) 1
streams, maximum opened 235
SU (Signal Unit)
categories
FISU 11
LSSU 11
MSU 11
filtering 13
HDLC frame 11
transmission 14
sub-network ID 51
SUERM (Signal Unit Error Rate Monitor) 10, 17
SUH (SU Header) 15, 20
synchronous mode 49

T

T1/E1 interface
attributes, common 164
channel map settings 156
channel, configure 146
components, identifying 128
definition 22
events, notify 129, 181
information, retrieve 144
management commands
summary 127
W_DI_TEST_CFG 137
W_GET_PHY_PIPE 144
W_GETCH_MAP 156
W_GETDI 164
W_GETDI_PORT 174
W_GETDI_STATS 176
W_SET_PHY_PIPE 140
W_SET_TIMESTAMP 183
W_SETCH_MAP 146
W_SETDI 158
W_SETDI_LPBK 179
W_SETDI_NOTIF 181
W_SETDI_PORT 165
W_ZERODI_STATS 178
operations 127
physical stream, configure 140
ports
attributes 174
configure common 158
configure particular 165
loopback, control 179
statistics, clear 178
statistics, obtain 176
programming paths, verify 137
service messages
WAN_NOTIFDI 129
WAN_NOTIFTIM 134
STREAMS service messages 128
TC (Transmit Control) 10
TDM (Time Division Multiplexed) 24
technical support xvi
TEST_INTERFACE, command line parameter 236
tick event 134
time, set the current 183
timestamped notification, send 134
transmission logic 16
transmission, SU 14
transparency, framing 7
TTC SS7 (Japan version of SS7)
configuration parameters 117, 120
description 11
line, configure 117
management commands
W_GETSS7_JPN 120
W_SETSS7_JPN 117
TX_BLKS, command line parameter 236

W

W_DI_TEST_CFG, management command 137

W_DISABLE, management command 69
W_ENABLE, management command 69
W_GET_ATM, management command for ATM 193
W_GET_ATM_STATS, management command for ATM 200
W_GET_PHY_PIPE, management command for T1/E1 144
W_GET_SNID, management command 101
W_GET_VCC_STATS, management command for ATM 195
W_GETCH_MAP, management command for T1/E1 156
W_GETDI_PORT, management command for T1/E1 174
W_GETDI_STATS, management command for T1/E1 176
W_GETDRVINFO, management command 71
W_GETHWTYPE, management command 74
W_GETLINE, RadiSys-specific management command 220
W_GETSIG, RadiSys-specific management command 223
W_GETSS7, management command for SS7 115
W_GETSS7_CCC, management command for SS7 125
W_GETSS7_JPN, management command for TTC SS7 120
W_GETSTATS, management command 78
W_GETTUNE, management command 93
W_INTERFACE_TYPE, command line parameter 237
W_MONITOR_MODE, command line parameter 239
W_NET_SWITCH_MODE, command line parameter 237
W_REL_SNID, management command 102
W_RESET, RadiSys-specific management command 225
W_SCBUS_FRAMING MODE, command line parameter 237
W_SCBUS_RECV_WIRE, command line parameter 237
W_SCBUS_XMIT_WIRE, command line parameter 237
W_SENDBREAK, RadiSys-specific management command 227
W_SET_ATM, management command for ATM 190
W_SET_PHY_PIPE, management command for T1/E1 140
W_SET_SNID, management command 94
W_SET_TIMESTAMP, management command 183
W_SETCH_MAP, management command for T1/E1 146
W_SETDI, management command for T1/E1 158
W_SETDI_LPBK, management command for T1/E1 179
W_SETDI_NOTIF, management command for T1/E1 181
W_SETDI_PORT, management command forT1/E1 165
W_SETLINE, RadiSys-specific management command 209
W_SETMODE, RadiSys-specific management command 229
W_SETSIG, RadiSys-specific management command 221
W_SETSS7, management command for SS7 112
W_SETSS7_CCC, management command for SS7 122
W_SETSS7_JPN, management command for TTC SS7 117
W_SETTUNE, management command 83
W_STATIONADDR, RadiSys-specific management command 231
W_TDM_CLOCK_RATE, command line parameter 240
W_ZERO_ATM_STATS, management command for ATM 203
W_ZERO_VCC_STATS, management command for ATM 198
W_ZERODI_STATS, management command for T1/E1 178
W_ZEROSTATS, management command 81
WAN driver
- ARTIC 1000/2000 Series, definition xii
- ARTIC960, definition xii
- configuring 33
- DMA blocks, maximum 236
- hardware test 236
- header files 243
- loading 233
- M_DATA block 236
- network switch 237
- non-clone open 235
- operational mode 237
- overview 1
- PMCs selection, processing 238
- SC bus 237
- STREAMS messages and commands 36
- streams, maximum opened 235
- WAN_DAT, uses 236

WAN_ACTSS7, service messages for SS7 105
wan_atm.h, header file 243
wan_cont.h, header file 243

WAN_CTL, service message *56*
wan_ibm.h, header file *243*
wan_mux.h, header file *243*
WAN_NOTIF_ATM, service message *187*
WAN_NOTIFDI, service message for T1/E1 *129*
WAN_NOTIFSS7, service message for SS7 *107*
WAN_NOTIFTIM, service message *134*
WAN_NOTIFY, service message for Serial WAN driver *207*
wan_prot.h, header file *243*
WAN_REG, service message *54*
WAN_RESETSS7, service message for SS7 *109*
WAN_SID, service message *51*
wan_ss7.h, header file *243*
wandefs.h, header file *243*
WC_CONCNF, command *57*
WC_CONNECT, command *56*
WC_DISC, command *57*
WC_DISCCNF, command *58*
web site *xvi*
wild card special file *33*