M28945/M28946/M28947/M28950 ZipWirePlus

Programmers Reference Manual

289xx-SWG-001-B
January 2005

Revision History

Revision	Date	Description
A (500182A)	March 2002	Initial Release.
B (500182B)	October 2002	Incorporated Errata. Revised and reorganized.
A (289xx-SWG-001-A)	September 2003	Added information for M28946, M28947 and M28950 Incorporated new document number
B (289xx-SWG-001-B)	January 2005	Added Enhanced G.shdsl, IDSL NT, ATM 4-wire mode and Unframed 4-wire mode

Table of Contents

Table of Contents iii

List of Tables xvii

List of Figures xxi

1.0 Introduction 1-1

1.1 Scope 1-1
1.2 ZipWirePlus Embedded Software Features 1-2
1.3 References 1-3

2.0 System Overview 2-1

2.1 Introduction 2-1
2.2 M28945 - ZipWirePlus™ G.shdsl Transceiver with Embedded Microprocessor 2-2
2.3 M28946 - ZipWirePlus™ G.shdsl Transceiver with Dual-Bearer Technology 2-3
2.4 M28947 - Advanced ZipWirePlus™ G.shdsl Transceiver with E1 Framer 2-4
2.5 M28950 - ZipWire$^{VPG™}$ Advanced Voice-PairGain Transceiver with Embedded Microprocessor 2-4
2.6 ZipWirePlus Product Matrix 2-5

3.0 ZipWirePlus Operating Modes Detailed Description 3-1

3.1 ZipWirePlus General Multiplexing Functional Summary 3-1
3.2 PCM Interface Mode Using DSL Framer PCM Channel 3-2
3.2.1 M28945, M28946, M28947 3-2
3.2.2 M28950 3-3
3.3 PCM & Narrowband Interface Mode Using DSL Framer 3-4
3.4 DSL Auxiliary Interface Mode Using DSL Framer Auxiliary Channel. 3-4
3.4.1 M28945, M28946, M28947 3-4
3.5 ATM Interface Mode Using DSL Framer PCM Channel 3-5
3.5.1 ATM UTOPIA Interface Mode Using DSL Framer PCM Channel 3-5
3.5.2 ATM UTOPIA Interface Mode Using DSL Framer Auxiliary Channel 3-6
3.6 Four-Wire Mode Using PCM Interface 3-7
3.7 Four-Wire Mode Using UTOPIA Interface 3-7

4.0 ZipWirePlus Embedded Software Overview 4-1

4.1 Embedded Software Features 4-1
4.1.1 Supported Standards 4-1
4.1.2 Full-Featured API Command Set 4-1

4.1.3 Performance Monitoring 4-1
4.1.4 Diagnostics and Test Modes 4-1
4.1.5 Pre-Activation 4-1
4.1.6 Clocking Modes 4-2
4.1.7 Multi-Pair Applications 4-2
4.1.8 Host Processor Communications 4-2
4.1.9 Operational Code Download 4-2
4.1.10 ZipWirePlus Hardware Application Interfaces 4-2
4.1.11 Tip/Ring Reversal 4-2
4.2 ZipWirePlus Embedded Software Architecture 4-2
4.3 ZipWirePlus Boot Code Overview 4-4
4.3.1 PRAM Download Protocol 4-5
4.3.2 DEVADR[2:0] and BOOT pins 4-6
4.3.3 Download Times 4-7
4.3.3.1 Host Port RAM 4-7
4.3.3.2 RS232—UIP 4-7
4.3.4 Download and Device Validation 4-7
4.3.5 Download API Commands 4-7
4.3.6 Program RAM Checksum 4-7
4.4 ZipWirePlus Operational Code Overview 4-8
4.4.1 ZipWirePlus Operational Code Architecture 4-8
4.4.2 ZipWirePlus Operational Code Program Flow 4-8
4.4.2.1 DSL Initialization 4-9
4.4.2.2 DSL Software Reset 4-9
4.4.2.3 API Manager 4-9
4.4.2.4 Bit Pump Manager 4-9
4.4.2.5 DSL Framer Manager 4-9
4.4.2.6 ATM PHY Manager 4-9
4.4.2.7 DSL Activation State Manager 4-9
4.4.3 ZipWirePlus Operational Code Activation Phases 4-9
4.4.3.1 Activation State Manager (ASM) 4-10
4.4.3.2 Pre-Activation 4-12
4.4.3.3 DSP Training 4-12
4.4.3.4 DSL Line Coding 4-12
4.4.3.5 DSL Frame Format 4-12
4.4.3.6 ATM TC PHY 4-12
4.4.3.7 Normal Operation 4-12

5.0 ZipWirePlus Clocking Architecture 5-1

5.1 Clocking Architecture Overview 5-1
5.1.1 ZipWirePlus Clocking Architecture Implementation 5-1
5.1.1.1 M28945, M28946 & M28947 5-1

5.1.1.2 M28950 5-1
5.2 HTU-C Clocking Modes 5-2
5.2.1 Plesiosynchronous Mode....5-2
5.2.2 Plesiosynchronous Mode With External Clock Reference 5-3
5.2.3 Synchronous Mode....5-3
5.2.3.1 Synchronous— Slave Transmit Data Clock 5-3
5.2.3.2 Synchronous—Master Transmit Data Clock 5-4
5.3 HTU-R Clocking—Network Reference Clock Output 5-4
5.3.1 Independent Transmit/Receive PCM clocks 5-4
5.3.2 PCM Loop Timed Clocking Mode 5-5

6.0 ZipWirePlus Application Interface Configurations 6-1

6.1 ZipWirePlus ATM Interface Configuration 6-1
6.1.1 ATM One Second Timing 6-1
6.1.2 ATM PHY API Commands....6-1
6.2 ZipWirePlus Narrowband Interface Configuration 6-2
6.2.1 Mapping into the DSL Frame....6-2
6.2.2 Narrowband API Command Set 6-3

7.0 Pre-Activation 7-1

7.1 G.hs 7-1
7.1.1 G.hs Detailed Description....7-1
7.1.1.1 G.hs Session Manager 7-2
7.1.1.2 Line Probing 7-3
7.1.1.3 Power Back-Off (PBO)....7-3
7.1.2 Remote Mode Configuration....7-3
7.1.2.1 Capabilities List vs. Mode Select Message....7-4
7.1.2.2 Mode Select–Which Modem is in Control 7-4
7.1.2.3 Determining the Final Data Rate....7-5
7.1.2.4 TPS-TC Clocking Mode Configuration....7-7
7.1.2.5 TPS-TC Mode Configuration....7-7
7.1.3 G.hs API Commands Summary....7-8
7.2 Auto Baud 7-8
7.2.1 Message Sequencing 7-8
7.2.2 Message Format and Descriptions 7-9
7.2.2.1 Bit Order Transmission 7-9
7.2.2.2 Turn Around Time....7-10
7.2.2.3 Message Time 7-10
7.2.2.4 Message Opcode Definitions....7-10
7.2.2.5 Unlock (Key) Phase 7-10
7.2.2.6 Discovery Phase (HTU-C and HTU-R) 7-10
7.2.2.7 Probe Phase....7-11
7.2.2.8 Select Data Rate Phase 7-12

7.2.2.9 NAK Portion 7-12
7.2.3 Line Probe Protocol 7-12
7.2.4 Remote Configuration....7-13
7.2.4.1 Layer 1/2 Information 7-13
7.2.4.2 DSL Data Rate....7-13
7.2.5 Auto Baud API Command Summary 7-14
7.3 HDSL2 (OPTIS) Pre-Activation 7-14

8.0 Performance Monitoring....8-1

8.1 Overhead Processing Detailed Description 8-1
8.1.1 Sync Word and Stuff Bits....8-1
8.1.2 CRC....8-2
8.1.2.1 CRC Tap....8-2
8.1.3 Indicator Bits....8-2
8.1.3.1 G.shdsl Indicator Bit Definitions 8-3
8.1.3.2 HDSL2 Indicator Bit Definitions....8-3
8.1.3.3 IDSL Indicator Bit Definitions 8-3
8.1.3.4 HDSL1 Indicator Bit Definitions....8-4
8.1.3.5 Framer Transparent Indicator Bit Definitions 8-4
8.1.4 System And Performance Monitoring API Commands 8-5
8.2 Embedded Operation Channel (EOC) 8-5
8.2.1 Feature Overview 8-5
8.2.2 EOC General Overview 8-5
8.2.2.1 EOC Frame Format 8-6
8.2.2.2 EOC Unit Addresses....8-6
8.2.2.3 EOC Message IDs 8-7
8.2.2.4 EOC G.shdsl, HDSL1, and HDSL2 Support 8-7
8.2.2.5 EOC 4-Wire Support....8-7
8.2.3 EOC Implementation Details 8-7
8.2.3.1 EOC Transmit....8-7
8.2.3.2 EOC Receive....8-8
8.2.3.3 EOC - Host Interaction....8-9
8.2.4 EOC API Command Summary 8-10

9.0 Internal E1 Framer....9-1

9.1 Overview....9-1
9.1.1 Distinguishing Features 9-1
9.1.2 E1 Framer Overview....9-2
9.1.2.1 Multiframe Alignment 9-2
9.1.2.2 Transmit Direction....9-3
9.1.2.3 Receive Direction....9-3
9.1.2.4 Loss of Multiframe Alignment....9-3
9.1.3 CRC-4 Processing....9-4

9.1.4 CRC-4 Error Indication Bit (E-bit) Processing 9-4
9.1.4.1 CRC-4 Error Bit (E-bit) Generator 9-4
9.1.4.2 CRC-4 Error Bit (E-bit) Monitor 9-4
9.1.5 Remote Alarm Bit (A-bit) Processing 9-5
9.1.6 Spare Bit (Sa4 to Sa8) Processing 9-5
9.1.6.1 Sa4, Sa5, Sa7, Sa8 Processing 9-5
9.1.6.2 Sa6 Processing 9-5
9.1.7 Inject CRC-4 Errors 9-5
9.1.8 TS16 Channel Associated Signaling (CAS) Multiframe Alignment 9-5
9.1.9 Detection and Monitoring of Alarm Indication Signal (AIS) 9-7
9.1.10 E1 Framer Alarms 9-7
9.1.10.1 RLOF Alarm 9-7
9.1.10.2 AIS Alarm 9-7
9.1.10.3 Y Alarm 9-7
9.1.10.4 RAI Alarm 9-7
9.1.10.5 LMFA Alarm 9-7
9.1.10.6 TS16AIS Alarm 9-7
9.2 API Commands 9-8
9.3 ZipWirePlus E1 Framer Implementation Details 9-8
9.3.1 E1 Framer and Host Port DPRAM 9-8
9.4 Operational Behavior 9-13
9.4.1 Manual Mode 9-13
9.4.2 Error Gating 9-13
9.4.3 E1 Framer Error Counters 9-13
9.4.4 AUXP 9-14
9.4.5 Framed AIS 9-14
9.4.6 Frame Alignment 9-14
9.4.7 Sa6 Processing 9-14
9.5 Test Setup 9-14

10.0 HDSL1 Feature 10-1

10.1 Overview 10-1
10.1.1 HDSL1 Frame Structure 10-1
10.1.2 Features 10-2
10.1.3 Limitations 10-2
10.2 HDSL 1 Overhead bits Processing. 10-3
10.2.1.1 Indicator bits 10-3
10.2.1.2 Z-bits 10-3
10.2.1.3 EOC bits 10-3
10.2.2 Accessing the Rx Overhead bits 10-3
10.2.2.1 Indicator bits 10-3
10.2.2.2 Z-bits 10-3

10.2.2.3 EOC bits 10-4
10.3 HDSL1 APIs 10-4

11.0 IDSL NT Feature 11-1

11.1 Introduction 11-1
11.2 Overview 11-1
11.2.1 IDSL Features 11-1
11.2.2 References 11-1
11.3 IDSL Overhead Bits 11-2
11.3.1 Setting the Transmit Overhead Bits 11-2
11.3.1.1 Indicator Bits 11-2
11.3.1.2 EOC bits 11-2
11.3.2 Accessing the Receive Overhead Bits 11-2
11.3.2.1 Indicator Bits 11-2
11.3.2.2 EOC bits 11-2
11.4 IDSL APIs 11-3

12.0 Diagnostics and Test Modes 12-1

12.1 DSL Operational Error Monitoring and Handling 12-1
12.1.1 Sync Status Monitoring 12-1
12.1.2 Operational Error Status Reporting 12-1
12.1.3 DPLL Manager Error Monitoring and Handling 12-1
12.1.4 Stuffing Manager Error Monitoring And Handling 12-2
12.1.5 Operational Error Monitoring API Commands 12-2
12.2 Loopbacks 12-3
12.2.1 Loopback Interfaces 12-3
12.2.2 Loopback Configurations 12-3
12.2.3 Loopback API Options 12-5
12.3 ERLE 12-7
12.3.1 Test Setup 12-7
12.3.2 ERLE TEST MODES 12-7
12.3.2.1 Background Noise Test—Transmitter OFF 12-7
12.3.2.2 ERLE Test—Transmitter ON 12-7
12.3.3 ERLE API Command Summary 12-8
12.3.4 ERLE Procedure 12-8
12.4 BER Meter 12-8
12.4.1 BER Meter API Commands 12-9
12.4.1.1 PCM and NB Common BER Meter API Commands 12-10
12.4.1.2 PCM BER Meter API Commands 12-10
12.4.1.3 Narrowband BER Meter API Commands 12-11
12.4.2 BER Meter Configuration Sequence 12-11
12.4.2.1 PCM BER Meter-API Sequencing Example 12-11
12.4.2.2 NB BER Meter - API Sequencing Example 12-12

13.0 Microprocessor Communication Channel Protocol 13-1
13.1 Peer-to-Peer Communication Protocol 13-1
13.2 API Message Time-Out 13-2
13.3 API Message Structure 13-2
13.3.1 Incoming Message Structure 13-2
13.3.2 Outgoing Message Structure 13-2
13.3.3 Header Section 13-3
13.3.3.1 Header Section–Destination Field 13-3
13.3.3.2 Header Section–Opcode Field 13-3
13.3.3.3 Header Section—Reserved Byte or ACK Status 13-3
13.3.3.4 Header Section-Message Length Field 13-5
13.3.3.5 Header Section-Checksum 13-5
13.3.4 Data Section 13-5
13.3.4.1 Data Section—Data Parameter Field 13-5
13.3.4.2 Data Section-Checksum 13-5
13.4 Host Port RAM Interface Protocol 13-6
13.4.1 INTR_HOST and INTR_8051 Registers 13-7
13.4.2 Host Port Acknowledge Register 13-7
13.4.3 Host Port Status Registers 13-7
13.4.4 Host Port RAM Interface Sequence Of Events 13-8
13.4.5 Host Processor Polling Method 13-9
13.4.6 Multi-Device System 13-9
13.5 Unsolicited Interrupt 13-9
13.5.1 Unsolicited Interrupt Overview 13-9
13.5.2 Interrupt Sources 13-9
13.5.3 Events and the Service Routines for Unsolicited Interrupts 13-11
13.5.4 Protocol Event Sequence for Unsolicited Interrupts 13-11
13.5.5 Interrupt Mask/Enable Support 13-13
13.6 RS232 Serial Interface Protocol 13-13
13.6.1 Host Processor to 8051 Processor Message Structure 13-13
13.6.2 RS232 Acknowledge Message Structure 13-13
13.6.3 ZipWirePlus 8051 Processor to Host Processor Status Message Structure 13-14
13.6.4 RS232 Message Transfer Protocol 13-14

14.0 ZipWirePlus API Configuration 14-1
14.1 API Command Sequencing Overview 14-1
14.1.1 Special Sequencing Order Requirements 14-2
14.1.2 Multirate Configuration 14-3
14.2 Basic API Sequencing Application Examples 14-4
14.2.1 Minimum API Commands For End-to-End Data Transfer 14-4
14.2.2 General API Sequence 14-4
14.2.3 Deactivating the ZipWirePlus System 14-4

14.2.4 Retrain the ZipWirePlus Modems 14-5
14.2.5 Setting the ZipWirePlus System to an Out-of-Service State 14-5
14.3 Advanced API Sequencing Application Examples 14-5
14.3.1 G.shdsl EVM to Lab Module Transport 14-5
14.3.2 G.shdsl ATM Byte Alignment Application–Using UTOPIA 14-6
14.3.3 G.shdsl ATM Non-Byte Alignment Application–Using UTOPIA 14-7
14.3.4 G.shdsl ATM–Using ATM Serial Interface 14-8
14.3.5 2B1Q Mode With Auto Baud Pre-activation and ATM Enabled 14-8
14.3.5.1 RS8973 EVM Configuration 14-9
14.3.6 Framed PCM Plus Unframed Narrowband Example 14-10
14.3.7 Common Sync Application Setup 14-11
14.3.8 Custom Mapping Example 14-11
14.3.9 Internal E1 Framer Example 14-13
14.3.10 2E1 Multi-Pair Cascade Mode Example 14-14
14.3.11 Configuring for G.shdsl Unframed PCM 4-wire Application 14-20
14.3.12 Configuring for G.shdsl ATM 4-wire Application 14-26
14.3.13 Configuring for G.shdsl Proprietary Low Data Rates 14-30
14.3.14 Configuring an IDSL NT Application 14-31
14.3.15 Configuring an HDSL1 Application 14-31
14.3.16 Configuring for Enhanced G.shdsl Higher Rates 14-32
14.3.17 Configuring for Proprietary 4.6Mbps Mode 14-33

15.0 ZipWirePlus API Commands 15-1

15.1 API Commands: Quick Reference 15-1
15.2 API Command Set Documentation Convention 15-7
15.3 Download API Commands 15-8
15.3.1 Download Start (Length) 15-8
15.3.2 Download Data 15-8
15.3.3 Download End (Checksum) 15-9
15.4 System Configuration and Status API Commands 15-9
15.4.1 DSL Reset 15-9
15.4.2 DSL System Enable 15-10
15.4.3 DSL Clock Configuration 15-11
15.4.4 DSL System Configuration 15-12
15.4.5 Bit Pump Training Mode 15-13
15.4.6 Multirate Configuration 15-14
15.4.7 DSL Activation 15-16
15.4.8 DSL Force Deactivate 15-17
15.4.9 Unsolicited Interrupts Mask 15-18
15.4.10 Unsolicited Interrupts API Command Sub Mask 15-19
15.4.11 DSL Thresholds 15-21
15.4.12 DSL Status 15-21

15.4.13 Versions 15-26
15.4.14 Read DSL Control Commands 15-28
15.5 Pre-Activation Configuration and Status API Commands 15-28
15.5.1 DSL Pre-Activation Mode 15-28
15.5.2 DSL Pre-Activation (G.hs) User Information 15-30
15.5.3 Pre-Activation Data Rate List 15-30
15.5.4 Pre-Activation Get Far-End Capabilities 15-32
15.5.5 Pre-Activation Get Optimal Data Rate 15-33
15.6 ATM Interface Configuration API Commands 15-33
15.6.1 ATM PHY Interface Mode 15-33
15.6.2 ATM PHY Configuration 15-35
15.6.3 ATM PHY UTOPIA Configuration 15-36
15.6.4 ATM PHY Mode 15-37
15.7 PCM Interface Configuration API Commands 15-38
15.7.1 PCM Clock Configuration 15-38
15.7.2 DSL Framer Transmit Clock Control 15-39
15.7.3 DSL Framer — PCM Configuration 15-40
15.7.4 PCM Multiframe Length 15-41
15.7.5 Transmit PCM Frame Offset 15-42
15.7.6 Receive PCM Frame Offset 15-42
15.7.7 Multipair Configuration 15-43
15.7.8 DSL Framer Transmit PCM Path Reset 15-43
15.7.9 DSL Framer Receive PCM Path Reset 15-44
15.7.10 DSL Framer PCM DPLL Clock Generator 15-45
15.7.11 DPLL Reference Configuration 15-45
15.7.12 DSL Framer PCM Water Level 15-46
15.8 PCM Mapper Configuration and Status API Commands 15-47
15.8.1 Transmit PCM Mapper Value 15-47
15.8.2 Transmit PCM Mapper Write 15-48
15.8.3 Transmit PCM Mapper Read 15-48
15.8.4 Receive PCM Mapper Value 15-49
15.8.5 Receive PCM Mapper Write 15-50
15.8.6 Receive PCM Mapper Read 15-50
15.9 Narrowband Interface Configuration API Commands 15-51
15.9.1 Narrowband Multirate Configuration 15-51
15.9.2 Narrowband Configuration 15-53
15.9.3 Transmit Narrowband Frame Offset 15-55
15.9.4 Receive Narrowband Frame Offset 15-55
15.9.5 DSL Framer Transmit Narrowband Path Reset 15-55
15.9.6 DSL Framer Receive Narrowband Path Reset 15-56
15.9.7 DSL Framer Narrowband DPLL Clock Generator 15-57
15.9.8 DSL Framer Narrow Band Water Level 15-57

15.10 Narrowband Mapper Configuration and Status API Commands 15-58
15.10.1 Transmit Narrowband Mapper Value 15-58
15.10.2 Transmit Narrowband Mapper Write 15-59
15.10.3 Transmit Narrowband Mapper Read 15-59
15.10.4 Receive Narrowband Mapper Value 15-60
15.10.5 Receive Narrowband Mapper Write 15-61
15.10.6 Receive Narrowband Mapper Read 15-61
15.11 DSL Interface Configuration API Commands 15-62
15.11.1 Set DSL Framer State Machine 15-62
15.11.2 DSL Framer Write Indicator Bits 15-63
15.11.3 DSL Framer Enable Automatic Update of Indicator Bits 15-66
15.11.4 DSL Framer Read Indicator Bits 15-67
15.11.5 DSL Framer Write Z-Bits 15-69
15.11.6 DSL Framer Read Z-Bits 15-70
15.11.7 DSL Framer Write EOC Bits 15-71
15.11.7.1 HDSL1 15-71
15.11.7.2 IDSL NT 15-72
15.11.8 DSL Framer–DSL Configuration 15-72
15.11.9 DSL Data Rate 15-73
15.11.10 DSL Configuration Pair Identification Feature 15-74
15.11.11 DSL Framer Configuration 15-74
15.12 DSL Mapper Configuration and Status API Commands 15-75
15.12.1 Transmit DSL Mapper Value 15-75
15.12.2 Transmit DSL Mapper Write 15-76
15.12.3 Transmit DSL Mapper Read 15-76
15.12.4 Receive DSL Mapper Value 15-78
15.12.5 Receive DSL Mapper Write 15-78
15.12.6 Receive DSL Mapper Read 15-79
15.13 AFE Configuration and Status API Commands 15-80
15.13.1 Analog Front End (AFE) Configuration 15-80
15.13.2 Read AFE Setting 15-80
15.13.3 Write AFE Transmit Gain 15-81
15.14 Performance Monitoring API Commands 15-81
15.14.1 Line Attenuation 15-81
15.14.2 Noise Margin 15-82
15.14.3 Power Back-Off Result 15-82
15.14.4 Read ZipWirePlus System Performance Error Counters 15-83
15.14.5 Read ZipWirePlus DSL Performance Error Counters 15-83
15.14.6 Read ATM PHY Performance Error Counters 15-84
15.14.7 Read ATM PHY Cell Counters 15-85
15.14.8 Available Seconds and Total Seconds 15-86
15.15 EOC API Commands 15-87

15.15.1 EOC Transmit Send Message 15-87
15.15.2 Get EOC Transmit Message Status 15-88
15.15.3 Get EOC Receive Message 15-88
15.15.3.1 For G.shdsl & HDSL2 Modes 15-88
15.15.3.2 For HDSL1 Mode (M28950 device only) 15-89
15.15.3.3 For IDSL Mode (M28950 device only) 15-90
15.15.3.4 HDSL1 Status 15-91
15.15.4 Get EOC Receive Statistics 15-91
15.15.5 EOC Reset 15-92
15.16 E1 Framer Configuration and Status API Commands (Valid on M28947 device only) 15-92
15.16.1 E1 Framer Configuration 15-92
15.16.2 E1 Framer Tx Path Generator Values 15-95
15.16.3 E1 Framer Rx Path Generator Values 15-96
15.16.4 Inject E1 Framer CRC Error 15-97
15.16.5 E1 Framer Transmit Code 15-97
15.16.6 E1 Framer Tx Path Monitor Change 15-98
15.16.7 E1 Framer Rx Path Monitor Change 15-99
15.16.8 E1 Framer Tx Path Monitor Values 15-100
15.16.9 E1 Framer Rx Path Monitor Values 15-101
15.16.10 E1 Framer Error Counters 15-102
15.16.11 E1 Framer Multi-frame Alignment 15-103
15.16.12 E1 Framer Alarm Status 15-104
15.17 Regenerator APIs 15-105
15.17.1 DSL G.hs Start Regenerator Silence 15-105
15.17.2 DSL G.hs Stop Regenerator Silence 15-105
15.17.3 DSL G.hs Regenerator Rate Override 15-106
15.17.4 DSL G.hs Get Final Rate 15-106
15.17.5 DSL G.hs Regenerator Diagnostics 15-107
15.18 Loopbacks, Test Modes and Diagnostics API Commands 15-107
15.18.1 Loopbacks (1 of 3) 15-107
15.18.2 Test Modes (1 of 2) 15-109
15.18.3 Transmit Isolated Pulses Test Bit Pump Mode 15-110
15.18.4 ERLE Test Mode 15-111
15.18.5 ERLE Results 15-112
15.18.6 Stage Number 15-112
15.18.7 Clear ZipWirePlus Error Counters 15-113
15.18.8 Read ZipWirePlus Operational Error Counters 15-114
15.18.9 Read ATM PHY Operational Error Counters 15-115
15.18.10 Inject DSL CRC Error 15-116
15.18.11 Inject ATM PHY HEC Error 15-116
15.18.12 Data Bank Contents 15-117
15.18.13 DSL Framer Read Minimum/Maximum Water Levels 15-117

15.18.14 Diagnostics Get Configuration 15-118
15.18.15 Diagnostics Get Status 15-119
15.18.16 Diagnostics Get Performance 15-119
15.19 Miscellaneous API Commands 15-120
15.19.1 DSL DSP Configure 15-120
15.20 BER Meter Configuration and Status API Commands 15-121
15.20.1 PRBS Configure 15-121
15.20.2 Fill Pattern (CONST_FILL) 15-122
15.20.3 Transmit PCM BER State 15-123
15.20.4 Transmit PCM BER Meter Results 15-123
15.20.5 Receive PCM BER State 15-124
15.20.6 Receive PCM BER Meter Results 15-125
15.20.7 Transmit NB BER State 15-126
15.20.8 Transmit NB BER Meter Results 15-126
15.20.9 Receive NB BER State 15-127
15.20.10 Receive NB BER Meter Results 15-128
15.21 ZipWirePlus Register Read/Write API Commands 15-129
15.21.1 Write Register 15-129
15.21.2 Read Register 15-130
15.21.3 Write AFE Register 15-130
15.21.4 Read AFE Register 15-131

16.0 Appendix A: DSL Frame Structure 16-1

16.1 G.shdsl Frame Structure 16-1
16.1.1 Payload Block Data Structure 16-3
16.1.2 Data Interleaving in 4-Wire Mode 16-3
16.2 HDSL2 Configurations 16-4
16.2.1 HDSL2_1T1 16-5
16.3 HDSL1 Configurations 16-6
16.3.1 HDSL1_1T1 16-7
16.3.2 HDSL1 _2T1 16-8
16.3.3 HDSL1_1E1 16-8
16.3.4 HDSL1_2E1 16-9
16.3.5 HDSL1_3E1 16-9
16.3.6 HDSL1_DSL_CUSTOM 16-10
16.4 IDSL Configurations 16-10

17.0 Appendix B: G.Shdsl Transmit PSD Masks 17-1

17.1 Annex A 17-1
17.1.1 Symmetric 17-1
17.1.2 Asymmetric 17-2
17.2 Annex B 17-2
17.2.1 Symmetric 17-2

17.2.2 Asymmetric....................17-2

18.0 Appendix C: Acronyms and Abbreviations....................18-1

List of Tables

Table 2-1 Feature Summary for ZipWirePlus Devices....................2-6
Table 4-1 DEVADR Pin Definition 4-6
Table 4-2 BOOT Pin Definition 4-7
Table 4-3 Boot API Command Summary 4-7
Table 4-4 Activation Phases....................4-10
Table 4-5 Scrambler/Descrambler Taps....................4-13
Table 5-1 Clocking Modes 5-1
Table 5-2 ZipWirePlus Clocks 5-1
Table 6-1 ATM PHY API Command Summary 6-2
Table 6-2 Narrowband API Commands Summary....................6-3
Table 7-1 G.Hs Message Types....................7-2
Table 7-2 Mode Select Sender Matrix....................7-4
Table 7-3 Data Rate Source Options 7-6
Table 7-4 Date Rate I-bit Mask....................7-6
Table 7-5 G.hs TPS-TC Configuration Matrix 7-7
Table 7-6 G.hs API Commands Summary....................7-8
Table 7-7 Message Opcode Definitions....................7-10
Table 7-8 Discovery: Byte #1 Definition....................7-10
Table 7-9 Discovery: Byte #2 Definition for ATM Applications 7-11
Table 7-10 Discover: Byte #2 Definition for T1/E1 Applications....................7-11
Table 7-11 HTU-C to HTU-R Probe Request Data Field Definition 7-11
Table 7-12 Select Rate Data Field Definition 7-12
Table 7-13 NAK Data Field Definition 7-12
Table 7-14 Auto Baud API Commands Summary 7-14
Table 8-1 DSL Frame Structure Bit Allocation 8-1
Table 8-2 DSL Frame Structure Sync Word and Stuff Bits 8-2
Table 8-3 CRC Tap....................8-2
Table 8-4 G.shdsl Indicator Bit Definitions....................8-3
Table 8-5 HDSL2 Indicator Bit Definitions 8-3

Table 8-6 HDSL1 Indicator Bit Definitions- LT to NT direction 8-3
Table 8-7 HDSL1 Indicator Bit Definitions- NT to LT direction 8-4
Table 8-8 HDSL1 Indicator Bit Definitions 8-4
Table 8-9 System And Performance Monitoring API Command Summary 8-5
Table 8-10 EOC Message Types 8-6
Table 8-11 EOC Frame Format 8-6
Table 8-12 EOC Device Addresses 8-6
Table 8-13 EOC - Number of Bytes per Frame 8-7
Table 8-14 Host Processor EOC Transmit Message Events 8-9
Table 8-15 Host Processor EOC Receive Message Events 8-9
Table 8-16 EOC API Commands Summary 8-10
Table 9-1 ITU-T CEPT Multiframe Structure 9-3
Table 9-2 Automatic Mode E-bit Examples 9-4
Table 9-3 ITU CEPT Time Slot 16 Channel Associated Signaling multiframe Structure 9-6
Table 10-1 HDSL1 API Commands 10-4
Table 11-1 IDSL API Commands 11-3
Table 12-1 Operational Error Monitoring API Command Summary 12-3
Table 12-2 Loopback Modes 12-5
Table 12-3 ERLE API Command Summary 12-8
Table 12-4 Common BER Meter API Commands 12-10
Table 12-5 PCM BER Meter API Commands 12-10
Table 12-6 Narrowband BER Meter API Commands 12-11
Table 12-7 PCM BER Meter API Sequencing Example 12-12
Table 12-8 NB BER Meter API Sequencing Example 12-13
Table 13-1 Incoming Message Structure From the Host Processor 13-2
Table 13-2 Outgoing Message Structure From the 8051 Processor 13-3
Table 13-3 API Opcode Type 13-3
Table 13-4 Acknowledge Status Register (Interrupt Source Register) 13-3
Table 13-5 Acknowledge Status Codes 13-4
Table 13-6 Host Port RAM Mapping 13-6
Table 13-7 Host Port RAM Message Protocol Events 13-8
Table 13-8 STATUS_8: Unsolicited Interrupt Source Bit Definition 13-10
Table 13-9 Unsolicited Interrupt Events and Service Routines 13-11
Table 13-10 Unsolicited Interrupt Protocol Sequence for Link Up/Down Events 13-11
Table 13-11 Unsolicited Interrupt Protocol Sequence for API Commands 13-12
Table 13-12 Unsolicited Interrupt Protocol Sequence for E1 Framer API Commands 13-12

Table 13-13 Host Processor To 8051 Processor RS232 Message Structure 13-13
Table 13-14 RS232 Acknowledge Response Message Structure....13-14
Table 13-15 8051 Processor to Host Processor RS232 Message Structure 13-14
Table 13-16 Example 1—Incoming RS232 Message....13-15
Table 13-17 Outgoing RS232 Message....13-15
Table 13-18 Example 2—Incoming RS232 Message....13-15
Table 13-19 Example 2—Outgoing RS232 Message....13-15
Table 13-20 Example 3—Incoming RS232 Message....13-15
Table 13-21 Example 3—Outgoing RS232 Message....13-16
Table 14-1 Special Case API Sequencing Requirements....14-2
Table 14-2 Multirate Configuration API Commands 14-3
Table 14-3 Minimum API Sequencing Requirements 14-4
Table 14-4 G.shdsl EVM to Lab Module API Sequencing Example....14-5
Table 14-5 G.shdsl ATM Byte Alignment API Sequencing Example....14-6
Table 14-6 G.shdsl ATM Non-Byte Alignment–API Sequencing Example 14-7
Table 14-7 ATM Serial Interface Mode API Sequencing Example 14-8
Table 14-8 2B1Q Mode With Auto Baud Pre-activation Sequencing Example 14-8
Table 14-9 PCM + Narrowband API Sequencing Example 14-10
Table 14-10 Custom Mapping API Sequence Example 14-12
Table 14-11 E1 Framer Sequence Example 14-13
Table 14-12 2E1 Multi-Pair Cascade Mode Sequence Example....14-14
Table 14-13 Unframed PCM 4-wire mode with the Application being the master of the Clock 14-20
Table 14-14 Unframed PCM 4-wire mode with the Application being the Slave to the Clock....14-23
Table 14-15 G.shdsl ATM 4-wire API Configuration Sequence 14-27
Table 14-16 G.shdsl Low Data Rate Configuration Sequence....14-30
Table 14-17 Configuring an IDSL NT Application 14-31
Table 14-18 API Configuration sequence for a HDSL1 Application....14-32
Table 14-19 Enhanced G.shdsl API Configuration Sequence 14-33
Table 14-20 4.6 Mbps Configuration Sequence Example....14-34
Table 15-1 API Command Summary....15-1
Table 15-2 ZipWirePlus API Command Set Documentation Convention 15-7
Table 16-1 SHDSL Frame Structure....16-1
Table 16-2 HDSL2 Frame Structure 16-4
Table 16-3 1T1 Framing....16-6
Table 16-4 HDSL1 Frame Structure and Overhead Bit Allocation 16-6
Table 16-5 1T1 Framing....16-8

Table 16-6 2T1 Framing........16-8
Table 16-7 1E1 Framing........16-9
Table 16-8 2E1 Framing........16-9
Table 16-9 3E1 Framing........16-10

List of Figures

Figure 1-1 ZipWirePlus DSL System Software Overview 1-1
Figure 2-1 High-Level Functional Diagram....2-1
Figure 2-2 M28945 Detailed Block Diagram 2-2
Figure 2-3 M28946 Detailed Block Diagram 2-3
Figure 2-4 M28947 Detailed Block Diagram 2-4
Figure 2-5 M28950 Detailed Block Diagram 2-5
Figure 3-1 General Multiplexing Functional Block Diagram for M28945, M28946 & M28947....3-2
Figure 3-2 General Multiplexing Functional Block Diagram for M28950....3-2
Figure 3-3 PCM Interface Mode Using DSL PCM Channel–ATM Disabled....3-3
Figure 3-4 PCM Interface Mode Using DSL PCM Channel....3-3
Figure 3-5 PCM & Narrowband Interface Mode Using DSL Framer....3-4
Figure 3-6 DSL Auxiliary Interface Mode Using DSL Framer Auxiliary Channel 3-5
Figure 3-7 ATM UTOPIA Interface Mode Using DSL Framer PCM Channel....3-6
Figure 3-8 ATM UTOPIA Interface Mode Using DSL Framer Auxiliary Channel 3-6
Figure 3-9 Four-Wire Mode Using PCM Interface Block Diagram....3-7
Figure 3-10 Four-Wire Mode Using UTOPIA Interface Block Diagram 3-8
Figure 4-1 ZipWirePlus Embedded Software Architecture....4-3
Figure 4-2 ZipWirePlus Embedded Code Program Flow 4-4
Figure 4-3 ZipWirePlus PRAM Download Overview....4-5
Figure 4-4 Host and 8051 Download Protocol 4-6
Figure 4-5 ZipWirePlus Operational Code Architecture Overview 4-8
Figure 4-6 Activation State Manager (ASM Overview) 4-11
Figure 4-7 Activation Steps....4-11
Figure 5-1 Plesiosynchronous Mode Block Diagram....5-2
Figure 5-2 Plesiosynchronous mode With External Reference Clock 5-3
Figure 5-3 Synchronous—Slave Transmit Data Clock....5-4
Figure 5-4 Synchronous—Master Transmit Data Clock 5-4
Figure 5-5 Synchronous—PCM Loop-Timed Clocking Mode....5-5
Figure 6-1 PCM / Narrowband into DSL Frame Mapping Examples 6-3
Figure 7-1 G.hs Pre-Activation Overview 7-1

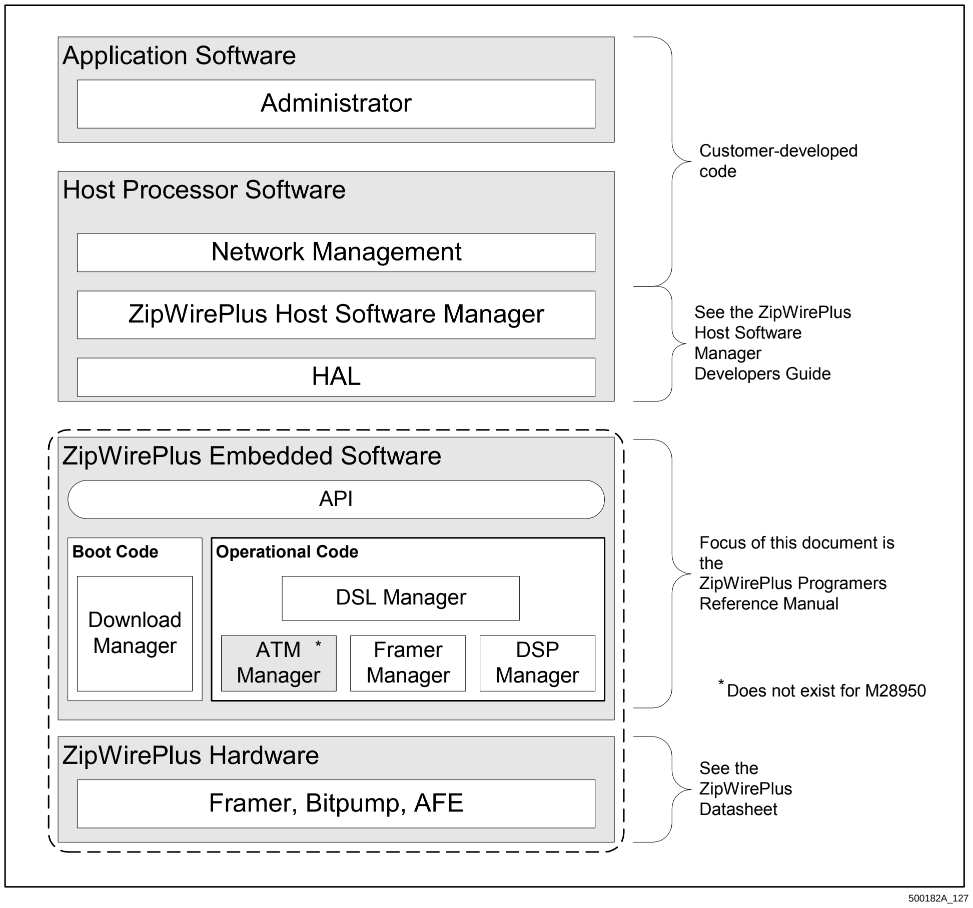
Figure 7-2 G.hs Session Overview 7-2
Figure 7-3 Power Back-off Selection....................7-3
Figure 7-4 Mode Select Sender Message Sequencing....................7-5
Figure 7-5 Auto Baud Pre-Activation Message Sequencing 7-9
Figure 7-6 Auto Baud Pre-Activation Line Probe 7-13
Figure 8-1 EOC Transmit Implementation Details 8-8
Figure 8-2 EOC Receive Implementation Details....................8-9
Figure 9-1 E1 Framer Overview 9-2
Figure 9-2 Structure for Dual Port RAM for E1 Framer 9-10
Figure 9-3 E1 Framer Test Setup....................9-15
Figure 10-1 HDSL 1 Frame Structure 10-2
Figure 12-1 DPLL State Machine 12-2
Figure 12-2 Stuffing Manager State Diagram 12-2
Figure 12-3 Loopbacks on M28945, M28946 & M28947 devices 12-4
Figure 12-4 Loopbacks on M28950 device 12-5
Figure 12-5 Detailed AFE Loopbacks....................12-6
Figure 12-6 ERLE Test Setup....................12-7
Figure 12-7 ZipWirePlus Framer PCM BER Meter 12-9
Figure 13-1 ZipWirePlus Microprocessor (8051) Communication Channels 13-1
Figure 13-2 Unsolicited Interrupt Sources 13-10
Figure 14-1 API Command Sequencing....................14-2
Figure 16-1 Structure Of Payload Blocks 16-3
Figure 16-2 Data Interleaving Within Payload Blocks....................16-4
Figure 16-3 HDSL2 Frame Structure 16-5
Figure 16-4 Payload Block Structure For 1T1 Application 16-6
Figure 16-5 HDSL1 Frame Structure 16-7
Figure 16-6 Payload Block Structure for 1T1 Application 16-8
Figure 16-7 Payload Block Structure for 2T1 Application 16-8
Figure 16-8 Payload Block Structure For 1E1 Application....................16-9
Figure 16-9 Payload Block Structure For 2E1 Application....................16-9
Figure 16-10 Payload Block Structure For 3E1 Application....................16-10
Figure 16-11 IDSL Frame Structure 16-11

1.0 Introduction

1.1 Scope

This document describes the embedded software for the ZipWirePlus family of devices. Figure 1-1 illustrates the typical software layers in a ZipWirePlus Digital Subscriber Line (DSL) solution. The embedded software consists of the boot code and the operational code. The ZipWirePlus devices support the G.shdsl, HDSL2 (OPTIS), IDSL NT, HDSL1 and SDSL standards. Figure 1-1 illustrates how the ZipWirePlus embedded software interfaces with the ZipWirePlus device (hardware) and the host processor software. The host processor software communicates to the ZipWirePlus embedded software via a set of well-defined API commands using the microprocessor communication channel.

Figure 1-1 ZipWirePlus DSL System Software Overview


Application Software
Administrator
Host Processor Software
Network Management
ZipWirePlus Host Software Manager
HAL
ZipWirePlus Embedded Software
API
Boot Code
Download Manager
Operational Code
DSL Manager
ATM Manager *
Framer Manager
DSP Manager
ZipWirePlus Hardware
Framer, Bitpump, AFE
Customer-developed code
See the ZipWirePlus Host Software Manager Developers Guide
Focus of this document is the ZipWirePlus Programers Reference Manual
* Does not exist for M28950
See the ZipWirePlus Datasheet
500182A_127

Chapter 2.0 provides an overview of the ZipWirePlus family devices. It helps the user understand the capabilities of these devices.

Chapter 3.0 describes the ZipWirePlus operational modes and the background to program different application interfaces on the device.

Chapter 4.0-Chapter 11.0 describe features of the ZipWirePlus embedded software.

Chapter 12.0 describes diagnostic and test modes.

Chapter 13.0 describes the microprocessor communication channel protocol that the external host processor uses to issue API commands.

Chapter 14.0 provides ZipWirePlus API configuration information for various applications.

Chapter 15.0 describes the ZipWirePlus API commands in detail.

Chapter 16.0 and 17.0 provide information on framing formats and Power Spectral Density (PSD) masks.

Due to the flexibility of the ZipWirePlus chipset, not all application configurations are addressed in this Reference Manual. Contact the local sales office or technical support to determine how to use the ZipWirePlus family of devices in a DSL application.

1.2 ZipWirePlus Embedded Software Features

The following software features are supported by the ZipWirePlus embedded software.

- Supported and Compliant Standards (STU-C and STU-R)
  - G.shdsl (on M28945, M28946, M28947 and M28950 devices)
  - HDSL2 (OPTIS) (on M28945, M28946, M28947 and M28950 devices)
  - 2B1Q (SDSL) (on M28945, M28946, M28947 and M28950 devices)
  - HDSL1 applications: 1T1, 1E1 and multirate (on M28950 device)
  - IDSL NT (on M28945, M28947 and M28950 devices)
- Full-Featured API Command Set
  - Application-level API commands
  - Device-level configuration and status
- Pre-Activation
  - G.hs
  - Auto Baud (2B1Q)
  - HDSL2 (OPTIS)
- Performance Monitoring
  - EOC
  - Error counters—system, performance and operational
  - Overhead (OH) bit processing
- Diagnostics and Test Modes
  - Loopbacks
  - ERLE (Echo Return Loss Enhancement)
  - BER meters
  - ZipWirePlus DSP/AFE test modes
- ATM
  - ATM-PHY TC management
  - One-second timer
  - Loopbacks, diagnostics, and status reporting
- Clocking Modes
  - Plesiosynchronous
  - Synchronous
  - Network Timing Reference (NTR)
- Multi-Pair Applications

- Four-wire mode using Framed PCM, Unframed PCM or ATM interface (on M28945, M28946 & M28947 devices)

♦ Host Processor Communications
  - Bi-directional support (unsolicited interrupts)
  - Supports both interrupt and polling-based communications

♦ Operational Code Download
  - Download via host port interface
  - Download via serial port interface (Development and debug purposes only)

♦ ZipWirePlus H/W Application Interfaces Supported
  - M28945 - PCM, Narrowband, UTOPIA (L1 and L2) and DSL auxiliary interface
  - M28946 - PCM, Narrowband, UTOPIA (L1 and L2) and DSL auxiliary interface
  - M28947 - PCM, Narrowband, UTOPIA (L1 and L2) and DSL auxiliary interface
  - M28950 – PCM interface

♦ TIP/RING reversal

1.3 References

♦ M28945 Data Sheet
♦ M28946 Data Sheet
♦ M28947 Data Sheet
♦ M28950 Data Sheet
♦ G.shdsl Standard: ITU G.991.2
♦ G.hs: ITU G.994.1
♦ HDSL2 ANSI Standard: TI/E1.4 (T1E1.4/99-006)
♦ HDSL1 ETSI Standard: ETSI TS 101 135 V1.4.1 (1998-02) –formerly ETR-152 Editions 1, 2, 3.
♦ IDSL ETSI Standard: ETSI TS 101 080 V1.3.1
♦ IDSL ANSI Standard: T1.601-1999
♦ SDSL: *RE/TM-06011-1*
♦ Auto Baud
♦ ETS 300 233
♦ CCITT G.704: General aspects of Digital Transmission Systems
♦ ETSI RTS/TMQ—06008[1]
♦ ITU-432
♦ ATM Forum Cell Based Transmission Convergence Sublayer for Clear Channel Interfaces
♦ ANSI TI.417—Spectral Management for Loop Transmission System

2.0 System Overview

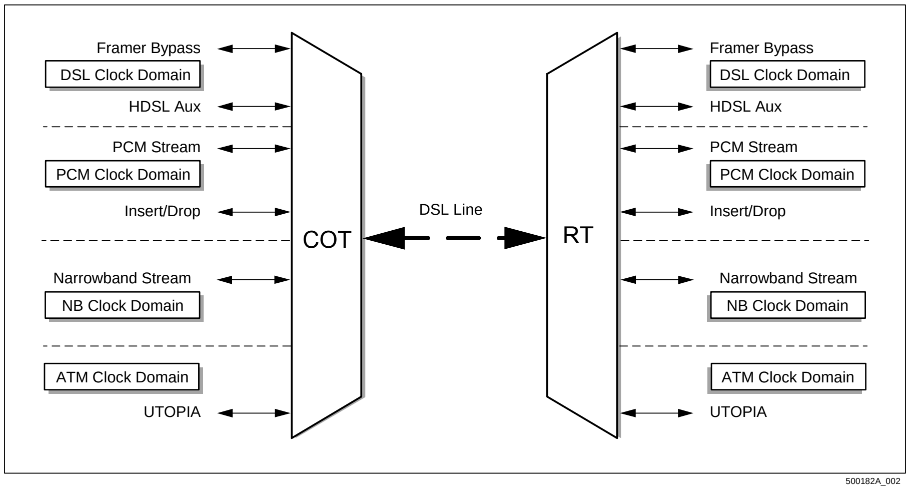
2.1 Introduction

For most applications, the ZipWirePlus device can be viewed as a pair of wires: What comes in on one terminal unit goes out the far-end terminal unit. Figure 2-1 illustrates the ZipWirePlus data interfaces. The DSL auxiliary interface operates at the DSL line rate. The PCM and Insert/Drop interfaces operate at the PCM clock rate. The Narrowband (NB) interface operates at the Narrowband clock rate. The UTOPIA interface operates at the utopia clock rate. The DSL line interfaces to the physical twisted pairs.

- The ZipWirePlus family comprises of the following devices
- M28945 - ZipWirePlus™ G.shdsl Transceiver with Embedded Microprocessor
- M28946 - ZipWirePlus™ G.shdsl Transceiver with Dual-Bearer Technology
- M28947 - Advanced ZipWirePlus™ G.shdsl Transceiver with E1 Framer
- M28950 - ZipWire$^{VPG™}$ Advanced Voice-PairGain Transceiver with Embedded Microprocessor

NOTE: Please note that M28950 does not support Narrowband and UTOPIA Interfaces

Figure 2-1 High-Level Functional Diagram



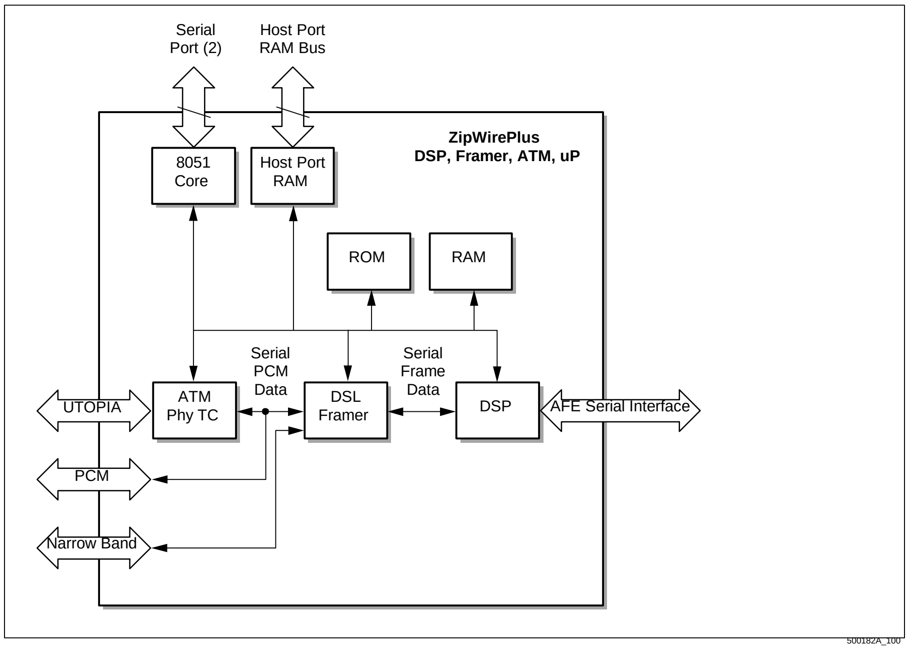
2.2 M28945 - ZipWirePlus™ G.shdsl Transceiver with Embedded Microprocessor

Figure 2-2 illustrates a detailed block diagram of the ZipWirePlus M28945 device. The ZipWirePlus embedded processor (8051) core contains an internal boot-up ROM, execution program RAM (PRAM), data storage RAM, and address decoding logic. The internal 8051 performs the transceiver startup, DSL framer overhead management, interrupt handling, etc.

A full-featured API command set allows the user to configure the ZipWirePlus system, query for status, execute loopbacks and test modes, and dictate the program flow.

The M28945 ZipWirePlus DSL solution offers two independent PCM interfaces as well as 2X operation at rates up to 4624 kbps, permitting the reliable transmission of two T1/E1 circuits over a single copper pair.

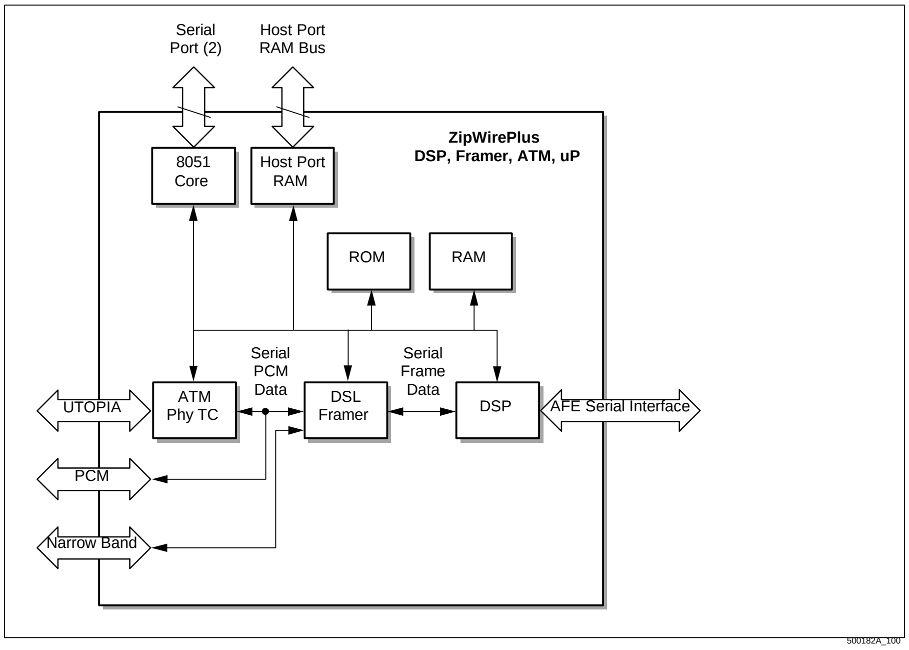
Figure 2-2 M28945 Detailed Block Diagram



2.3 M28946 - ZipWirePlus™ G.shdsl Transceiver with Dual-Bearer Technology

Figure 2-3 illustrates a detailed block diagram of the ZipWirePlus M28946 device.

Figure 2-3 M28946 Detailed Block Diagram



The M28946 ZipWirePlus Dual-Bearer DSL solution offers two independent PCM interfaces as well as 2X operation at rates up to 4624 kbps, permitting the reliable transmission of two T1/E1 circuits over a single copper pair. The M28946 is a full-function G.shdsl transceiver, fully pin-for-pin and software compatible with Mindspeed's popular M28945 Multi-Mode G.shdsl transceiver, requiring no additional hardware or software.

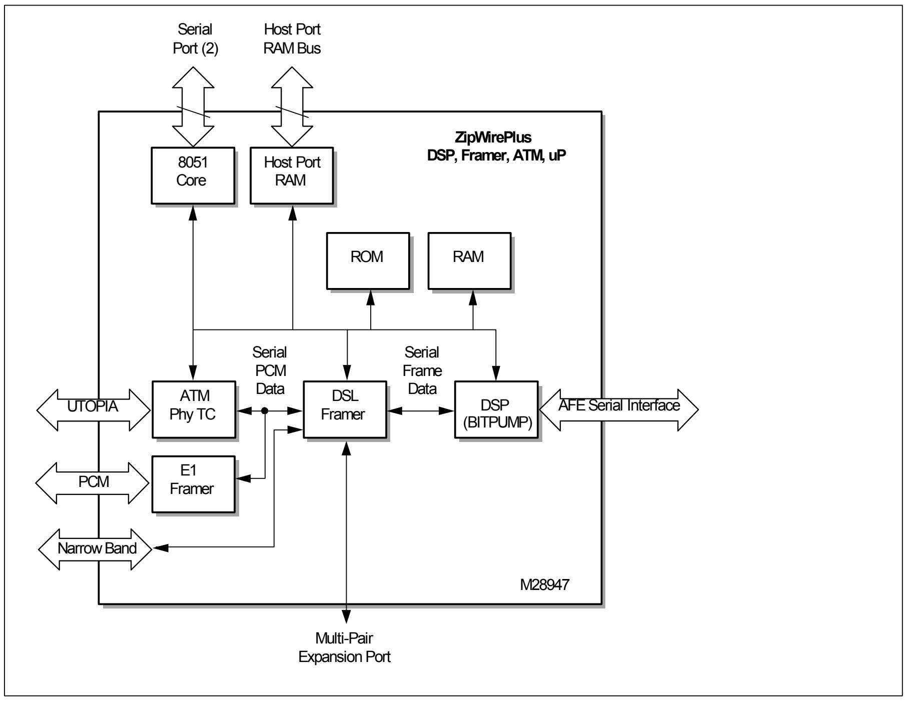
2.4 M28947 - Advanced ZipWirePlus™ G.shdsl Transceiver with E1 Framer

Figure 2-4 illustrates a detailed block diagram of the ZipWirePlus M28947 device.

The ZipWirePlus device includes E1 framing capability such that an external framer is not required to generate E1 frame synchronization. It supports all the features of the M28946 device.

The M28947 is a full-function G.shdsl transceiver, fully pin-for-pin and software compatible with Mindspeed's popular M28946 Multi-Mode G.shdsl transceiver, requiring no additional hardware or software.

Figure 2-4 M28947 Detailed Block Diagram



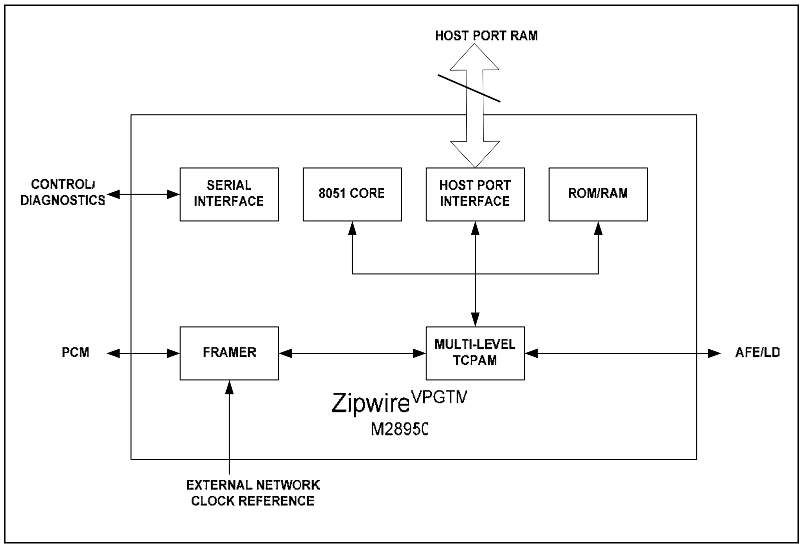
T1 framer is currently not supported.

2.5 M28950 - ZipWire$^{VPG™}$ Advanced Voice-PairGain Transceiver with Embedded Microprocessor

Figure 2-5 illustrates a detailed block diagram of the ZipWirePlus M28950 device.

The M28950 device supports G.shdsl, HDSL1 and IDSL NT operation. It provides interoperability with Mindspeed's market-leading ZipWire transceivers through operation in 2B1Q multirate mode. The 2B1Q mode includes support of AutoBaud for SDSL interoperability, rate optimization and fast connect times, as well as standards-based HDSL operation

Figure 2-5   M28950 Detailed Block Diagram



2.6 ZipWirePlus Product Matrix

The Table 2-1 summarizes the salient features of the ZipWirePlus family of devices.

Table 2-1 Feature Summary for ZipWirePlus Devices

	M28945	M28946	M28947	M28950
Description	ZipWirePlus™ G.shdsl Transceiver with Embedded Microprocessor	ZipWirePlus™ G.shdsl Transceiver with Dual-Bearer Technology	Advanced ZipWirePlus™ G.shdsl Transceiver with E1 Framer	ZipWire$^{VPG™}$ Advanced Voice-PairGain Transceiver with Embedded Microprocessor
Standards Supported	G.shdsl.bis, HDSL, IDSL, HDSL2, SDSL	G.shdsl.bis, HDSL, HDSL2, SDSL	G.shdsl.bis, HDSL, IDSL, HDSL2, SDSL	G.shdsl, HDSL, SDSL, IDSL
G.shdsl Repeater	Supported	Supported	Supported	Supported
G.shdsl 4-wire Operation	Supported	Supported	Supported	Not Supported
G.shdsl Line Probe	Supported	Supported	Supported	Supported
Dual Bearer Operation	Supported	Supported	Supported	Not Supported
Internal E1 Framer	Not Supported	Not Supported	Supported	Not Supported
Enhanced G.shdsl	Supported	Supported	Supported	Not Supported
Proprietary Lower DSL Rates (using 4 TCPAM)	Supported	Supported	Supported	Supported
Application Interfaces	PCM, Narrowband, UTOPIA L1/L2, DSL auxiliary interface, ATM Serial Interface	PCM, Narrowband, UTOPIA L1/L2, DSL auxiliary interface, ATM Serial Interface	PCM, Narrowband, UTOPIA L1/L2, DSL auxiliary interface, ATM Serial Interface	PCM interface

3.0 ZipWirePlus Operating Modes Detailed Description

This section provides a detailed description of the modes supported by the ZipWirePlus devices. The transmit data path and receive data path have independent control. This flexibility allows for a mixture of voice plus data, voice only, data only, or multiservice applications.

3.1 ZipWirePlus General Multiplexing Functional Summary

Figure 3-1 illustrates a block diagram of the general multiplexing section for the M28945, M28946 and M28947. The general multiplexing block contains all multiplexers used to select different operating modes of the device by selecting different internal connections and different I/O configurations. The application interfaces supported are the PCM Interface, the Narrowband Interface, the ATM (UTOPIA) Interface and the DSL auxiliary Interface. Internally the ZipWirePlus framer can source data from the PCM, DSL auxiliary or Narrowband channel. The ATM data can be routed internally using either the PCM or DSL auxiliary channel by appropriately selecting the multiplexers. For multi-service (ATM plus T1/E1) applications, it is recommended to route ATM data using the DSL auxiliary channel and the T1/E1 type data over PCM. The modes supported by these devices are listed below.

- PCM interface with DSL framer enabled (E1/T1 transport)
- Narrowband PCM interface with DSL framer enabled (multi-transport applications)
- DSL auxiliary interface with DSL framer enabled (non standard applications)
- ATM mode with DSL framer enabled (ATM applications)
- UTOPIA interface using DSL framer PCM channel
- ATM UTOPIA mode using DSL framer auxiliary channel (ATM applications)
- Four-wire mode using PCM interface
- Four-wire mode using ATM interface

NOTE: The narrowband PCM interface is always available. This provides a second PCM interface for multi-transport applications, such as, ISDN plus E1, V.35 plus E1, etc.

Figure 3-1 General Multiplexing Functional Block Diagram for M28945, M28946 & M28947

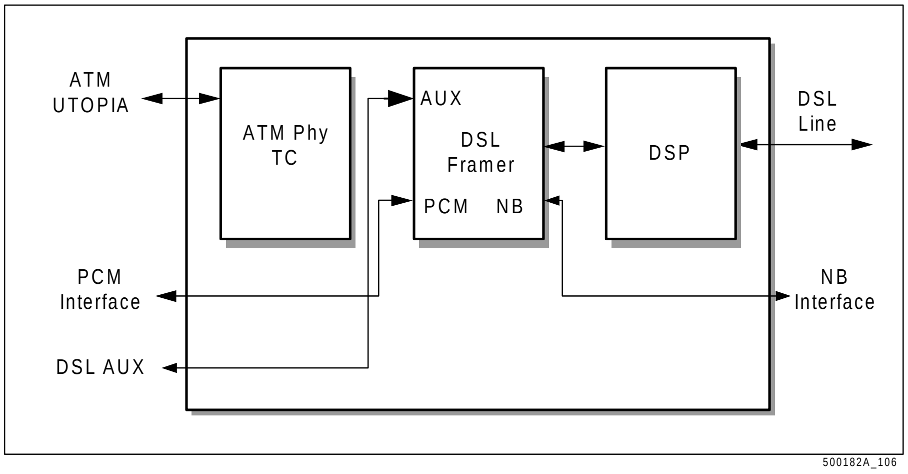
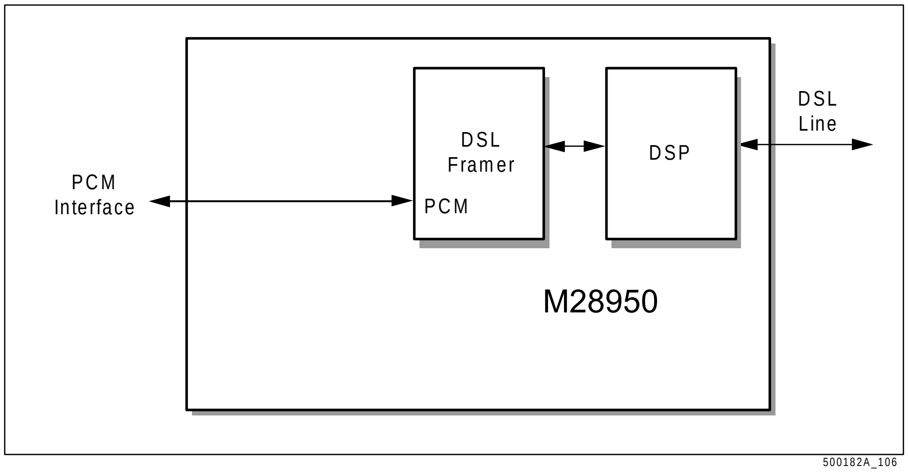


Figure 3-2 illustrates a block diagram of the general multiplexing section for the M28950. The application interface supported is the PCM Interface. Internally the M28950 framer sources the data from the PCM interface. The modes supported by M28950 are listed below.

- PCM interface with DSL framer enabled (E1/T1 transport)

Figure 3-2 General Multiplexing Functional Block Diagram for M28950



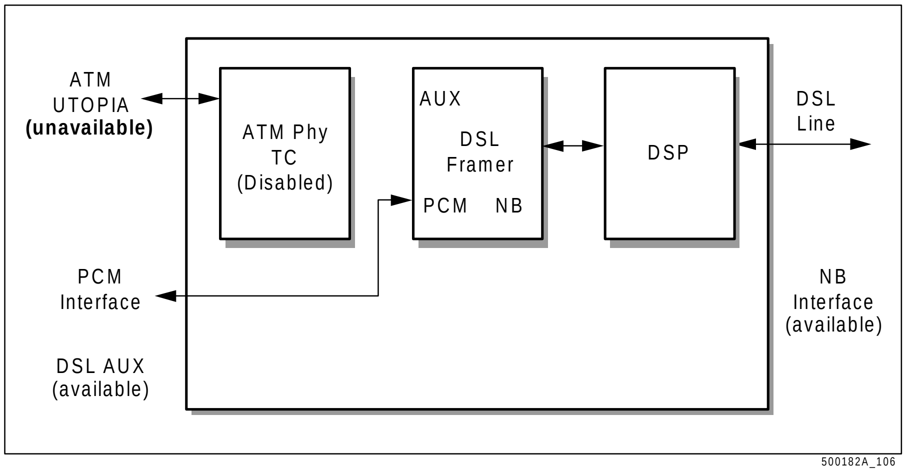
3.2 PCM Interface Mode Using DSL Framer PCM Channel

3.2.1 M28945, M28946, M28947

Figure 3-3 illustrates the PCM interface mode using the DSL framer PCM channel. This mode provides the PCM application Interface access to the DSL framer PCM interface; the ATM block is bypassed. Use the following API commands to configure the ZipWirePlus for this mode:

- Issue the DSL System Configuration API command _DSL_SYSTEM_CONFIG (0x06) with DSL configuration parameter set to 0x06 (multirate) and Frame Structure parameter to the appropriate framed option.
- Issue the Multirate Configuration API command _DSL_MULTI_RATE_CONFIG (0x1B) to configure the PCM application Interface.
- Issue the ATM PHY Interface Mode API command _ATM_PHY_IF_MODE (0x1E) with the ATM Enable parameter set to 0x00 (disabled). The remaining ATM PHY Interface Mode parameters are ignored.

Figure 3-3 PCM Interface Mode Using DSL PCM Channel–ATM Disabled

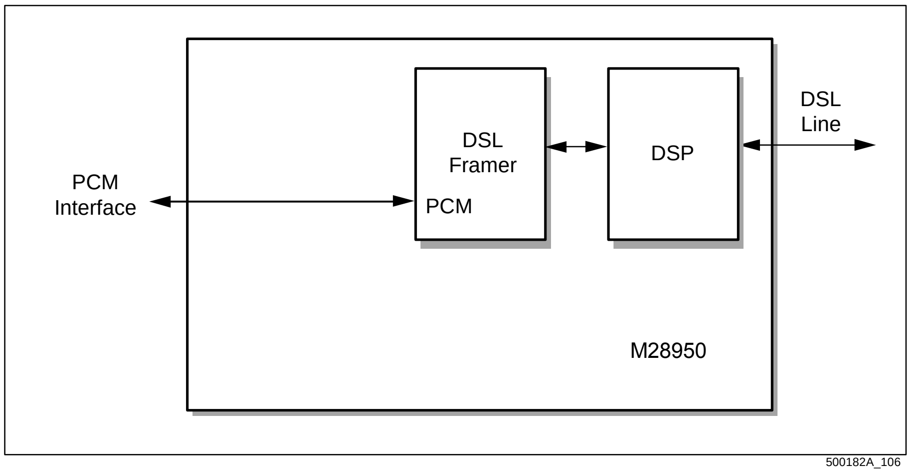


3.2.2 M28950

Figure 3-4 illustrates the PCM interface mode using the DSL framer PCM channel. This mode provides the PCM application Interface access to the DSL framer PCM interface. Use the following API commands to configure the ZipWirePlus for this mode:

- Issue the DSL System Configuration API command _DSL_SYSTEM_CONFIG (0x06) with DSL configuration parameter set to 0x06 (multirate) and Frame Structure parameter to the appropriate framed option.
- Issue the Multirate Configuration API command _DSL_MULTI_RATE_CONFIG (0x1B) to configure the PCM application Interface.

Figure 3-4 PCM Interface Mode Using DSL PCM Channel



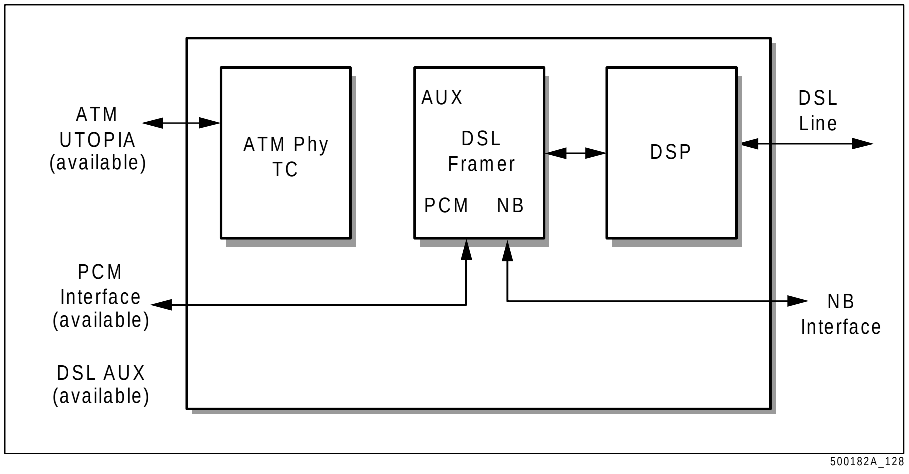
3.3 PCM & Narrowband Interface Mode Using DSL Framer.

Figure 3-5 illustrates the PCM and Narrowband (NB) interface mode using the DSL framer. This mode provides the PCM and NB Application Interfaces access to the DSL framer NB channel and DSL framer PCM channel respectively. The DSL auxiliary channel is still available for use with other ZipWirePlus application interfaces. This mode can be used for multi-transport applications such as ISDN plus E1, V.35 plus E1 etc. This mode is always available and can be used with all other modes described in this chapter. Use the following API commands to configure for this mode:

- ♦ Issue the DSL System Configuration API command _DSL_SYSTEM_CONFIG (0x06) with DSL Configuration parameter set to 0x06 (multirate) and Frame Structure parameter to the appropriate framed option.
- ♦ Issue the _DSL_FR_PCM_CONFIG (0x10) and set the PCM/NB mode to NB plus PCM mode as required by the application.
- ♦ Issue the Multirate Configuration API command _DSL_MULTI_RATE_CONFIG (0x1B) to configure the PCM application Interface.
- ♦ Set the Narrowband Multirate Configuration API command _DSL_NB_MULTI_RATE_CONFIG (0x1B) to configure the NB application Interface.

This mode is valid for M28945, M28946 and M28947 devices only.

Figure 3-5 PCM & Narrowband Interface Mode Using DSL Framer



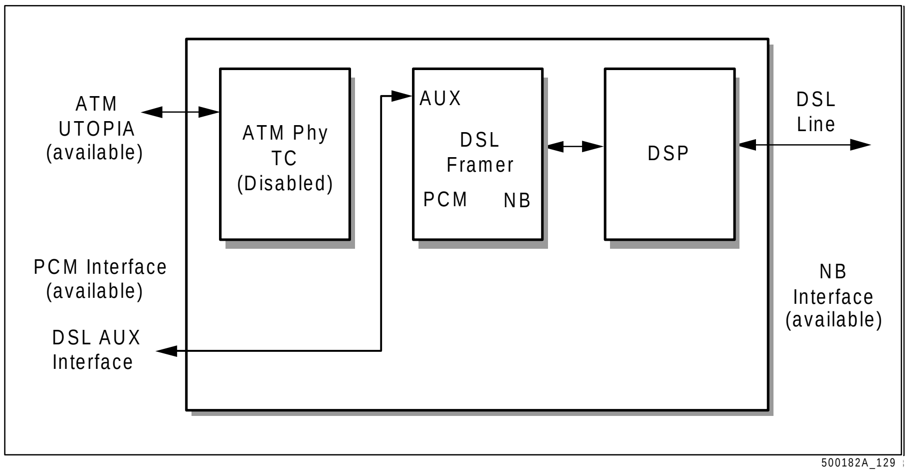
3.4 DSL Auxiliary Interface Mode Using DSL Framer Auxiliary Channel.

3.4.1 M28945, M28946, M28947

Figure 3-6 illustrates the DSL Auxiliary interface mode using the DSL framer auxiliary channel. This mode provides the DSL auxiliary application interface access to the DSL framer's auxiliary channel. The DSL framer PCM and NB channels are still available for use with other ZipWirePlus application interfaces. This is a non-standard mode and should be evaluated to meet customer-specific applications. Use the following API commands to configure for this mode:

- ♦ Issue DSL System Configuration API command _DSL_SYSTEM_CONFIG (0x06) with DSL Configuration parameter set to 0x06 (multirate) and Frame Structure parameter to the appropriate framed option.
- ♦ Issue the DSL framer configuration API command _DSL_FR_HDSL_CONFIG (0x11) with the AUX Enable parameter set.
- ♦ If the DSL framer PCM interface is being used, the PCM and DSL Mapper API commands must be programmed to allocate the desired time slots.

Figure 3-6 DSL Auxiliary Interface Mode Using DSL Framer Auxiliary Channel



3.5 ATM Interface Mode Using DSL Framer PCM Channel

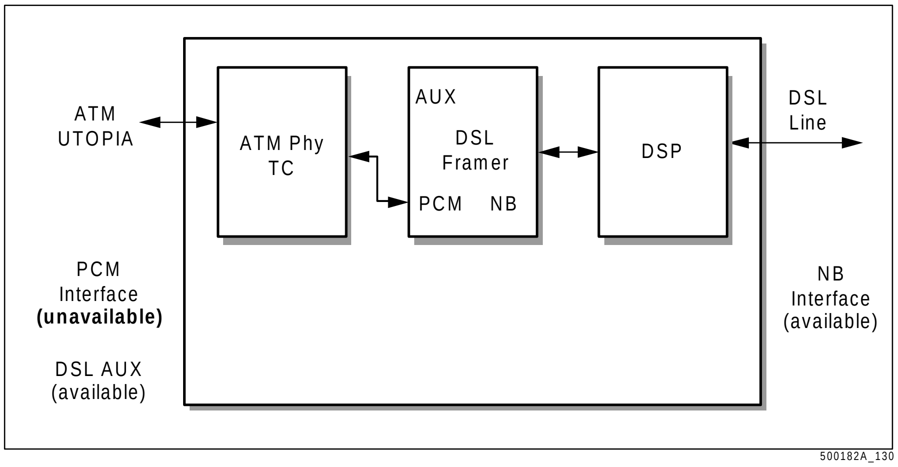
The ATM Interface mode provides the ATM UTOPIA interface or ATM SIF access to the DSL framer PCM channel. The ATM UTOPIA or ATM SIF interface is internally connected to the DSL framer PCM interface, which is then connected to the DSP interface. The DSL framer can operate in either framed or unframed mode. In framed mode, the ATM PHY operates in either T1 or E1 mode based on the DSL configuration setting. In unframed mode, the ATM PHY operates in General Purpose mode. In this mode, the PCM application Interface is unavailable. These modes are valid for M28945, M28946 and M28947 only.

3.5.1 ATM UTOPIA Interface Mode Using DSL Framer PCM Channel

Figure 3-7 illustrates the ATM UTOPIA interface mode using the DSL framer PCM channel. This mode provides the ATM UTOPIA interface access to the DSL framer PCM channel. Use the following API commands to configure the ZipWirePlus for this mode:

- Issue the DSL System Configuration API command _DSL_SYSTEM_CONFIG (0x06) with DSL Configuration parameter set to 0x06 (multirate) and Frame Structure parameter to the appropriate framed option.
- Issue the ATM PHY Interface Mode API command _ATM_PHY_IF_MODE (0x1E) with ATM Enable parameter must be enabled and the ATM Transmit/Receive Interface Mode parameter must be set based on the desired framed vs. unframed format.

Figure 3-7 ATM UTOPIA Interface Mode Using DSL Framer PCM Channel



3.5.2 ATM UTOPIA Interface Mode Using DSL Framer Auxiliary Channel

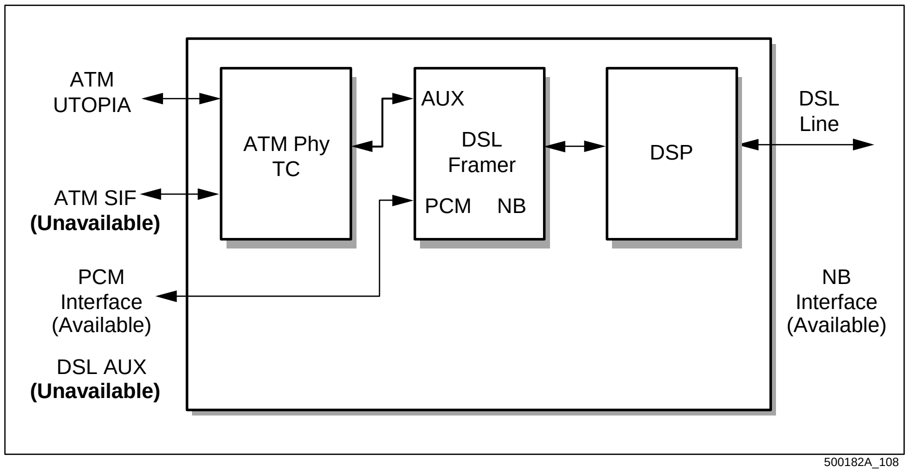
This mode provides the ATM UTOPIA interface access to the DSL framer auxiliary channel. The ATM data is internally connected to the DSL framer Aux interface which is then connected to the DSP interface. The ATM PHY operates in General Purpose mode.

This mode is valid for M28945, M28946 and M28947 devices only.

Figure 3-8 illustrates the ATM UTOPIA interface mode using the DSL framer auxiliary channel. This mode provides the ATM UTOPIA interface access to the DSL framer auxiliary channel. The PCM interface should be used to create a voice plus data application. Use the following API commands to configure the ZipWirePlus device for this mode:

- Issue DSL System Configuration API command _DSL_SYSTEM_CONFIG (0x06) with DSL Configuration parameter set to 0x06 (multirate) and Frame Structure parameter to the appropriate framed option.
- Issue the ATM PHY Interface Mode API command _ATM_PHY_IF_MODE (0x1E) with ATM Enable parameter must be enabled, and the ATM Transmit/Receive Interface Mode parameter must be set to the DSL Aux Mode option (0x01).
- Issue the DSL framer configuration API command _DSL_FR_HDSL_CONFIG (0x11) with the AUX Mode parameter set to clock-gated mode (0x01). This creates an internal gated clock between the ATM and auxiliary channel for the relevant time slots.
- If the DSL framer PCM interface is being used, the PCM and DSL Mapper API commands must be programmed to allocate the desired time slots.

Figure 3-8 ATM UTOPIA Interface Mode Using DSL Framer Auxiliary Channel



3.6 Four-Wire Mode Using PCM Interface

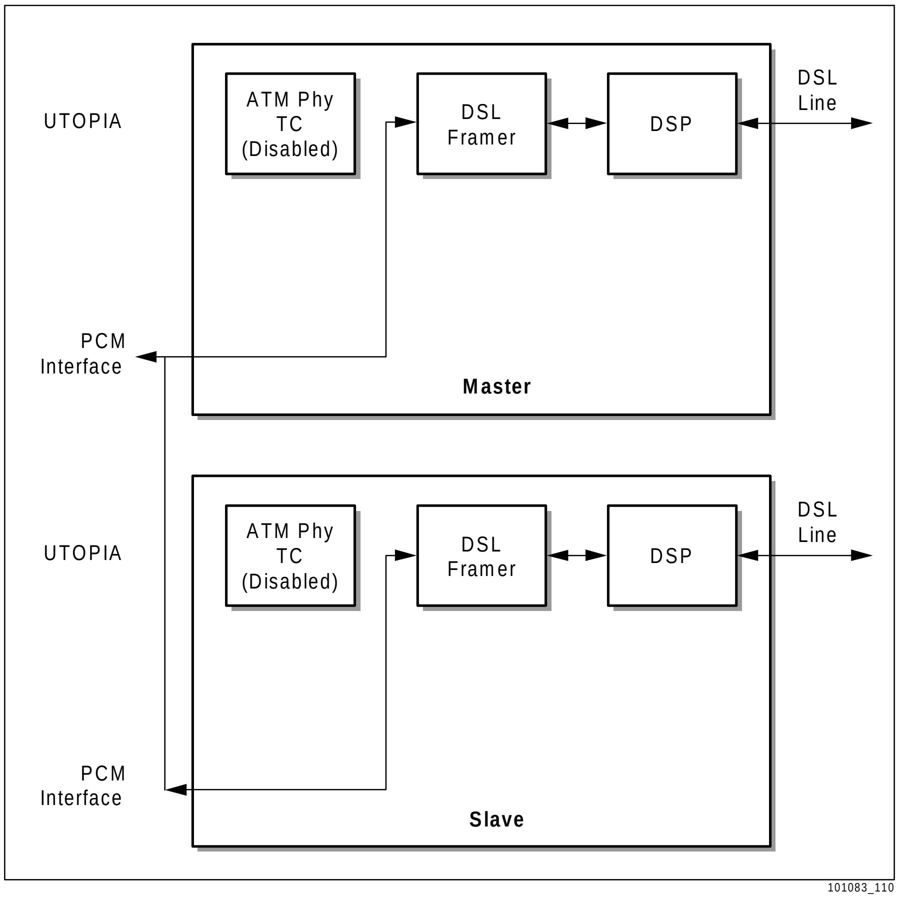
Figure 3-9 illustrates the four-wire mode using the PCM interface. This mode provides PCM data using two DSL channels (4-wire). For both the master and slave channel, the data is connected to the PCM interface and the UTOPIA interface is disabled. The DSL framers must operate in framed mode. The PCM interface can operate in either PCM Bused or PCM Cascade mode. See the ZipWirePlus data sheet for details regarding these modes.

This mode is valid for M28945, M28946 and M28947 devices only.

Use the following API commands to configure the ZipWirePlus device for this mode:

- Set DSL System Configuration API command (0x06), Frame Structure parameter to the appropriate framed option.
- The ATM PHY Interface Mode API command (0x1E), ATM Enable parameter must be disabled (0). The remaining ATM PHY Interface Mode parameters are ignored.
- Issue the _DSL_MULTI_PAIR_CONFIG (0x19) API command to select between PCM Bused mode and PCM Cascade mode. This API also configures the device to be the loop master or loop slave.

Figure 3-9 Four-Wire Mode Using PCM Interface Block Diagram



3.7 Four-Wire Mode Using UTOPIA Interface

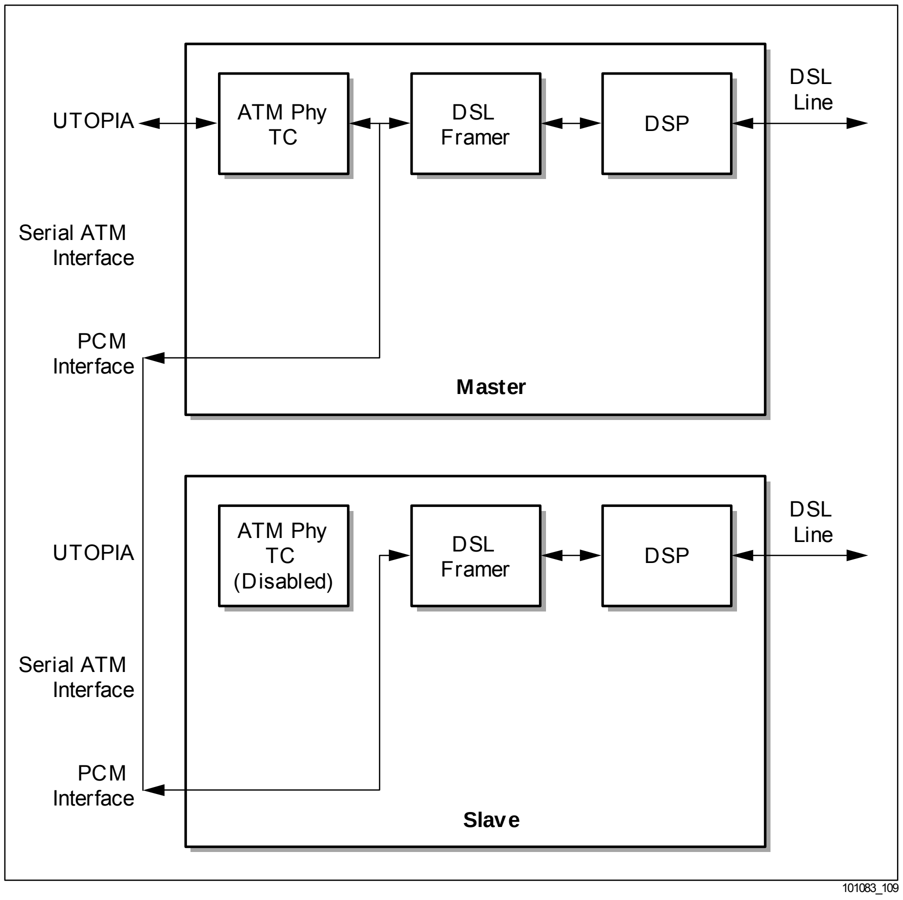
Figure 3-10 illustrates the four-wire mode using UTOPIA interface. This mode provides ATM data using two DSL channels (4-wire). The ATM data is connected to the master's channel UTOPIA interface, and the slave's UTOPIA interface is disabled. In the master channel, the ATM data is internally connected to the DSL framer PCM interface which is then connected to the DSP interface. The DSL framers must operate in framed mode.

This mode is valid for M28945, M28946 and M28947 devices only.

Use the following API commands to configure the ZipWirePlus device for this mode:

- ♦ Set DSL System Configuration API command (0x06), Frame Structure parameter to the appropriate framed option.
- ♦ The ATM PHY Interface Mode API command (0x1E), ATM Enable parameter must be enabled and the ATM Transmit/Receive Interface Mode parameter must be set based on the desired framed vs. unframed format.
- ♦ The ATM PHY Interface Mode API command (0x1E), One Second parameters must be set based on the application.

Figure 3-10 Four-Wire Mode Using UTOPIA Interface Block Diagram



4.0 ZipWirePlus Embedded Software Overview

4.1 Embedded Software Features

4.1.1 Supported Standards

The ZipWirePlus software supports G.shdsl, HDSL2 (OPTIS), 2B1Q, IDSL and HDSL1 framing formats and training modes. The software allows the ZipWirePlus devices to be configured for both CO and RT applications.

4.1.2 Full-Featured API Command Set

The ZipWirePlus implements a well-defined API command set that simplifies the host processor application interface. API's configure the ZipWirePlus hardware interfaces (PCM, Narrowband, ATM, and DSL Auxillary) as required by the application. Several low-level API's are also provided for development and debug purposes.

4.1.3 Performance Monitoring

The ZipWirePlus operational code supports overhead bit processing, and EOC performance, and system error counters that enable performance monitoring of the ZipWirePlus modem. Chapter 8.0 discusses performance monitoring in detail.

4.1.4 Diagnostics and Test Modes

The ZipWirePlus software supports several loopback configurations to help in debug and development of the ZipWirePlus modem. Echo Return Loss Enhancement (ERLE) test mode is provided to verify the AFE and hybrid circuitry. On-chip BER meters can be used to measure the bit error rate on the DSL line without the need for external BER equipment. Chapter 12.0 provides details on the supported diagnostic and test modes.

4.1.5 Pre-Activation

The ZipWirePlus operational code supports G.hs, HDSL2 (OPTIS) and 2B1Q (Auto Baud) pre-activation sequences. G.hs is specified in the *ITU G.shdsl* standards. The G.hs protocol pings to determine if the far end is present, determines line quality, and remotely configures the optimal data rate.

HDSL2 (OPTIS) pre-activation pings to determine if the far end is present and also communicates with the transmit power back-off control. The HDSL2 (OPTIS) pre-activation only supports the HDSL2 (OPTIS) mode.

The 2B1Q (Auto Baud) pre-activation is a Mindspeed solution that pings to determine if the far end is present, determines line quality, and remotely configures the optimal data rate and frame format (layer 2 information).

4.1.6 Clocking Modes

The ZipWirePlus family of devices provides a flexible clocking architecture to support applications defined by the various standards (G.shdsl, HDSL2, HDSL1, SDSL, etc). Chapter 5.0 discusses the clocking modes supported by the ZipWirePlus software.

4.1.7 Multi-Pair Applications

The ZipWirePlus software supports multipair operation. Four-wire mode using PCM and ATM interfaces are described in Section 3.6 and Section 3.7.

4.1.8 Host Processor Communications

The ZipWirePlus software allows the external host processor to communicate to the internal 8051 Microprocessor via interrupts or polling. The 8051 can communicate events like EOC, link up, link down, dying gasp, etc., via unsolicited interrupts. The microprocessor communication channel protocol is used by the host to communicate with the 8051 is described in Chapter 13.0.

4.1.9 Operational Code Download

The ZipWirePlus software allows for operational code download via the host port interface or the RS232 interface. The RS232 interface is provided for the purpose of development and debug only. Details of downloading the operational code are provided in Section 4.3.1.

4.1.10 ZipWirePlus Hardware Application Interfaces

The following application interfaces are supported by the ZipWirePlus operational code on the ZipWirePlus family of devices. Chapter 3.0 describes the various operational modes of the ZipWirePlus device. Chapter 6.0 describes the ATM and Narrowband interfaces configuration in detail.

- PCM Interface (M28945, M28946, M28947 and M28950)
- Narrowband Interface(M28945, M28946 and M28947)
- ATM–UTOPIA Interface (M28945, M28946 and M28947)
- DSL Auxiliary Interface (M28945, M28946 and M28947)

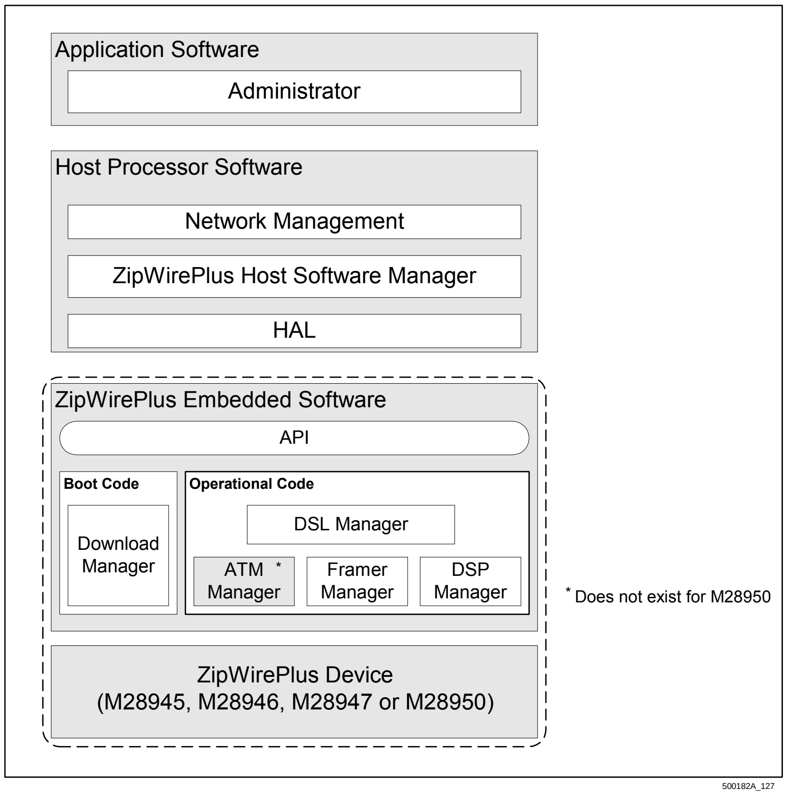
4.1.11 Tip/Ring Reversal

Tip/ring reversal is defined as the reversal of a twisted pair of wires. The ZipWirePlus device and software automatically handle any tip/ring reversal.

4.2 ZipWirePlus Embedded Software Architecture

The embedded software consists of the boot code and the operational code as illustrated in Figure 4-1. The host communicates to the ZipWirePlus device via a set of well-defined API commands. The API interface and message structure are the same for both the boot code and operational code.

Figure 4-1 ZipWirePlus Embedded Software Architecture

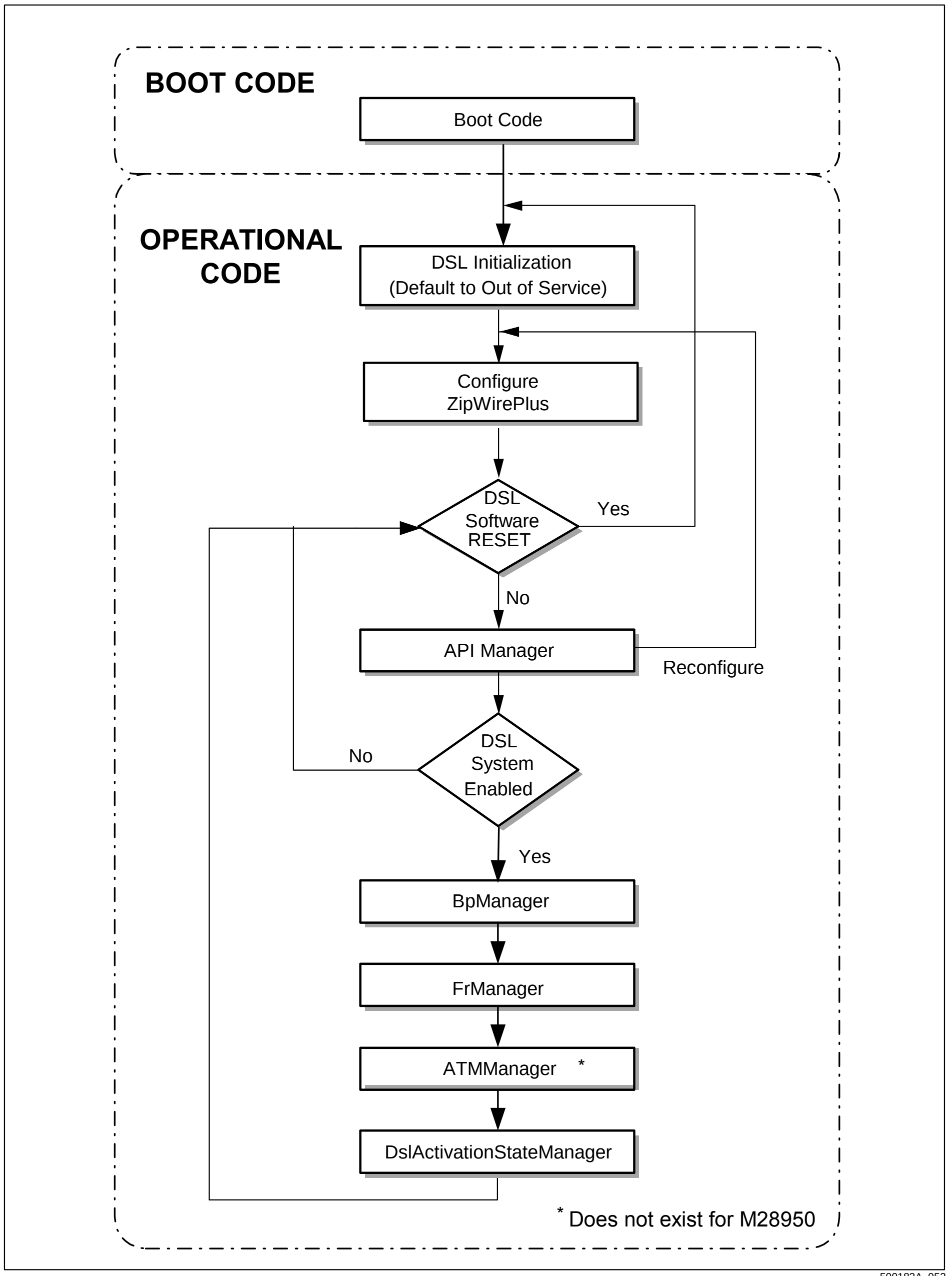


The host and ZipWirePlus embedded processor (8051) use a mailbox scheme to exchange API messages. The mailbox is implemented as a dual port shared memory. The API In Box is used by the host to send an API message to the ZipWirePlus embedded processor.

The API Out Box is used to send an API message from the ZipWirePlus embedded processor to the host processor. The microprocessor communicator channel protocol specifies the rules used by the ZipWirePlus and host processors to issue API commands. The API message structure and the protocol used for communication are described in Chapter 13.0.

Figure 4-2 illustrates the ZipWirePlus embedded software program flow. At power-on reset, the ZipWirePlus embedded processor executes the boot code. The boot code is then used to download the operational code to the ZipWirePlus internal program memory (PRAM). The program flow control is then transferred to the operational code.

Figure 4-2 ZipWirePlus Embedded Code Program Flow



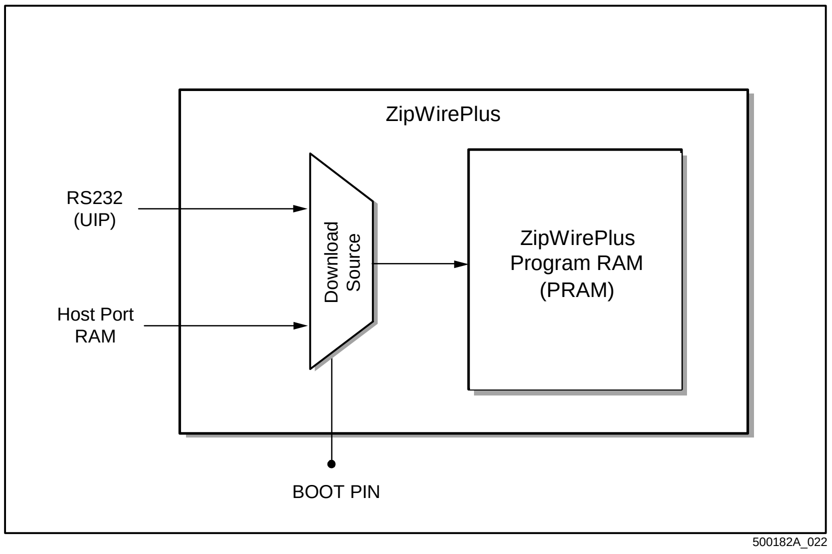
The boot code overview is provided in Section 4.3 and the operational code overview is provided in Section 4.4.

4.3 ZipWirePlus Boot Code Overview

The boot code is located on the ZipWirePlus internal ROM and is executed at power up. The primary role of the boot code is to download the operational code into the ZipWirePlus program memory. At startup, the boot code does power-on initialization of the ZipWirePlus device and is ready to accept the operational code. The AFE reset is asserted while executing the boot code. The operational code can be downloaded via the host port RAM interface or RS232 serial port interface. The UIP uses the RS232 serial port interface for download. The

UIP is the User Interface Program (TestExec) , which is an demonstration/debugging GUI software which is used to control and manage the ZipWirePlus EVMs. Both the download interfaces use the mailbox scheme to exchange API messages. Figure 4-3 illustrates the host port RAM interface and RS232 interface options available for downloading the operational code. The boot pin determines the download mode for the internal PRAM. After boot load, operational code validates download via API commands. This allows the ZipWirePlus to perform a thorough self-test.

Figure 4-3 ZipWirePlus PRAM Download Overview



NOTE: The RS232 serial interface is provided only for debugging purposes and should not be used as the main download mechanism for the program RAM.

4.3.1 PRAM Download Protocol

Figure 4-4 illustrates the host and 8051 download protocol. After power-on initialization, the 8051 generates an interrupt to the host with an _ACK_BOOT_WAKE_UP (0x0D) API message. The host waits for the _ACK_BOOT_WAKE_UP (0x0D) API message before initiating a download.

To start the download process the host issues the _DSL_DOWNLOAD_START (0x53) API command specifying the length of the operational code to be downloaded. The length is a two-byte field with the low byte programmed first. The length information is later used by the boot code to compute the PRAM checksum.

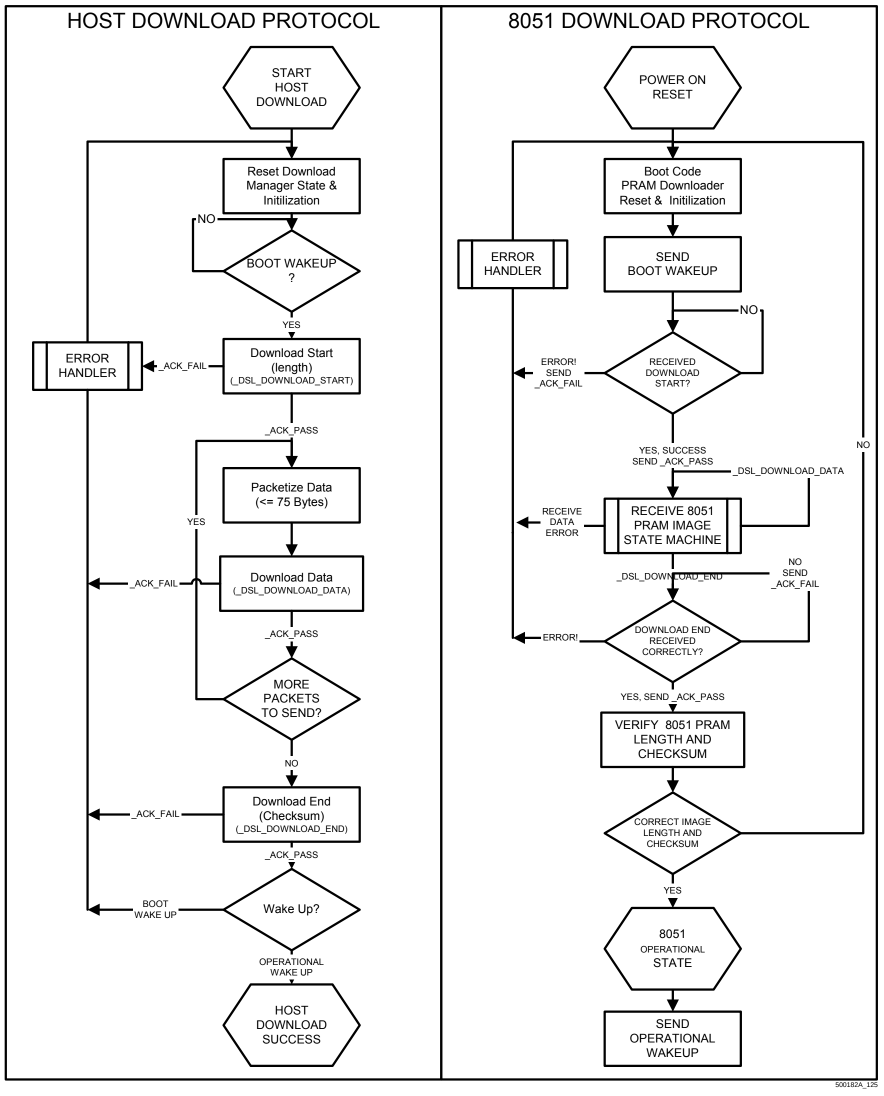
The host then breaks the downloadable image into 75-byte packets and transfers one packet at a time sequentially using the _DSL_DOWNLOAD_DATA (0x54) API command. All packets have a data parameter length of 75 bytes except the last one, which contains only the length of bytes necessary to complete the download.

It is up to the host to ensure that the previous packet was received correctly by the 8051 before sending the next sequential packet. The _DSL_DOWNLOAD_END (0x55) API command marks the end of the download process. The command also has a one-byte checksum used for validating the download to the 8051 PRAM. The one-byte checksum computation is shown in Section 4.3.6.

Upon completion of the download, the boot code computes the checksum of the downloaded code and compares it with the checksum provided by the host. If the computed checksum matches the checksum provided by the host, the program control transfers to the operational code sending an _ACK_OPER_WAKE_UP (0x0E) API message to the host.

A failed checksum or an incorrect download resets the program execution to the beginning of the boot code sending an _ACK_BOOT_WAKE_UP (0x0D) API message to the host. The download process can be reset at anytime by issuing the _DSL_RESET_SYSTEM (0x00) API command.

Figure 4-4 Host and 8051 Download Protocol



4.3.2 DEVADR[2:0] and BOOT pins

The DEVADR[2:0] and BOOT pins on the ZipWirePlus Device determine the startup operation mode. All DEVADR[2:0] and BOOT pins are inputs pins. The DEVADR[2:0] pins must be set to 000. The BOOT pin definition is given in Table 4-2.

Table 4-1 DEVADR Pin Definition

BIT#	NAME	DESCRIPTION
2-0	DEVADR[2:0]	Set to 000.

Table 4-2 BOOT Pin Definition

BIT#	NAME	DESCRIPTION
0	Boot Mode	0 = Serial–Download the program RAM from the Serial Interface (UIP). 1 = Host–Download the program RAM from the Host Port Interface.

4.3.3 Download Times

4.3.3.1 Host Port RAM

Downloading the 8051 PRAM via the host port RAM interface completes in 2–3 seconds. The exact time is a function of the internal 8051 and the host processor's clock speeds and loading.

4.3.3.2 RS232—UIP

Downloading to the 8051 Internal PRAM via the Serial Interface completes in ten seconds.

4.3.4 Download and Device Validation

There are two aspects of validation: download and device validation. The download validation consists of a simple checksum that validates that the transmitted program data successfully reached the device. The download validation is performed during the boot code mode. The checksum byte is the last byte sent. The device validation is a basic self-test that qualifies the integrity of the device. A thorough self-test should be performed on the host.

4.3.5 Download API Commands

Table 4-3 lists the boot API commands supported while the ZipWirePlus device executes the boot code. During operational mode, the device ignores the download API commands. See Chapter 15.0 for details on each command.

Table 4-3 Boot API Command Summary

COMMAND	OPCODE	DATA LENGTH	DESCRIPTION
RESET (_DSL_RESET_SYSTEM)	0x00	1	Generates a soft reset of the ZipWirePlus device.
Download Start (_DSL_DOWNLOAD_START)	0x53	2	Begins a new download. The size of the program code (length) is used to validate the download procedure. The low byte is sent first.
Download Data (_DSL_DOWNLOAD_DATA)	0x54	Up to 75 bytes	Transfers the next block of the program data.
Download End (_DSL_DOWNLOAD_END)	0x55	1	Indicates the end of the download. The download data checksum is used by the 8051 to validate the download contents.

4.3.6 Program RAM Checksum

The Program RAM checksum uses the following formula to determine the checksum.

Checksum = ~ (Byte 1 + Byte 2 + … + Byte L) + 1

Sum all of the bytes and take the 2s complement.

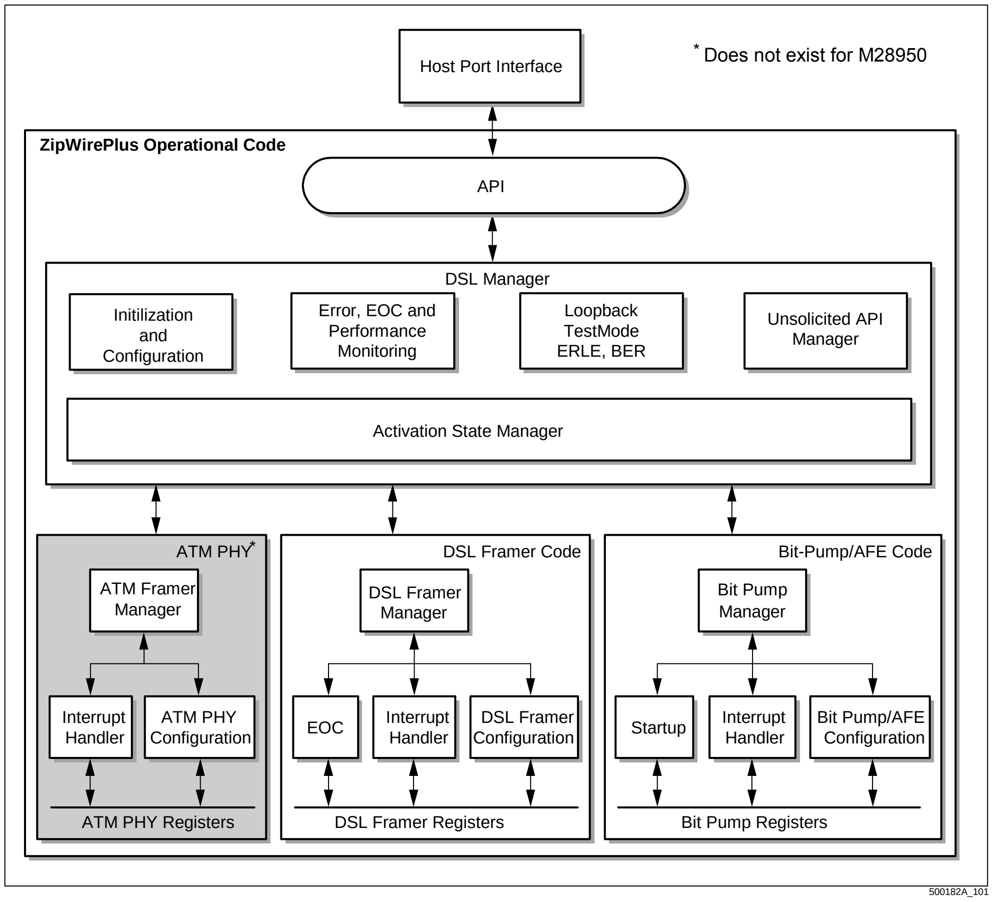
4.4 ZipWirePlus Operational Code Overview

4.4.1 ZipWirePlus Operational Code Architecture

Figure 4-5 illustrates the operational code architecture. The DSL manager calls the bit pump, DSL framer, and ATM PHY managers to activate and maintain the system. The host port RAM interface provides a way to control the system using the host processor.

NOTE: Although the AFE is a separate chip, the AFE is controlled by the DSP.

Figure 4-5 ZipWirePlus Operational Code Architecture Overview



4.4.2 ZipWirePlus Operational Code Program Flow

Figure 4-2 illustrates a detailed block diagram of the operational code main program flow. The major stages of the program execution are described below. Each stage is implemented as a separate state machine. The _DSL_STAGE_NUMBER (0x8F) API command is used to determine the execution state of the various state machines.

4.4.2.1 DSL Initialization

In the initialization stage, software puts the devices in a default configuration. The device defaults to an out-of-service state.

4.4.2.2 DSL Software Reset

When the program is continuously executing the 8051 main program loop, the host processor can issue the _DSL_RESET_SYSTEM (0x00) API command. When this API is issued, the activation stage changes to the DSL Initialization stage.

4.4.2.3 API Manager

The API manager processes API commands from the host port interface and RS232 (UIP) interface. The host processor must not send any new message until the previous message is acknowledged. The API protocol is described in Chapter 13.0.

4.4.2.4 Bit Pump Manager

The bit pump manager maintains the bit pump portion of the activation state manager. This includes the startup training process, as well as adapting to temperature and environment changes during normal operation.

4.4.2.5 DSL Framer Manager

The DSL framer manager maintains the DSL framer portion of the activation state manager.

- Framing
- Overhead bits
- Sync word
- Indicator bits
- EOC
- Performance monitoring

4.4.2.6 ATM PHY Manager

The ATM PHY manager initializes and configures the ATM TC PHY. This does not exist on the M28950 ZipWirePlus device.

4.4.2.7 DSL Activation State Manager

The activation state manager is implemented using the bit pump manager, ATM manager, and DSL framer manager. Section 4.4.3 describes the activation sequence for the ZipWirePlus modem.

4.4.3 ZipWirePlus Operational Code Activation Phases

Activating the ZipWirePlus modem requires Pre-Activation, DSP training, line coding, frame formatting, and ATM configuration(for M28945, M28946 & M28947 devices only). Table 4-4 lists the major phases of activation for the ZipWirePlus device. The Pre-Activation phase does modem detection and mode selection. Activation of the ZipWirePlus modem starts when the _DSL_ACTIVATION (0x0B) API command is issued. The time from when _DSL_ACTIVATION (0x0B) command is issued until the ZipWirePlus device is passing payload data is defined as the activation period. The ZipWirePlus solution is extremely flexible and can support sequencing and overlap of different activation modes. The Activation State Manager (ASM) coordinates the sequencing and scheduling of the activation phases.

Table 4-4 Activation Phases

Phase	Description
Pre-Activation	Pre-activation allows the two modems to communicate messages to determine the other aspects of training. The pre-activation protocol is typically independent of the line coding.
DSP Training	DSP training is the handshake that allows the two modems to train the DSP engine (adapt filters, perform timing recovery, etc.).
DSL Line Coding	DSL line coding sets the final line coding configuration, such as 4 PAM, 8 PAM, Trellis On/Off, etc.
Frame Format	Frame format determines how payload data is encapsulated into the DSL frame. The frame typically consists of a sync word, EOC, CRC, etc.
ATM PHY	ATM PHY performs cell delineation.

4.4.3.1 Activation State Manager (ASM)

The Activation State Manager (ASM) sits on top of the activation phases to determine when the link is good, and when to bring down and retrain the link. The ASM performs flow control and error handling. Figure 4-6 illustrates an overview of a typical activation sequence. Separate Activation State Managers exist for G.shdsl, HDSL2 (OPTIS), HDSL1, IDSL and SDSL. The ASM will be different for a HTU-C and HTU-R. In certain standards, the different activation phases overlap while other standards are completely independent tasks. For example, in HDSL1 (2B1Q), the frame format passes the sync word and indicator bits before the DSP finishes its training. In HDSL2 (OPTIS), the DSP must complete its training before the DSL framer begins transmitting and looking for the sync word.

Figure 4-6 Activation State Manager (ASM Overview)

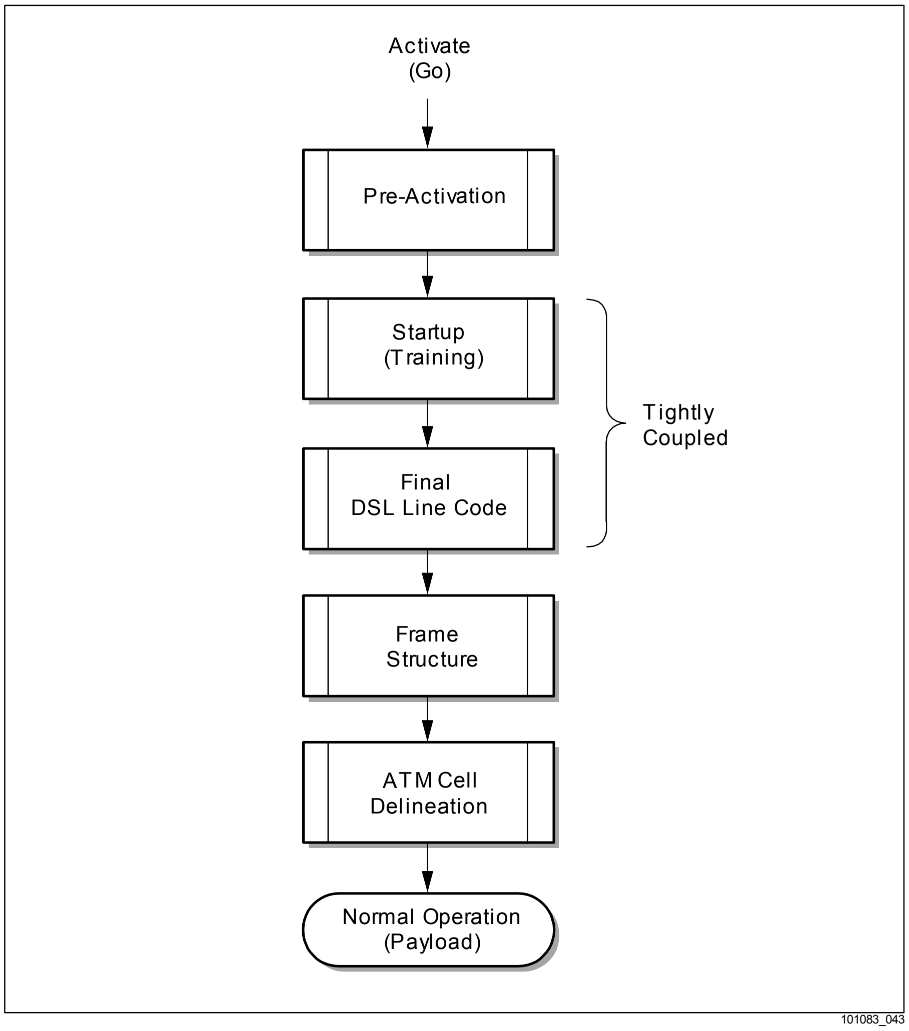
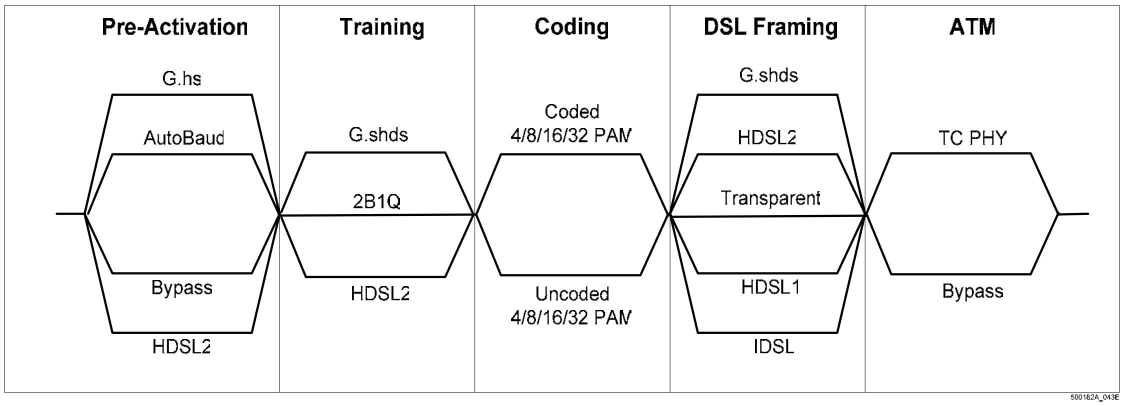


Figure 4-7 illustrates the possible modes during each activation phase. For example, on a M28945 device to configure the ZipWirePlus for G.shdsl with ATM transport, the G.hs is carried out in the Pre-Activation phase, G.shdsl is selected in the training mode, line coding is 16 PAM Coded, DSL Framing is G.shdsl, and ATM transport is TC PHY.

Figure 4-7 Activation Steps



4.4.3.2 Pre-Activation

Pre-activation can be as simple as determining if a far-end unit is present or as sophisticated as remotely configuring the far-end unit and determining the optimal data rate. The ZipWirePlus supports the following pre-activation modes:

- G.hs
- HDSL2 (OPTIS) Pre-Activation
- Auto Baud

4.4.3.3 DSP Training

The DSP (bit pump) training handshake allows the two modems to train and pass data. This includes the 2, 4, and 16-level timelines and precoder tap exchange. The ZipWirePlus supports the following training modes:

- G.shdsl—includes precoder tap exchange and can support both coded and uncoded modes
- HDSL2 (OPTIS)—includes precoder tap exchange, can support both coded and uncoded modes
- 2B1Q—always uncoded

4.4.3.4 DSL Line Coding

DSL line coding is typically defined by standards such as uncoded 4-PAM for 2B1Q or coded 16-PAM for OPTIS and G.Shdsl. The DSL line coding is tightly coupled with the training mode. In non-standard applications, there could be some advantages to switching to a different DSL line code. For example, ZipWirePlus modems could use the 2B1Q training to train the DSP then switch to uncoded 16-PAM line code; this could allow the modems to achieve twice the data rate at the expense of loop length. Another example would be using the G.shdsl to train (which includes the precoder tap exchange), then switch to a coded 4-PAM line code; this would allow the modems to achieve an extended reach at the expense of data rate.

4.4.3.5 DSL Frame Format

The ZipWirePlus supports the following DSL frame formats:

- G.shdsl
- HDSL2 (OPTIS)
- HDSL1
- IDSL (For NT Only)
- Framer Transparent (no DSL framer overhead)

The frame format is typically defined by the standards. However, as with the DSL line coding, there is flexibility in how the frame format can be configured. For example, the modems could train using the 2B1Q mode then switch to uncoded 16-PAM and use the G.shdsl frame format. Another example could train using the HDSL2 (OPTIS) coded 16-PAM then use the G.shdsl frame format. There is minimal dependency from the frame format to the DSP training. The main requirement is that the DSP data rate (regardless of training mode and DSL line coding) matches the expected frame format.

4.4.3.6 ATM TC PHY

The ATM TC PHY performs ATM cell processing (cell delineation). The ATM PHY can be enabled or disabled (bypassed) depending on the application. In ATM applications, the ATM PHY is enabled. The ATM PHY is typically bypassed in PCM applications, such as T1/E1 transport, HDLC, ISDN, etc.

NOTE: This block does not exist on the M28950 device.

4.4.3.7 Normal Operation

Based on the selected configuration settings in the previous activation phases, the following parameters are appropriately set by the operational code.

- Pulse templates
- PSD

- Transmit power
- Tomlinson coefficients—exchange of DFE coefficients into Tx precoder
- Encoder coefficients
- Scrambler/descrambler
- CRC
- Activation sequence
- EOC
- Frame structure
- Activation sequence/timeline

Scrambler/Descrambler Taps

The DSL scrambler and descrambler taps are based on the appropriate standards, as listed in Table 4-5. The terminal type and frame structure API commands determine the proper tap selection. The transmit scrambler matches the far end's receive descrambler.

Table 4-5 Scrambler/Descrambler Taps

Standard	HTU-C to HTU-R	HTU-R to HTU-C
G.shdsl	x23 + x5 + 1	x23 + x18 + 1
HDSL2	x23 + x5 + 1	x23 + x18 + 1
HDSL1	x23 + x5 + 1	x23 + x18 + 1
IDSL	x23 + x5 + 1	x23 + x18 + 1

5.0 ZipWirePlus Clocking Architecture

5.1 Clocking Architecture Overview

The ZipWirePlus family of devices provides a flexible clocking architecture to support a number of applications defined by the various standards (G.shdsl, HDSL2, HDSL1, SDSL, IDSL et al). Table 5-1 lists the supported clocking modes. In general, the HTU-C clocking architecture must support the different modes while the HTU-R clock reference is derived from the received symbol clock. The HTU-C specifics are described in Section 5.2, and the HTU-R specifics are described in Section 5.3.

Table 5-1 Clocking Modes

Mode #	Mode Name	HTU-C Clock Reference	Application
1	Plesiosynchronous	Local oscillator (free running)	Classic HDSL
2	Plesiosynchronous with timing reference	Network reference clock	Classic HDSL
3	Synchronous	Transmit data clock or network reference clock	—

5.1.1 ZipWirePlus Clocking Architecture Implementation

5.1.1.1 M28945, M28946 & M28947

The on-chip clock synthesizer (PLL) block generates the DSP, microprocessor, AFE, DSL framer, and ATM reference clocks. The ZipWirePlus DSL framer has two independent DPLLs to generate PCM and NB clocks. The on-chip clock synthesizer can operate in free running network timing reference, or HTU-R DSL/G.hs timing recovery mode. In free running mode, the ZipWirePlus clock synthesizer operates at the crystal (or local clock) phase offset. In network timing reference mode, the ZipWirePlus clock synthesizer phase-locks its timing to the external reference clock. The external reference clock can be sourced from EXT_CLK_REF pin or the internal i_TPCLK signal. In HTU-R DSL/G.hs timing recovery mode, the ZipWirePlus clock synthesizer phase-locks its timing to the far end modem (HTU-C) via the G.hs or DSL signal.

5.1.1.2 M28950

The on-chip clock synthesizer (PLL) block generates the DSP, microprocessor, AFE, and DSL framer. The ZipWirePlus DSL framer has a DPLL to generate the PCM clock. The on-chip clock synthesizer can operate in free running network timing reference, or HTU-R DSL/G.hs timing recovery mode. In free running mode, the ZipWirePlus clock synthesizer operates at the crystal (or local clock) phase offset. In network timing reference mode, the ZipWirePlus clock synthesizer phase-locks its timing to the external reference clock. The external reference clock can be sourced from EXT_CLK_REF pin or the internal i_TPCLK signal. In HTU-R DSL/G.hs timing recovery mode, the ZipWirePlus clock synthesizer phase-locks its timing to the far end modem (HTU-C) via the G.hs or DSL signal.

Table 5-2 ZipWirePlus Clocks

Clock	Frequency	Description
Crystal	22.1184 MHz	External crystal or clock input
XTALI / XTALO	22.1184 MHz	Crystal input/output

Clock	Frequency	Description
XTALO_B	22.1184 MHz	Buffered crystal output
EXT_CLK_REF (Input)	8 kHz–18.432 MHz	Bidirectional network timing reference clock
EXT_CLK_REF (Output)	8 kHz	Bidirectional network timing reference clock
SYS_CLK	43–53 MHz	Internal DSP system clock
AFE_CLK	SYS_CLK / 2	AFE clock reference
G.hs Clock	12 kHz or 20 kHz	G.hs clock – 12 kHz (HTU-R) or 20 kHz (HTU-C)
Baud Rate	115200	Baud rate that controls serial ports 0 and 1
HXCLK	64 k – 4640 kbps	Data rate clock output
Crystal	22.1184 MHz	External crystal or clock input
XTALI / XTALO	22.1184 MHz	Crystal input/output

5.2 HTU-C Clocking Modes

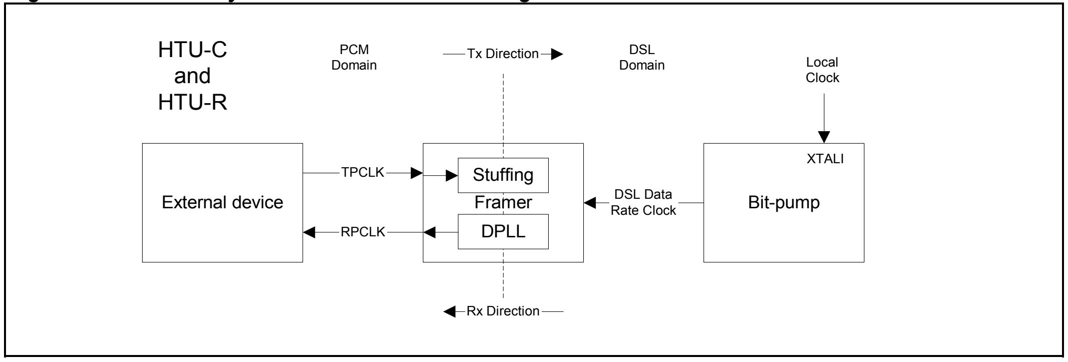
This section describes how the various clocking modes are targeted for HTU-C applications.

5.2.1 Plesiosynchronous Mode

Figure 5-1illustrates a simplified block diagram of the plesiosynchronous clocking mode. In this mode, the transmit PCM clock and DSL clock operate independently (within appropriate PPM tolerance) and do not have any phase relationship with respect to each other. The stuffing generator is used to compensate for any phase differences between the PCM and DSL clock domains. The HTU-C locks the DSL clock to a local clock or oscillator. The transmit PCM clock (TPCLK) signal can be generated by an external device such as BT8370 (T1/E1 framer) or can be sourced from the PCM DPLL when operating in open loop mode. The transmit PCM and receive PCM clocks can operate at independent rates (within the appropriate PPM tolerance). T1/E1 transport applications use this mode.

NOTE: This figure applies to both the HTU-C and HTU-R

Figure 5-1 Plesiosynchronous Mode Block Diagram

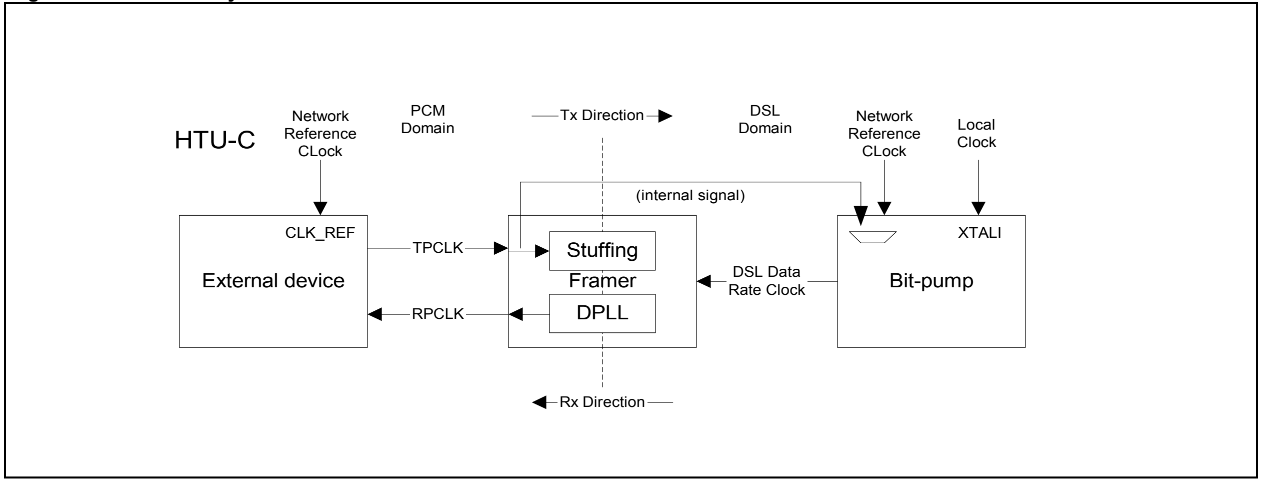


5.2.2 Plesiosynchronous Mode With External Clock Reference

Figure 5-2 illustrates a simplified block diagram of the plesiosynchronous mode with external reference clock. This mode is similar to the plesiosynchronous mode except the HTU-C locks the DSL clock to an external reference clock—either from a network reference clock or the transmit PCM clock.

If the external reference clock is sourced from the TPCLK pin, the DSL and PCM clock domains are synchronized. However, the stuffing generator is still used and is therefore a slightly different configuration than the synchronous modes described in Section 5.2.3. This configuration is used when the transmit PCM clock is locked to a network reference clock and the DSL clock needs to be synchronized to the network reference clock via the transmit PCM clock.

Figure 5-2 Plesiosynchronous mode With External Reference Clock



5.2.3 Synchronous Mode

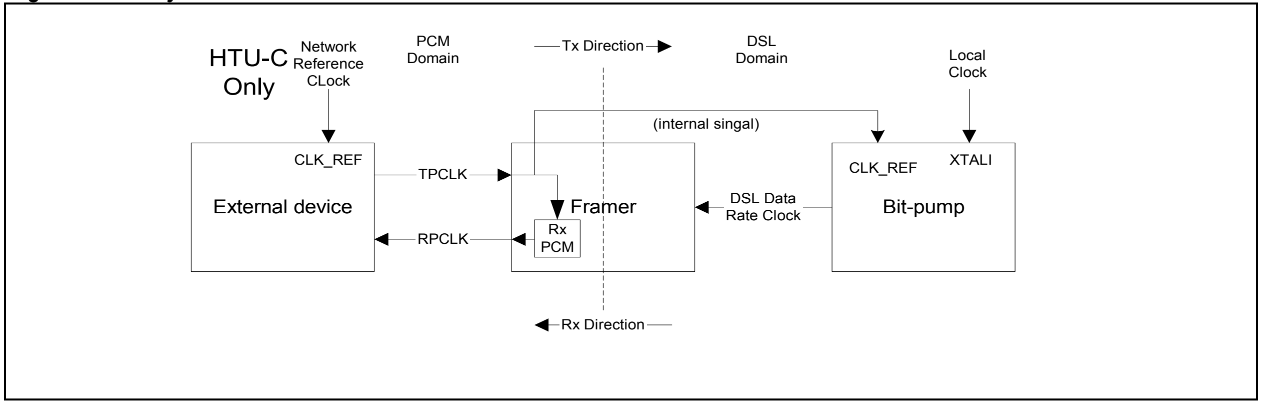
In the synchronous mode, the DSL and PCM clock domains are synchronized. The stuffing generator is therefore disabled. The synchronous mode can be achieved in 2 different ways:

- Synchronous—slave transmit data clock
- Synchronous—master transmit data clock

5.2.3.1 Synchronous— Slave Transmit Data Clock

Figure 5-3 illustrates a simplified block diagram of the slave transmit data clock mode. In this configuration, the ZipWirePlus's transmit PCM clock is sourced from an external device. In this mode, the HTU-C locks the DSL clock to the transmit PCM clock (TPCLK). The transmit PCM clock (TPCLK) and receive PCM clock (RPCLK) are both sourced (slaved) from the TPCLK input pin. The TPCLK must be supplied from an external source. The PCM DPLL is not used.

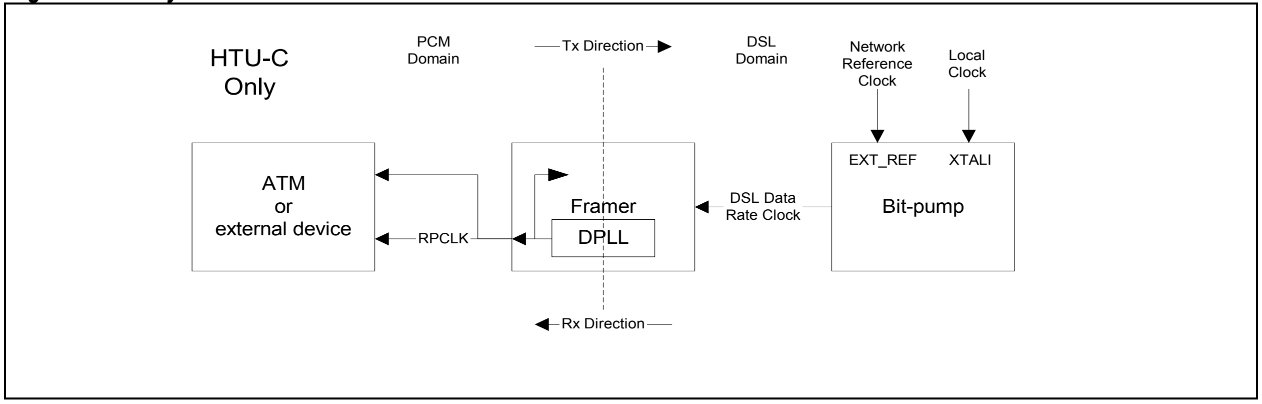
Figure 5-3 Synchronous—Slave Transmit Data Clock



5.2.3.2 Synchronous—Master Transmit Data Clock

Figure 5-4 illustrates a simplified block diagram of the master transmit data clock mode. In this configuration, the ZipWirePlus device is the PCM clock master. Any external device must be slaved to the ZipWirePlus device clocks. The DSL clock domain is synchronized to the local oscillator or the external network reference clock. The transmit PCM clock (TPCLK) and receive PCM clock (RPCLK) are then synchronized to the DSL clock using the PCM DPLL.

Figure 5-4 Synchronous—Master Transmit Data Clock



5.3 HTU-R Clocking—Network Reference Clock Output

This section describes how the clocking modes are targeted for HTU-R applications. In general, the HTU-R recovers the DSL, PCM, and network references clocks from the incoming DSL line. The PCM clock can operate in looped timed or independent transmit/receive PCM clock modes. The network reference clock output is optional.

NOTE: When operating as an HTU-R, the device can support all of the HTU-C clocking schemes but the HTU-C modes are not required in most applications.

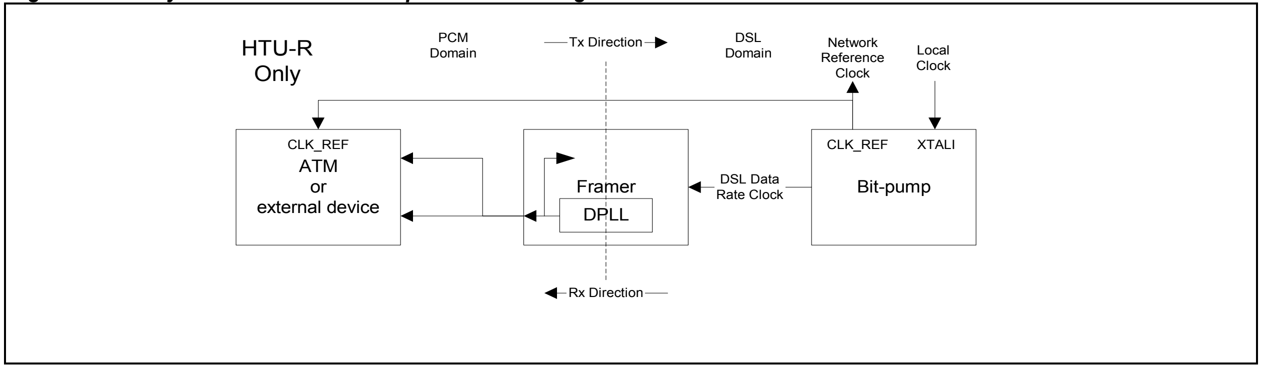
5.3.1 Independent Transmit/Receive PCM clocks

The transmit PCM and receive PCM clocks can operate at independent rates (within the appropriate PPM tolerance). This mode is the same as the HTU-C plesiosynchronous mode. See Section 5.2.1 for details and block diagram.

5.3.2 PCM Loop Timed Clocking Mode

Figure 5-5 illustrates a simplified block diagram of the HTU-R PCM loop-timed clocking mode. The PCM loop-timed clock mode takes the PCM DPLL recovered clock (RPCLK) and uses it for both the transmit PCM and receive PCM directions. This mode is applicable in stuffing and non-stuffing modes. This configuration is similar to the Section 5.2.1.

Figure 5-5 Synchronous—PCM Loop-Timed Clocking Mode



6.0 ZipWirePlus Application Interface Configurations

This chapter contains configuration information specific to the ATM and Narrowband interfaces. Section 6.1 describes the ATM interface configuration details and Section 6.2 describes the Narrowband interface configuration details for the ZipWirePlus device.

Please note that the ATM Interface and the Narrowband Interface is available on the M28945, M28946 and M28947 devices. This section is not valid for the M28950 device.

6.1 ZipWirePlus ATM Interface Configuration

6.1.1 ATM One Second Timing

The ATM PHY TC requires a one-second tick to update internal status and performance monitoring counters. In a system, all devices must use a common one-second counter in order for performance monitor counters to be synchronized. The device can self-generate the one-second tick from the clock synthesizer block or optionally source the one second tick from an external device. In addition, the device can optionally output the one-second tick to external devices. The One Second Master/Slave bit in the _ATM_PHY_IF_MODE (0x1E) determines the one-second source. The following HDSL Interface signals are used to drive the one second signals.

- One Second Output is connected to DSLSYNCO
- One Second Input is connected from DSLSYNCI

6.1.2 ATM PHY API Commands

Table 6-1 is a summary of the ATM PHY API related commands. See Chapter 15.0 for a detailed description of the API commands.

Table 6-1 ATM PHY API Command Summary

Command	Opcode	Description
_DSL_LOOPBACK	0x09	The ATM PHY provides source loopback (_ATM_SOURCE_LB). See Section 12.2.
_ATM_PHY_MODE	0x1C	Configure the ATM PHY section to the desired mode (general purpose, 1T1 or 1E1).
_ATM_PHY_UTOPIA_CONFIG	0x1D	Configure the ATM UTOPIA interface mode.
_ATM_PHY_IF_MODE	0x1E	Configure the ATM general multiplexing mode.
_ATM_PHY_INJECT_HEC_ERROR	0x1F	Inject a HEC error in the next ATM cell.
_ATM_PHY_CONFIGURE	0x20	Configure the ATM UTOPIA address, scrambler, HEC Coset, and Auto Correct HEC modes. The valid UTOPIA address range is 0x00 to 0x1E where 0x1F is reserved for the NULL address.
DSL_CLEAR_ERROR_CTRS	0x40	Clear the error and statistical counters.
_DSL_STATUS	0x85	The STATUS_5 byte contains ATM PHY error indicators.
_ATM_PHY_OPER_ERR_CTRS	0xB8	Read ATM PHY transmit and receive cell counters.
_ATM_PHY_PERF_ERR_CTRS	0xB9	Read ATM PHY performance counters.
_ATM_PHY_CELL_CTRS	0xBA	Read ATM PHY transmit and receive cell counters.

6.2 ZipWirePlus Narrowband Interface Configuration

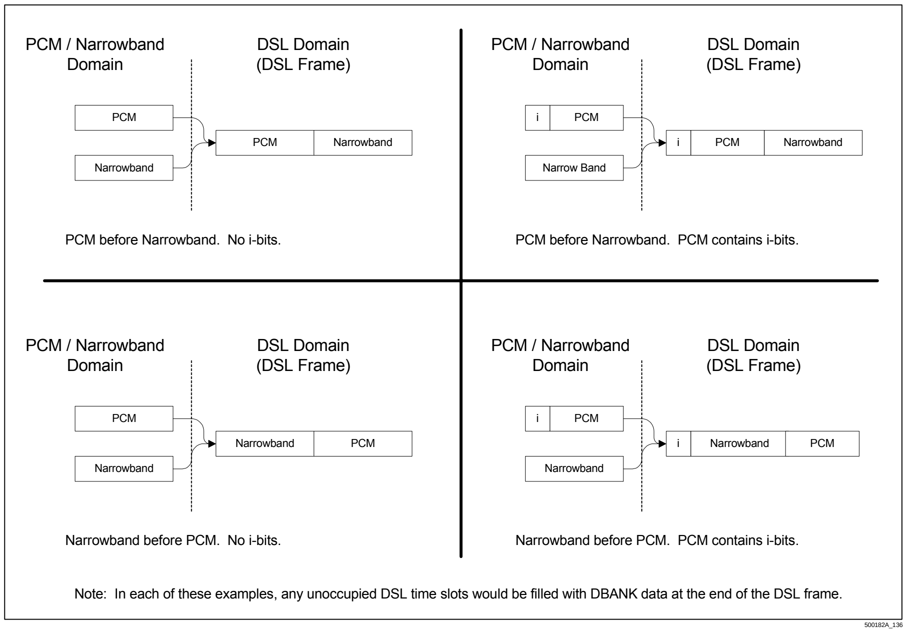
The Narrowband interface is a second PCM channel targeted for multitransport applications, such as ISDN plus E1, V.35 plus E1, etc. The ZipWirePlus device can operate in PCM only and PCM plus Narrowband modes.

6.2.1 Mapping into the DSL Frame

The Narrowband and PCM mapping to the DSL frame creates an enormous possible set of combinations. To simplify the application interface, the _DSL_NB_MULTI_RATE_CONFIG (0x1A) applies simple rules to address most applications. The PCM, Narrowband, and DSL mapper API commands allow for custom mapping.

1. Mapping Order: The customer can select either the PCM or Narrowband to be located first in the DSL frame. However, if the PCM contains an i-bit, the i-bit are always located first.
2. Only single-pair applications and block mapping are supported.

Figure 6-1 PCM / Narrowband into DSL Frame Mapping Examples



6.2.2 Narrowband API Command Set

Table 6-2 lists a summary of the Narrowband API commands.

Table 6-2 Narrowband API Commands Summary

Command	Control Status	Opcode	Description
_DSL_FR_PCM_CONFIG	Control	0x10	Selects the PCM/NB active ports.
_DSL_NB_CONFIG	Control	0x28	Configure the Narrowband interface parameters.
_DSL_NB_MULTI_RATE_CONFIG	Control	0x1A	Configure the Narrowband data rate and mapping.
_DSL_TNB_MAPPER_VALUE	Control	0x38	Transmit Narrowband mapper table.
_DSL_TNB_MAPPER_WRITE	Control	0x39	Transmit Narrowband mapper table write.
_DSL_TNB_MAPPER_READ	Status	0xA7	Transmit Narrowband mapper table read
_DSL_RNB_MAPPER_VALUE	Control	0x3A	Receive Narrowband mapper table.
_DSL_RNB_MAPPER_WRITE	Control	0x3B	Receive Narrowband mapper table write.
_DSL_RNB_MAPPER_READ	Status	0xA8	Receive Narrowband mapper table read.
_DSL_TNB_FRM_OFST	Control	0x29	Configure the transmit Narrowband frame offset.
_DSL_RNB_FRM_OFST	Control	0x2A	Configure the receive Narrowband frame offset.
_DSL_NB_DPLL_CLOCK_GEN	Control	0x59	Configure the Narrowband DPLL.
_DSL_OPER_ERR_CTRS	Status	0x9C	Contains PCM and Narrowband operational error counters.

7.0 Pre-Activation

This chapter describes the G.hs, SDSL (Auto Baud) and HDSL2 pre-activation sequences.

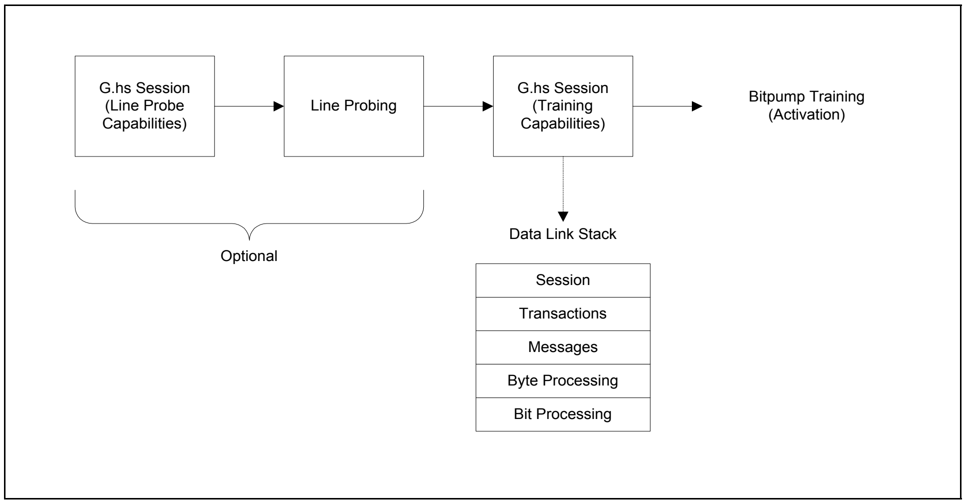
7.1 G.hs

The G.hs is a pre-activation sequence used to synchronize the modems, perform remote configuration, and to optionally perform line probing in order to determine an optimal data rate in rate-adaptive applications. Remote configuration consists of the DSL data rate, clocking mode, framer modes, application interfaces, etc.

7.1.1 G.hs Detailed Description

Figure 7-1 illustrates the G.hs pre-activation sequence. The G.hs session manager is described in Section 7.1.1.1, the line probing is described in Section 7.1.1.2, and power back-off is described in Section 0.

Figure 7-1 G.hs Pre-Activation Overview

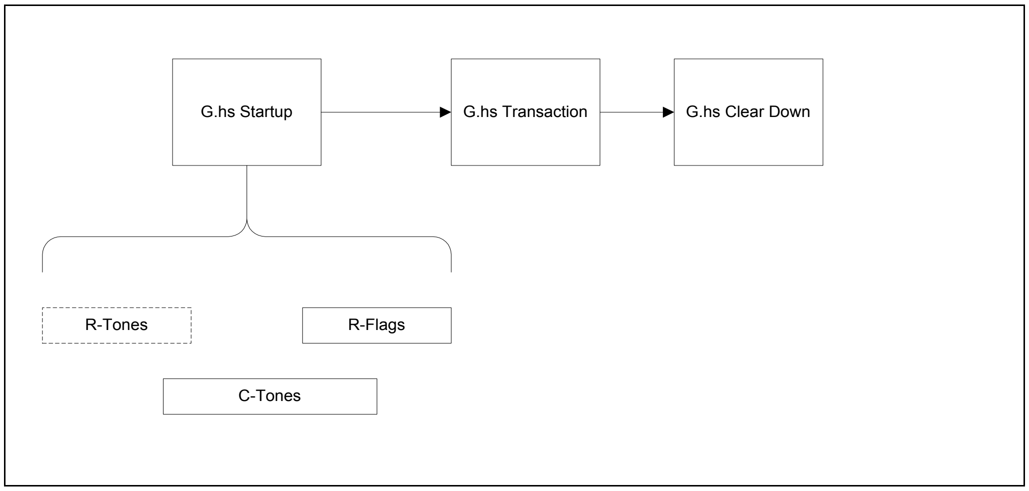


G.hs Session (Line Probe Capabilities)
Line Probing
G.hs Session (Training Capabilities)
Bitpump Training (Activation)
Optional
Data Link Stack
Session
Transactions
Messages
Byte Processing
Bit Processing

7.1.1.1 G.hs Session Manager

Figure 7-2 illustrates the G.hs session manager.

Figure 7-2 G.hs Session Overview



G.hs Startup

The G.hs startup synchronizes the two modems and adapts the bit pump. The HTU-R also performs timing recovery. The G.hs standard defines an R-initiated and C-initiated started. The ZipWirePlus firmware automatically handles both the cases.

G.hs Transaction Manager

The G.hs transaction consists of one or more messages, including ACKs and NACKs. A G.hs message uses an HDLC protocol (begin flags, data, data transparency, FCS, and end flags). The ZipWirePlus firmware handles all protocols and messages. The standard requires the HTU-R to initiate all transactions. Several API commands are provided to allow the host to control the message flow and to specify certain message parameter information, see Section 7.1.3 for more details.

Table 7-1 G.Hs Message Types

Value	Message Type	Description
0	MS	Mode Select
1	MR	Mode Request
2	CL	Capabilities List
3	CLR	Capabilities List and Request
4	MP	Mode Proposal
16	ACK (1)	Acknowledge, Type 1
17	ACK (2)	Acknowledge, Type 2
32	NAK-EF	Negative Acknowledge, Errored Frame
33	NAK-NR	Negative Acknowledge, Not Ready
34	NAK-NS	Negative Acknowledge, Not Supported
35	NAK-CD	Negative Acknowledge, Clear Down
52	REQ-MS	Request MS Message

Value	Message Type	Description
53	REQ-MR	Request MR Message
55	REQ-CLR	Request CLR Message

G.hs Clear Down

The G.hs clear down terminates the G.hs session. The G.hs clear down consists of sending a R-GALF or C-GALF depending on the transaction sequencing.

7.1.1.2 Line Probing

Line probing is optional and is only required in rate-adaptive applications. The Line Probe Enable field in the _DSL_PREACTIVATION_CFG (0x0F) API command sets the line probe state. The line probe sends out a series of broadband signals to allow the modems to characterize the channel (noise, attenuation, bridge taps, etc). This information is used to determine the optimal data rate.

When line probing is enabled, the ZipWirePlus firmware performs all of the line probing functions, transmission and reception of the signals, power back-off, and uses a proprietary algorithm to determine the optimal data rate.

NOTE: The line probe will only be performed if the Line Probe Enable API command is set and the far end also has enabled line probe.

7.1.1.3 Power Back-Off (PBO)

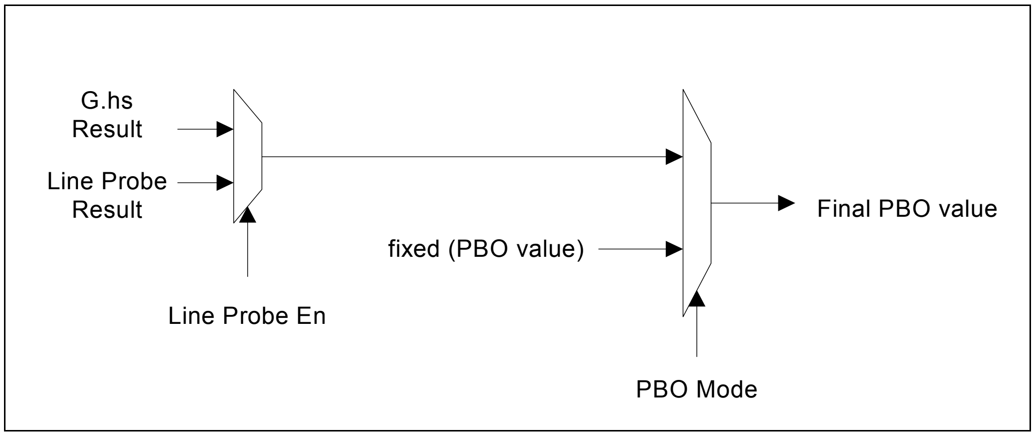
Power back-off (PBO) indicates the desired received power (the amount of transmit power to drop at the far end). This is used on short loops to avoid saturating the receiver.

There is separate line probe and training PBO value. The ZipWirePlus firmware automatically handles the line probe PBO including the relative power level field specified in the G.hs information fields.

The training PBO value is determined from the G.hs startup (C-tone or R-tone) or from line probing (if enabled). The PBO Mode and PBO Value fields in the _DSL_PREACTIVATION_CFG (0x0F) API command give the customer the following options (as illustrated in Figure 7-3):

- ♦ Automatic–use PBO value from G.hs or line probe
- ♦ Fixed–use PBO value to specify a fixed PBO value, use 0 for no PBO.

Figure 7-3 Power Back-off Selection



7.1.2 Remote Mode Configuration

The primary goal of G.hs is to establish a common mode (configuration) between the two modems. The G.hs session configures the data rate, PBO, sync word, stuff bits, and clocking modes.

NOTE: The PBO, sync word, and stuff bits can have different values in the downstream (HTU-C towards HTU-R) vs. upstream (HTU-R towards HTU-C) direction.

7.1.2.1 Capabilities List vs. Mode Select Message

The capabilities list (request) message exchanges all capabilities of each modem to the far end. The mode select message then selects only one mode from the capabilities list based on API configuration settings, line probe, etc.

The _DSL_PREACT_GET_FE_CAPS (0x88) API command provides the far end's capabilities list to the host processor.

7.1.2.2 Mode Select–Which Modem is in Control

The G.hs standard allows either the HTU-C or HTU-R to have control in determining the final configuration. This can create a situation where the two modems do not reach an agreement and thus never train. The mode select sender field in the _DSL_PREACTIVATION_CFG (0x0F) API command is used to prevent a false agreement. The ZipWirePlus firmware provides the following options:

- HTU-C configures HTU-R
- HTU-R configures HTU-C
- No configuration information exchanged, use hard coded configuration

The Mode Select message is the final message used to determine the final configuration (data rate, modes, etc). The G.hs block is written to give the HTU-C (central office) the control. However, if the far-end refuses to support the desired mode because it is configured differently, the driver will fall back to a default mode. The Mode Select matrix is shown in Table 7-2.

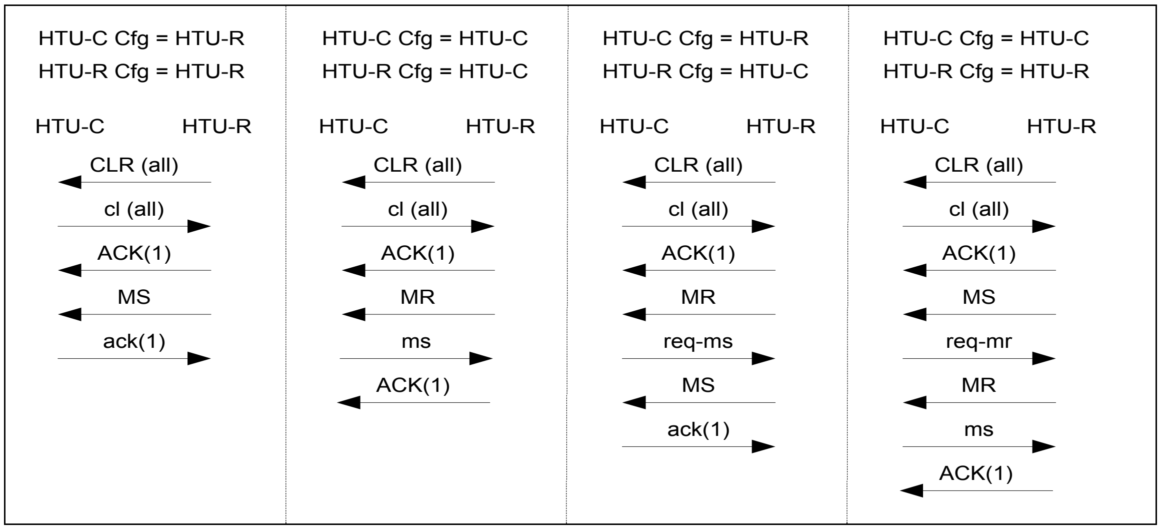
NOTE: The final mode select sender matches the HTU-C configuration, regardless of the HTU-R configuration.

Table 7-2 Mode Select Sender Matrix

HTU-C Configuration	HTU-R Configuration	Final Mode Select Sender
HTU-R	HTU-R	HTU-R
HTU-C	HTU-C	HTU-C
HTU-R	HTU-C	HTU-R
HTU-C	HTU-R	HTU-C

Figure 7-4 illustrates the G.hs message (transaction) sequencing for the various mode select sender configuration options. Uppercase letters represent HTU-R to HTU-C messages while lowercase represent HTU-C to HTU-R messages.

Figure 7-4 Mode Select Sender Message Sequencing



No Remote Configuration

This option uses a proprietary bit in the G.hs messages. Both HTU-C and HTU-R must be configured in the same mode for this option to function properly. If there is a disagreement between the modem configurations, the modems default to the HTU-C configures HTU-R option.

7.1.2.3 Determining the Final Data Rate

The final data rate is determined from one of the following options:

- _DSL_MULTI_RATE_CONFIG (0x1B) API command is used for fixed data rate applications
- _DSL_PREACT_RATE_LIST (0x15) API command is used for line probe or fixed rate applications (depending on the value of the “Line Probe or Fixed” field.
- Line Probe–when line probing is enabled
- G.hs Mode Select–slaved to the far end, see Section 7.1.2.2.

The final mode is a function of several API commands and G.hs transactions.

Data Rate: Fixed-Rate Applications

In fixed-rate applications, use the _DSL_MULTI_RATE_CONFIG (0x1B) API command to determine the final data rate. This provides a consistency with existing API commands and when executing loopbacks and test modes. A fixed rate application is determined by disabling the line probe functionality.

Or alternatively the _DSL_PREACT_RATE_LIST (0x15) API command can be used to determine the final data rate with the “Line Probe or Fixed” field being set to 0x01. Only a single rate can be specified in the _DSL_PREACT_RATE_LIST (0x15) API command parameter for the fixed-rate Application.

Data Rate Table: Rate Adaptive Applications

In rate-adaptive applications, the modem responsible for determining the final data rate uses the line probing information to determine the theoretical capabilities of the channel. The driver then cross-references the line probe data rate with a customer-defined data rate table. The driver selects the highest possible data rate from the table. The data rate source byte in the _DSL_PREACTIVATION_CFG (0x0F) API command allows the customer to enter the data rate table in two different mechanisms.

Table 7-3 Data Rate Source Options

Option	Description
List	Use a list to determine which rates are allowable during the G.hs transactions. See Data Rate Source: List Option below.
Range	Use all supported data rates within a specified range. See Data Rate Source: Range Option below.

Data Rate Entry, Specifying the i-bit Mask

The Data Rate Source List and Range options both include an i-bit mask parameter. The i-bits mask values apply to all N x 64 k values within the specified list or range. The i-bit mask bit definition is consistent with the G.hs standards.

Table 7-4 Date Rate I-bit Mask

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
i = 7 supported	i = 6 supported	i = 5 supported	i = 4 supported	i = 3 supported	i = 2 supported	i = 1 supported	i = 0 supported

Data Rate Source: List Option

Use the _DSL_PREACT_RATE_LIST (0x15) API command to specify the list of supported rates. This command is only valid if the data rate source byte is set to the list option. The number of entries in the list can be from 0-72.

The i-bits mask (byte #1) applies to all other N x 64 k entries in the list table. The modem can support up to 72 N x 64 k entries. The N x 64 k values must be in increasing order (lowest rate first and highest rate last).

For example, assume the following settings:

i-bit = 0 and 1 are enabled, i-bits = 2 to 7 are disabled

1st N x 64k = 3

2nd N x 64k = 12

3rd N x 64k = 24

4th N x 64k = 36 (also the last entry)

The possible payload data rate values become:

192k, 200k (N = 3 for i=0,1)

768k, 776k (N = 12 for i=0,1)

1536k, 1544k (N = 24 for i=0,1)

2304k, 2312k (N = 36 for i=0,1)

Data Rate Source: Range Option

The Range: i-bits mask, min N x 64k, and Max N x 64 k bytes of the _DSL_PREACTIVATION_CFG (0x0F) API command are valid only if the Data Rate Source byte is set to the Range option. Otherwise, these parameters must be 0. For example, assume the following settings:

i-bit = 0 and 1 are enabled, i-bits = 2 to 7 are disabled.

Min N x 64k = 3

Max N x 64k = 89

The possible payload data rate values become:

192k, 200k (N = 3 for i=0,1)

256k, 264k (N = 4 for i=0,1)

320k, 328k (N = 5 for i=0,1)

:

2240k, 2248k (N = 35 for i=0,1)

2304k, 2312k (N = 36 for i=0,1)

7.1.2.4 TPS-TC Clocking Mode Configuration

The following clocking modes are defined by the G.shdsl standard:

- ♦ Plesiosynchronous
- ♦ Plesiosynchronous with timing reference
- ♦ Synchronous
- ♦ Hybrid (Not Supported by ZipWirePlus devices)

The modem that sends the Mode Select uses the _DSL_CLOCK_CONFIG (0x04) API command to set the G.hs clocking mode. See Chapter 5.0 for detailed information on ZipWirePlus clocking modes.

7.1.2.5 TPS-TC Mode Configuration

The TPS-TC parameters are used to configure the application interface. Based on the application and device, this can also be considered the PCM or ATM interface. The TPS-TC information is not remotely configured via G.hs. The TPS-TC configuration should be done using the EOC channel.

The TPS-TC parameter in the _DSL_PREACTIVATION_CFG (0x0F) API command determines how the local modem's application interface is configured after G.hs is completed. The ZipWirePlus family of devices supports the following TPS-TC modes:

- ♦ None–do not reconfigure the application interface. Used when the application is not defined by the standards, i.e. PCM = 8 MHz.
- ♦ Clear channel
- ♦ Clear channel byte-aligned
- ♦ Unaligned DS1 (T1/E1)
- ♦ Aligned/Fractional DS1 (T1/E1)
- ♦ ATM
- ♦ Synchronous ISDN
- ♦ Single/Dual Bearer Modes

Table 7-5 defines how the modem is configured for the various TPS-TC combinations. The table simplifies the matrix and combines bits when possible.

Table 7-5 G.hs TPS-TC Configuration Matrix

ISDN	DS1 (aligned or Unaligned)	ATM	Clear Chan Byte aligned	Clear Chan	Device Configuration
0	0	0	0	0	Do not modify
0	0	0	0	1	HDLC application ATM disabled PCM = DSL PCM Float = 1 (unaligned)
0	0	0	1	0	Clear Channel Byte-aligned Application ATM disabled PCM = DSL PCM Float = 0 (aligned)
0	0	1	X	X	ATM application ATM enabled, general-purpose mode PCM = DSL ATM Byte-aligned = 1 (aligned)

ISDN	DS1 (aligned or Unaligned)	ATM	Clear Chan Byte aligned	Clear Chan	Device Configuration
0	1	0	X	X	E1/T1 transport application ATM disabled PCM = Fixed rate; 1.544 MHz (T1) or 2.048 MHz (E1) Alignment depends on bit 3 and bit 4
0	1	1	X	X	ATM over E1/T1 application ATM enabled; T1 or E1 mode PCM = Fixed rate; 1.544 MHz (T1) or 2.048 MHz (E1) Alignment depends on bit 3 and bit 4
1	X	X	X	X	Synchronous ISDN

7.1.3 G.hs API Commands Summary

Table 7-6 lists a summary of the G.hs API commands.

Table 7-6 G.hs API Commands Summary

Command	Control Status	Opcode	Description
_DSL_PREACTIVATION_CFG	Control	0x0F	Configures the DSL pre-activation settings.
_DSL_PREACT_USER_INFO	Control	0x14	Configures the G.hs user information.
_DSL_PREACT_RATE_LIST	Control	0x15	Specifies the list of supported payload data rates.
_DSL_PREACT_GET_FE_CAPS	Status	0x88	Returns the far-end capabilities information exchanged during the G.hs session.
_DSL_PREACT_GET_OPT_DATA_RATE	Status	0x89	Returns optimal data rate determined during line probe.

7.2 Auto Baud

The Auto Baud is a 2B1Q (SDSL) pre-activation sequence used to synchronize the modems, perform remote configuration, and to optionally perform line probing in order to determine an optimal data rate in rate-adaptive applications. Remote configuration consists of setting the DSL data rate and layer 1 and 2 information. The Auto Baud pre-activation is compatible across the ZipWirePlus family, M28985, RS8973, BT8970, and BT8960 HDSL/SDSL product lines.

NOTE: This document only describes the Auto Baud pre-algorithm. ZipWirePlus firmware has not implemented the Auto Baud post; it is up to the customer to implement this feature.

7.2.1 Message Sequencing

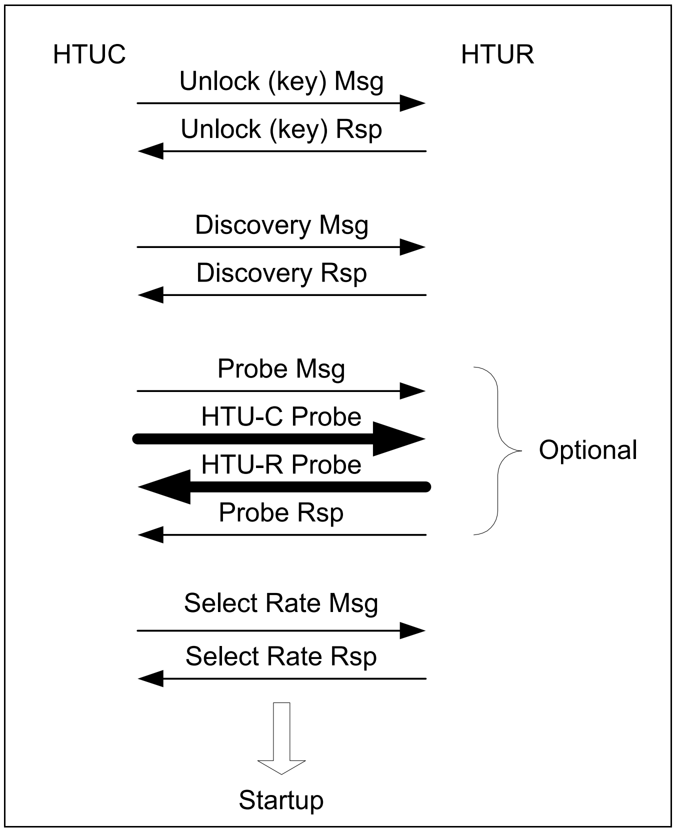
The Auto Baud pre-activation issues a series of half-duplex messages to determine if the far end is present, determine any layer 1 and 2 requirements, optionally perform line probe, and select the desired DSL data rate. The message channel is a master/slave implementation where the HTU-C initiates all messages. The HTU-R only reacts to incoming messages.

The protocol has built in error recovery, retry, and timeout logic. The protocol consists of four transactions: 1. the UNLOCK (key) phase, 2. the DISCOVERY phase, 3. PROBE phase, and 4. RATE phase. Each message consists of an op-code with associated data (if required). The HTU-R decodes the op-code and takes the appropriate action. The HTU-R indicates the successful reception of a message by returning a response with the op-code that was received. If there was an error in reception, the HTU-R returns a NAK packet to the HTU-C. The HTU-C returns to the beginning of the current phase and retries three times. If the retry threshold is exceeded, the HTU-C returns to the beginning of the protocol and retry.

Each message phase of the HTU-C has a 3-second timeout threshold. The HTU-R's state machine is event-driven and, therefore, does not use timeouts.

Figure 7-5 illustrates the message sequencing.

Figure 7-5 Auto Baud Pre-Activation Message Sequencing



7.2.2 Message Format and Descriptions

Each message consists of

- 1 start bit, logic 1
- 4-bit opcode field
- 20-bit data field
- 8-bit checksum field

7.2.2.1 Bit Order Transmission

The most significant bit is transmitted first. The start bit is sent before the 32-bit message.

7.2.2.2 Turn Around Time

A delay of 200 ms is added before the near end transmits its message (start bit) to allow the far end to prepare its receiver.

7.2.2.3 Message Time

Each message takes 1.65 seconds while a transaction takes ~3.5 seconds (message + response + turnaround time). The message sequence takes ~15 seconds when the line probe is enabled.

7.2.2.4 Message Opcode Definitions

Table 7-7 provides a summary of the messages. The detailed data field is described for each message.

Table 7-7 Message Opcode Definitions

Message Type	Opcode Value	Data Field Description
HTU-C Discovery	0x0	Layer 1 and 2 information
HTU-R Discovery	0x1	Layer 1 and 2 information
Probe	0x2	Line Probe Rate
Select Rate	0x3	Desired Data Rate
Unlock (key)	0x7	Fixed value
NACK	0xF	Error Code

7.2.2.5 Unlock (Key) Phase

The unlock (key) phase initiates the Auto Baud pre-activation messaging. All other messages are ignored until the key message is detected. This message is provided to prevent false message detection when interfacing with legacy non-Auto Baud pre-activation systems.

7.2.2.6 Discovery Phase (HTU-C and HTU-R)

The discovery phase is used to exchange layer 1 and layer 2 information between the two DSL units. This allows the HTU-R application layer to be remotely configured from the HTU-C.

The byte #1 of the data field contains the layer 1 and layer 2 information. The byte #2 of the data field contains specific information about the selected protocol. This implies the second byte will have a different meaning depending on the first byte's values. The extended byte contains the training mode.

Table 7-8 Discovery: Byte #1 Definition

Bit 7:6 Version Number	Bit 5 Reserved	Bit 4 AB Post Enabled	Bit 3:2 Framer Type	Bit 1:0 Layer 2 Type
Auto Baud Version Number	0	0 = Disable 1 = Enable	0 = None 1 = HDSL T1/E1 2 = ATM over CAP	0 = HDLC 1 = ATM 2 = T1/E1/Other

Table 7-9 Discovery: Byte #2 Definition for ATM Applications

Bit 7:4 Reserved	Bit 3 ILMI Protocol (1)	Bit 2 COSET	Bit 1:0 Scrambler Type
0x00	0 = Not supported 1 = Supported	0 = Disabled 1 = Enabled	0 = No Scrambling 1 = SSS Scrambling 2 = DSS Scrambling 3 = Reserved

NOTE: The ILMI protocol is a management channel protocol.

Table 7-10 Discovery: Byte #2 Definition for HDLC Applications

Bit 7:4 Reserved	Bit 3 Data-bit Order	Bit 2 CRC Type	Bit 1 CMCP Protocol (1)	Bit 0 ILMI Protocol (1)
0x00	0 = ANSI 1 = not ANSI	0 = HDLC 16 1 = HDLC 32	0 = Not supported 1 = Supported	0 = Not supported 1 = Supported

NOTE: The ILMI and CMCP protocols are management channel protocol.

Table 7-10 Discover: Byte #2 Definition for T1/E1 Applications

Bit 7:3 Reserved	Bit 2:1 Framing Mode	Bit 0 T1/E1 Mode
0x00	0 = Unframed 1 = ESF 2 = SF	0 = E1 1 = T1

7.2.2.7 Probe Phase

The probe transaction phase performs rate determination by resolving line length and line quality. The sequence starts by the HTU-C issuing a probe message to the HTU-R followed by a 1-second line probe; the message contains the desired line probe rate. The HTU-R then sends a 1-second line probe to the HTU-C. The HTU-R then sends a response to the probe message with the line attenuation value calculated by the HTU-C probe in the data section of the message. The HTU-C compares the line attenuation received with the value it calculated using the HTU-R probe and uses the worst-case value. The worst-case value determines the highest rate the HTU-C and HTU-R can achieve without errors.

The data field determines the data rate of the line probe. The data field formula is consistent with the _DSL_DATA_RATE (0x0E) API command.

Table 7-11 HTU-C to HTU-R Probe Request Data Field Definition

Bit 15:10 Reserved	Bit 9:0 Data Rate
0x00	Value = Data Rate / 8 k A value of 0x00 will force 160 kbps

NOTE: The ZipWirePlus firmware automatically selects the data rate(s) that run during the probe.

7.2.2.8 Select Data Rate Phase

The select rate transaction phase performs rate negotiation between the HTU-C and the HTU-R. The HTU-C side uses the line quality information and customer rate to list to determine the final data rate.

The HTU-C issues the rate message with the suggested rate to the HTU-R. If the HTU-R cannot support that rate based on its rate tables, it will pick the nearest lower rate it can support and suggest it to the HTU-C. The HTU-R achieves this by inserting the suggested rate into the data section of the Rate message response. If the HTU-C cannot support the rate suggested by the HTU-R, it will pick the nearest lower rate from one of its rate tables. This negotiation continues until both sides agree on a rate. When that occurs, each side begins training at the agreed rate.

The data field determines the desired data rate to train the modems. The data field formula is consistent with the _DSL_DATA_RATE (0x0E) API command.

Table 7-12 Select Rate Data Field Definition

Bit 15:10 Reserved	Bit 9:0 Data Rate
0x00	Value = Data Rate / 8 k

7.2.2.9 NAK Portion

A NAK notifies the HTU-C that the HTU-R did not receive the correct opcode. A NAK causes the HTU-C to resend the phase it was in.

The data field determines the NAK error code.

Table 7-13 NAK Data Field Definition

NAK Value	Description
0xFFFF	Invalid message
Rest	Reserved for future use

7.2.3 Line Probe Protocol

Line probing is optional and is required only in rate-adaptive applications. The Line Probe Enable field in the _DSL_PREACTIVATION_CFG (0x0F) API command is used to set the line probe state. The line probe sends a series of broadband signals to allow the modems to characterize the channel (noise, attenuation, bridge taps, etc). This information is used to determine the optimal data rate. This protocol is identical for both HTU-C towards HTU-R and HTU-R towards HTU-C.

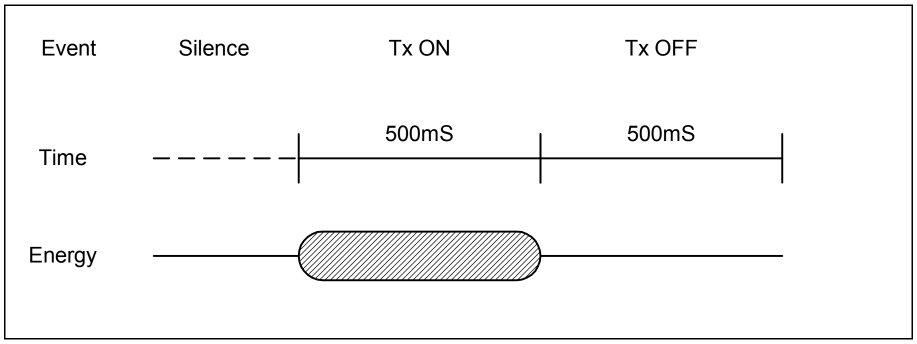
When line probing is enabled, the ZipWirePlus firmware performs the line probing functions, transmission and reception of the signals, and determines the optimal data rate.

After each modem measures its respective parameters, the HTU-R sends its results back to the HTU-C via the Probe Response message. The HTU-C uses both the HTU-C and HTU-R information to determine the optimal data rate.

While transmitting, the local modem measures the Signal Level Meter (SLM) to get an estimate of the hybrid's ability to cancel the echo (same concept as ERLE). During silence, the local modem remains idle.

While transmitting, the far-end modem measures the SLM to get an estimate of the attenuation. During silence, the far-end modem measures the SLM to get an estimate of the noise.

Figure 7-6 Auto Baud Pre-Activation Line Probe



7.2.4 Remote Configuration

The following list specifies several aspects that are configured via the Auto Baud sessions:

- Layer 1 and 2 information
- DSL data rate

7.2.4.1 Layer 1/2 Information

The Layer 1 and 2 information is exchanged via the discovery message. The information is exchanged between both the HTU-C and HTU-R. The ZipWirePlus firmware does not perform any action of the Layer 1 and 2 information; the information is received/forwarded from/to the host processor and it is up to the host processor to take any action. See Section 7.2.2.6 for the layer 1 and 2 information description.

The _DSL_PREACT_USER_INFO (0x14) API command is used to set the Layer 1 and 2 information. The _DSL_PREACT_GET_FE_CAPS (0x88) API command returns the far-end's Layer 1 and 2 information to the host processor.

7.2.4.2 DSL Data Rate

The Auto Baud Send Rate message determines the final DSL data rate to train the modems. In the SDSL (2B1Q) applications that typically use Auto Baud, there is no additional framer overhead. This must be taken into consideration when specifying the desired data rate. The final data rate is determined from one of the following options:

- _DSL_MULTI_RATE_CONFIG (0x1B) API command–used in fixed data rate applications (line probe disabled)
- _DSL_PREACT_RATE_LIST (0x15) and Line Probe results–when line probing is enabled

Data Rate: Fixed-Rate Applications

In fixed-rate applications, the _DSL_MULTI_RATE_CONFIG (0x1B) API command is used to determine the final data rate. This provides a consistency with existing API commands and when executing loopbacks and test modes. A fixed-rate application is determined by disabling the line probe functionality.

Or alternatively the _DSL_PREACT_RATE_LIST (0x15) API command can be used to determine the final data rate with the "Line Probe or Fixed" field being set to 0x01. Only a single rate can be specified in the _DSL_PREACT_RATE_LIST (0x15) API command parameter for the fixed-rate Application.

Data Rate Table: Rate-Adaptive Applications

In rate-adaptive applications, the HTU-C uses the line probing information to determine the theoretical capabilities of the channel. The driver then cross-references the line probe data rate with a customer defined data rate table, and selects the highest possible data rate from the table.

The HTU-C uses the _DSL_PREACT_LIST (0x15) API command to specify the list of supported rates. The data rate source byte in the _DSL_PREACTIVATION_CFG (0x0F) API command must be set to the List option.

The HTU-R must set the data rate source byte in the _DSL_PREACTIVATION_CFG (0x0F) API command must be set to the All option. This allows the HTU-R to be a dumb terminal.

Data Rate Source: List Option

Use the _DSL_PREACT_RATE_LIST (0x15) API command to specify the list of supported rates. This command is only valid if the data rate source byte is set to the list option. The number of entries in the list can be from 1-32. Each entry requires 2 bytes (N, i), which is consistent with the _DSL_MULTI_RATE_CONFIG (0x1B) API command (rate = N x 64 k + i x 8 k). The data rate entries must be in increasing order (lowest rate first and highest rate last).

7.2.5 Auto Baud API Command Summary

Table 7-14 lists a summary of the Auto Baud API commands.

Table 7-14 Auto Baud API Commands Summary

Command	Control Status	Opcode	Description
_DSL_PREACTIVATION_CFG	Control	0x0F	Configures the DSL pre-activation settings.
_DSL_PREACT_USER_INFO	Control	0x14	Configures the pre-activation user information.
_DSL_PREACT_RATE_LIST	Control	0x15	Specifies the list of supported payload data rates.
_DSL_PREACT_GET_FE_CAPS	Status	0x88	Returns the far-end capabilities information exchanged during the pre-activation.
_DSL_PREACT_GET_OPT_DATA_RATE	Status	0x89	Returns optimal data rate determined during line probe.

7.3 HDSL2 (OPTIS) Pre-Activation

HDSL2 (OPTIS) pre-activation provides a simple ping to determine if the far end is present and also communicates with the transmit power back-off control. The HDSL2 (OPTIS) pre-activation only supports the HDSL2 (OPTIS) mode.

The ZipWirePlus operational code handles HDSL2 pre-activation. No configuration is required by the host processor.

8.0 Performance Monitoring

The ZipWirePlus device supports several performance monitoring error counters. These error counters are updated based on overhead bits defined by various standards and DSL line conditions. Section 8.1 describes the overhead bits defined in G.shdsl, HDSL2 and HDSL1 standards.

8.1 Overhead Processing Detailed Description

DSL overhead consists of the following:

- Sync Word–used for DSL frame alignment
- Stuff Bits–used to compensate for different PCM and DSL clock offsets
- CRC-6–used for monitoring bit errors
- EOC–used as a management channel
- Indicator Bits–used to indicate critical status that can not tolerate latency

These components are found in the various DSL frame structures (G.shdsl, HDSL2, HDSL1, IDSL etc). The exact number of bits and definition vary from frame structure to frame structure. The ZipWirePlus firmware handles the differences between the frame structure as well as processes the low-level, real-time events. The ZipWirePlus interfaces with the host processor via API commands whenever the ZipWirePlus firmware cannot determine the appropriate action.

This section provides a detailed description of the overhead processing. Table 8-1 lists the number of overhead bits (per frame) for each of the frame structures.

Table 8-1 DSL Frame Structure Bit Allocation

Frame Structure	# Sync Word Bits	# Stuff Bits	# CRC Bits	# EOC Bits	# Ind Bits
G.shdsl	14	4	6	20	4
HDSL2	10	4	6	24	4
HDSL1	14	4	6	13	13
Framer Transparent	0	0	0	0	0
IDSL (12 ms superframe)	144	0	12	24	12

8.1.1 Sync Word and Stuff Bits

The sync word is used to establish and maintain DSL frame alignment. The sync word is based on the standards. The ZipWirePlus firmware determines the appropriate sync word based on the frame structure as shown in Table 8-2. The frame structure and DSL configuration API commands, as well as the physical channel number are used to determine the proper sync word. The sync word is the same for both the HTU-C to HTU-R direction and HTU-R to HTU-C direction. The least significant bit is transmitted first (right to left). The G.shdsl standard defines a 14-bit sync word, HDSL2 standards define a 10-bit sync word, IDSL standard defines a 18-bit sync word and HDSL1 standard defines a 14-bit sync word.

The stuff bits are used to compensate for phase differences between the PCM and DSL clock domains. This is also referred to plesiosynchronous mode. In synchronous mode, the DSL frame always contains two stuff bits.

Table 8-2 DSL Frame Structure Sync Word and Stuff Bits

Frame Structure	Sync Word Value	Stuff Bit Value
G.shdsl–ITU (1)	(00110000111111)	(1111)
G.shdsl–ETSI	(00110000111111)	(1111)
HDSL2 (1)	(1011110000)	(1011)
HDSL1–ANSI	(10101000001000)	(1010)
HDSL1–ETSI	(00010000010101)	(1010)
Framer Transparent	N/A	N/A
IDSL	(100010001000100010)	N/A

1. The G.shdsl (ITU) and HDSL2 standards allow for any sync word; the far-end determines what the near-end transmit sync word should be. The G.hs (G.shdsl) or Activation Exchange (HDSL2) protocols handle the sync word exchange.

8.1.2 CRC

The CRC-6 is used to indicate one (or more) bit errors within the DSL frame for G.shdsl, HDSL1 and HDSL2 modes.

The CRC-12 is used to indicate one (or more) bit errors within the DSL frame for IDSL modes.

In the transmit direction, the ZipWirePlus device automatically calculates and inserts the expected CRC value into the DSL frame. The _DSL_INJECT_CRC_ERROR (0x41) is used to insert CRC errors. This allows the system to test the far-end error handling.

In the receive direction, the ZipWirePlus device automatically calculates the payload CRC value and compares it to the expected CRC value. The ZipWirePlus firmware maintains error counters. Use the _DSL_HDSL_PERF_ERR_CTRS (0x9E) to read the number of CRC errors.

8.1.2.1 CRC Tap

The DSL CRC tap is based on the appropriate standards, as listed in Table 8-3. The frame structure API command is used to determine the proper tap selection. The CRC polynomial is the same for both the HTU-C to HTU-R direction and the HTU-R to HTU-C direction.

Table 8-3 CRC Tap

Standard	CRC Tap
G.shdsl	CRC6: $x^6 + x + 1$
HDSL2	CRC6: $x^6 + x + 1$
HDSL1	CRC6: $x^6 + x + 1$
IDSL	CRC12: $x^{12} + x^{11} + x^3 + x^2 + x + 1$

8.1.3 Indicator Bits

The indicator bits are used to indicate critical status that can not tolerate latency. Typically, the indicator bits are set to 1 to denote a condition that does not exist and set to 0 to denote a condition that does exist.

The _DSL_WRITE_IND_BITS (0x60) allows the customer to determine the outgoing indicator bits. The mechanism for the host to extract the indictors varies from bit to bit.

The SEGA, SEGD, and LOSD error count are extracted from the _DSL_HDSL_PERF_ERR_CTRS (0x9E) API command.

The Power Status (Dying Gasp) is treated specially. When the driver detects this condition, an Unsolicited Interrupt is generated and sent to the host.

NOTE: The bit # represents the relative order within the DSL frame but not the exact bit location. The ZipWirePlus firmware automatically handles inserting the bits into the appropriate location within the frame based on the frame structure.

8.1.3.1 G.shdsl Indicator Bit Definitions

The G.shdsl frame structure contains four indicator bits.

Table 8-4 G.shdsl Indicator Bit Definitions

Bit #	Name	Description
0	LOSD	Loss of signal (application interface)
1	SEGA	Segment anomaly (only applicable when regenerator present)
2	PS	Power status (dying gasp)
3	SEGD	Segment defect (only applicable when regenerator present)

8.1.3.2 HDSL2 Indicator Bit Definitions

The HDSL2 frame structure contains four indicator bits.

Table 8-5 HDSL2 Indicator Bit Definitions

Bit #	Name	Description
0	LOSD	Loss of signal (application interface)
1	UIB	Unspecified Indicator Bit
2	SEGA	Segment anomaly (only applicable when regenerator present)
3	SEGD	Segment defect (only applicable when regenerator present)

8.1.3.3 IDSL Indicator Bit Definitions

The IDSL frame structure contains the following indicator bits

Table 8-6 HDSL1 Indicator Bit Definitions- LT to NT direction

Name	Description
act	startup bit (set = 1 during startup)
dea	turn-off bit (set = 0 during turn-off)
febe	Far-end block error bit (set = 0 for errored superframe)
uoa	U-only activation bit (optional, set = 1 to activate S/T)
aib	alarm indication bit (set = 0 to indicate interruption)

Table 8-7 HDSL1 Indicator Bit Definitions- NT to LT direction

Name	Description
act	startup bit (set = 1 during startup)
ps[1]	Power status bit #1 (set = 0 to indicate Power problems)
febe	Far-end block error bit (set = 0 for errored superframe)
ps[2]	Power status bit #2 (set = 0 to indicate Power problems)
ntm	NT in test mode bit (set = 0 to indicate test mode)
cso	Cold-start-only bit (set = 1 to indicate cold-start-only)
sai	S-activation indication bit (set = 1 for S/T activity)

8.1.3.4 HDSL1 Indicator Bit Definitions

The HDSL1 frame structure contains 13 indicator bits.

Table 8-8 HDSL1 Indicator Bit Definitions

Bit #	Name	Description
0	LOSD	Loss of signal (application interface)
1	FEBE	Far-end block error
2	PS1	Power status bit #1
3	PS2	Power status bit #2
4	BPV	Bipolar violation (application interface)
5	HRP	DSL repeater (regenerator) present
6	RRBE	Repeater remote block error
7	RCBE	Repeater central block error
8	REGA	Repeater alarm
9	RTA	Remote terminal alarm
10	RTR	Ready to receive
11	UIB	Unspecified Indicator bit #1
12	UIB	Unspecified Indicator bit #2

8.1.3.5 Framer Transparent Indicator Bit Definitions

The Framer Transparent frame structure contains no overhead bits.

8.1.4 System And Performance Monitoring API Commands

Table 8-9 lists the API commands used for system and performance monitoring.

Table 8-9 System And Performance Monitoring API Command Summary

COMMAND	OPCODE	DESCRIPTION
_DSL_FAR_END_ATTEN	0x82	Gets the far-end signal attenuation.
_DSL_NOISE_MARGIN	0x83	Gets the receiver noise margin (NMR).
_DSL_POWER_BACK_OFF_RESULT	0x94	Gets the transmit and receive power back-off result.
_DSL_SYSTEM_PERF_ERR_CTRS	0xA2	Queries the system performance error counters.
_DSL_HDSL_PERF_ERR_CTRS	0x9E	Queries the DSL performance error counters.
_ATM_PHY_PERF_ERR_CTRS	0xB9	Queries the ATM PHY performance error counters.
_ATM_PHY_CELL_CTRS	0xBA	Queries the ATM PHY cell error counters.
_DSL_TIME	0x9D	Gets the available seconds and total seconds since power-on or reset.

8.2 Embedded Operation Channel (EOC)

This section provides an overview of the Embedded Operations Channel (EOC) across the ZipWirePlus family of devices.

8.2.1 Feature Overview

- Host processor handles all message processing.
- The internal processor handles all low-level HDLC protocol layer, e.g., flag insertion, data transparency, frame error checking (FCS), and frame alignment (G.shdsl).
- Dynamic queuing supports handling of multiple transmit and receive messages.
- Host processor uses polling or unsolicited interrupts to retrieve EOC messages.
- Supports G.shdsl, HDSL2, and HDSL1 EOC standards. Please note that the HDSL1 is supported only on M28950 device.

8.2.2 EOC General Overview

The EOC provides a communication channel between the ZipWirePlus terminal units. This allows the units to communicate configuration and status messages. The HTU-C is the master and initiates the EOC messages. The HTU-R can optionally support initiating a message. Both the HTU-C and HTU-R must respond to message requests.

There are two types of messages: requests and responses. A complete transaction consists of a request by the near-end followed by a response from the far-end. Table 8-10 shows the message types and message ID range for a request message and response message.

Table 8-10 EOC Message Types

Msg Type	Msg ID Range	Description
Request	0x00–0x7F	Used to configure the far end, query the far end for status, or command the far end to perform some task.
Response	0x80–0xFF	Used to acknowledge a request and provide status (or other information).

8.2.2.1 EOC Frame Format

The EOC channel carries messages in an HDLC-like format. The channel is treated as a stream of octets; all messages are an integral number of octets.

The frame format uses a compressed form of the HDLC header and is illustrated in Table 8-11. The destination address field is the least significant 4 bits of octet 1; the source address field occupies the most significant 4 bits of the same octet (the address field.) There is no control field. One or more sync octets (0x7E) are present between each frame. Inter-frame fill is accomplished by inserting sync octets as needed. The Information Field contains exactly one message as defined in Table 8-11. The maximum length of a frame shall be 75 octets, not including the sync pattern or any octets inserted for data transparency.

Table 8-11 EOC Frame Format

	Contents		
Octet #	MSB	LSB	Responsibility
	Sync pattern 0x7E		Internal processor
1	Source address Bits 7..4	Destination address Bits 3..0	Host processor is responsible for inserting the correct source and destination addresses. (The internal processor performs data transparency.)
2	Message ID		
	Message Content - Octet 1		
	…		
	Message Content - Octet N		
N+3	FCS octet 1		Internal Processor
N+4	FCS octet 2		
	Sync pattern 0x7E		

8.2.2.2 EOC Unit Addresses

Each unit uses one source and destination address when communicating with upstream units, and a separate independent source and destination address when communicating with downstream units. Each address has a value between 0x0 and 0xF, as listed in Table 8-12.

Table 8-12 EOC Device Addresses

Value (C Constant)	Device
0x00	Adjacent device
0x01 (_EOC_H2TUC)	HTU-C
0x02 (_EOC_H2TUR)	HTU-R

Value (C Constant)	Device
0x03 (_EOC_H2RU1)	Regenerator 1
0x04 (_EOC_H2RU2)	Regenerator 2
0x05 (_EOC_H2RU3)	Regenerator 3
0x06 (_EOC_H2RU4)	Regenerator 4
0x07 (_EOC_H2RU5)	Regenerator 5
0x08 (_EOC_H2RU6)	Regenerator 6
0x09 (_EOC_H2RU7)	Regenerator 7
0x0A (_EOC_H2RU8)	Regenerator 8
0x0B- 0x0E	Reserved
0x0F (_EOC_H2BCAST)	Broadcast

8.2.2.3 EOC Message IDs

Messages 0-127 represent request messages. Messages 128-255 represent messages sent in response to request messages. Each request message is acknowledged with the corresponding response. Request/Response Message IDs usually differ by an offset of 128.

8.2.2.4 EOC G.shdsl, HDSL1, and HDSL2 Support

The ZipWirePlus device and firmware handle the HDLC protocol differences between G.shdsl, HDSL1, and HDSL2 frame formats. The host processor handles any message differences.

The G.shdsl standard defines 2 ½ bytes per frame (5 bytes per 2 frames); in other modes, the standard defines an integer byte count per frame as shown in Table 8-13. In G.shdsl mode, a flag sync detector ensures receiving the flag pattern (0x7E) before processing any messages.

Table 8-13 EOC - Number of Bytes per Frame

Mode	Bytes per Frame Count
G.shdsl	5 bytes per 2 frames
HDSL1	13 bytes per 8 frames
HDSL2	3 bytes per 1 frame

8.2.2.5 EOC 4-Wire Support

In 4-Wire mode, the EOC messages are redundantly sent on each pair (loop). The host processor handles any message differences.

8.2.3 EOC Implementation Details

This section provides the EOC implementation details.

8.2.3.1 EOC Transmit

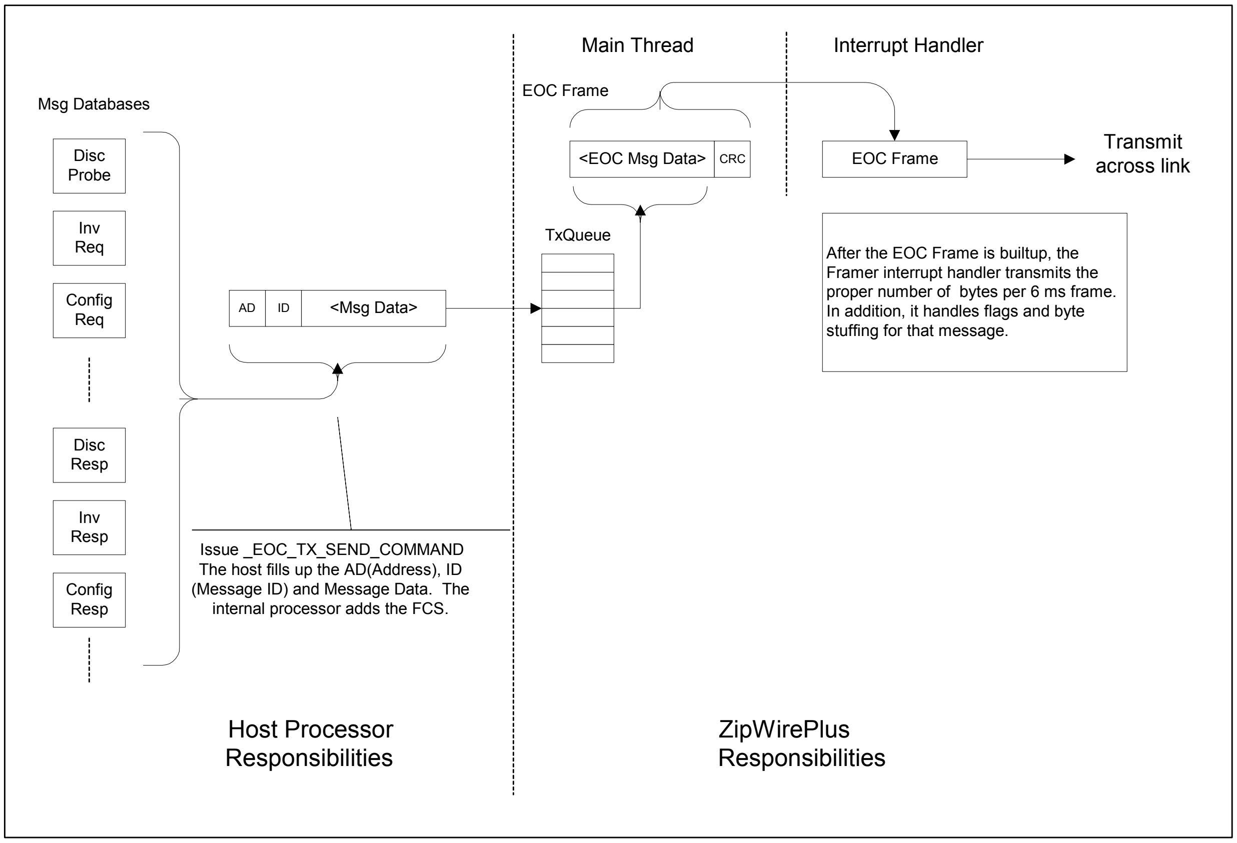
Figure 8-1 illustrates the EOC transmit implementation. The host processor builds up the appropriate request or response message (excluding the FCS) and issues the _EOC_TX_SEND_COMMAND (0xB0) API command. The host then processes the API acknowledge status to determine if the message was successfully placed in the EOC transmit queue. The EOC transmit queue is 384 bytes.

The 8051 returns the _ACK_NO_RESULT to denote the EOC transmit queue is full; the host processor then handles any retries. If the _EOC_TX_SEND_COMMAND (0xB0) is successful, the internal processor calculates the FCS, inserts any data transparency bytes, and transmits the EOC frame across the EOC channel. Sending an EOC message out the EOC channel does not guarantee that the remote end received the EOC message correctly. The host is responsible for verifying that messages are correctly transmitted to the remote end.

The host can get the status of these messages using the _EOC_TX_GET_MSG_STATUS (0xB2) API command and delete any messages from the queue using the _EOC_TX_DELETE_MSG (0xB3) API command.

The internal processor continuously transmits the sync octets (flag-0x7E) whenever there is no EOC message to send.

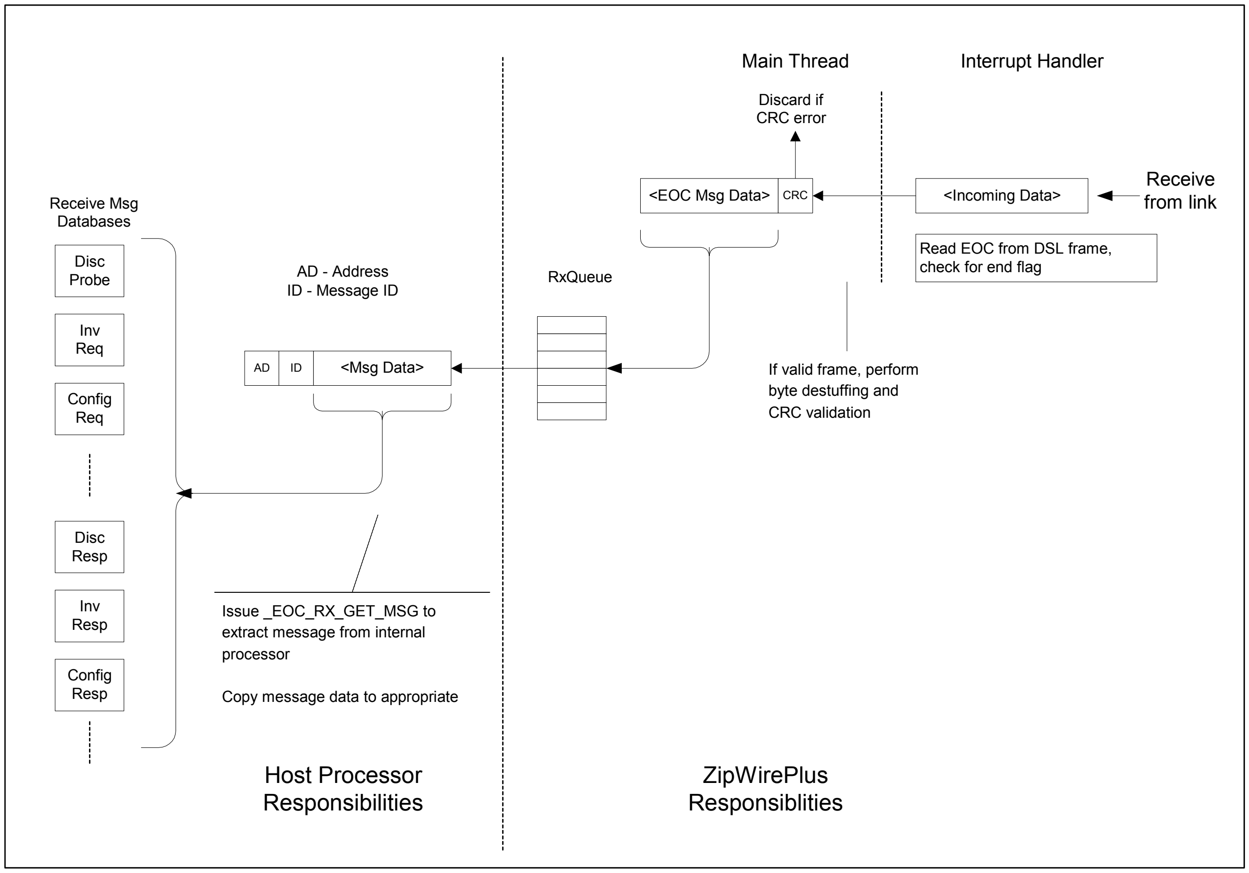
Figure 8-1 EOC Transmit Implementation Details


Main Thread
Interrupt Handler
Msg Databases
EOC Frame
Disc Probe
Inv Req
Config Req
Disc Resp
Inv Resp
Config Resp
<EOC Msg Data>
CRC
EOC Frame
Transmit across link
TxQueue
AD
ID
<Msg Data>
After the EOC Frame is builtup, the Framer interrupt handler transmits the proper number of bytes per 6 ms frame. In addition, it handles flags and byte stuffing for that message.
Issue _EOC_TX_SEND_COMMAND The host fills up the AD(Address), ID (Message ID) and Message Data. The internal processor adds the FCS.
Host Processor Responsibilities
ZipWirePlus Responsibilities

8.2.3.2 EOC Receive

Figure 8-2 illustrates the EOC receive implementation. The DSL framer receives the raw EOC bytes every 6 ms DSL frame. The interrupt handler examines the incoming EOC data for a non-flag character to mark the beginning of the EOC frame. The next flag character marks the end of the EOC frame. If the data is a valid frame, the internal processor will undo the data transparency and validate the FCS. If the FCS is valid, an EOC message is inserted in the EOC receive queue. The EOC receive queue is 384 bytes. The host processor can then retrieve the EOC message via an unsolicited interrupt or polling. When using unsolicited interrupts, the message is automatically placed in the API Out mailbox. The driver generates one interrupt per message. When using polling, the host issues the _EOC_RX_GET_MSG (0xB1) command to retrieve the EOC messages. The _EOC_RX_GET_STATS (0xAE) command returns statistics regarding the EOC receiver.

Figure 8-2 EOC Receive Implementation Details


Main Thread
Interrupt Handler
Discard if CRC error
<EOC Msg Data>
CRC
<Incoming Data>
Receive from link
Read EOC from DSL frame, check for end flag
If valid frame, perform byte destuffing and CRC validation
RxQueue
Receive Msg Databases
Disc Probe
Inv Req
Config Req
Disc Resp
Inv Resp
Config Resp
AD - Address
ID - Message ID
AD
ID
<Msg Data>
Issue _EOC_RX_GET_MSG to extract message from internal processor
Copy message data to appropriate
Host Processor Responsibilities
ZipWirePlus Responsiblities

8.2.3.3 EOC - Host Interaction

The host processor uses polling to determine the EOC status. Table 8-14 lists the EOC transmit message sequence of events.

Table 8-14 Host Processor EOC Transmit Message Events

Processor	Action
Host	Build message data (excluding FCS and data transparency).
Host	Issue _EOC_TX_SEND_COMMAND (0xB0) API command.
Host and 8051	Acknowledge API command. The 8051 returns _ACK_NO_RESULT to denote the EOC transmit queue is full.
8051	Transmit message across EOC (may require several 6 ms DSL frames).
Host	The host can verify (via polling) if a message was sent by checking on the status of these messages using the _EOC_TX_GET_MSG_STATUS (0xB2) API command. An acknowledge status of _EOC_STATUS_NA indicates that the message was sent over the EOC channel and/or deleted from the queue.

Table 8-15 Host Processor EOC Receive Message Events

Processor	Action
8051	Receive EOC message

Processor	Action
Host	Host polls for any EOC messages by issuing the _EOC_RX_GET_MSG (0xB1) API command.

8.2.4 EOC API Command Summary

Table 8-16 lists a summary of the EOC API commands.

Table 8-16 EOC API Commands Summary

Command	Control Status	Opcode	Description
_EOC_RESET	Control	0x4D	Reset the EOC queues
_EOC_RX_GET_STATS	Status	0xAE	Gets EOC statistics (error counters)
_EOC_TX_SEND_COMMAND	Status	0xB0	Initiate a transmission of an EOC message
_EOC_RX_GET_MSG	Status	0xB1	Get next available received EOC message
_EOC_TX_GET_MSG_STATUS	Status	0xB2	Get status of a pending EOC transmission
_EOC_TX_DELETE_MSG	Status	0xB3	Delete a pending EOC transmission.

9.0 Internal E1 Framer

9.1 Overview

This chapter describes the internal E1 framer software feature. This feature is available on the M28947 device only.

9.1.1 Distinguishing Features

The following provides a list of the E1 Framer features. The list is broken into the ZipWirePlus and host processor responsibilities. The software can perform the E1 Framer functionality in both transmit and receive directions.

ZipWirePlus Responsibilities

- Align E1 frame in DSL Frame
- Automatic Multiframe Alignment (or use external sync reference)
- Output 2 millisecond Multiframe sync pulse in receive direction
- Automatic FAS & MFAS generation when the DSL Link is down
- Automatic CRC detection and insertion according to G.706
- Automatic CRC-4 error indication bits (E bits)
- Transmit remote alarm indication bits (A bits) pattern
- Process spare bits (Sa4, Sa5, Sa6, Sa7, Sa8) up to message level
- Timeslot 16 processing includes CAS Multiframe Alignment
- Generating E1 Framer alarms
- Alarm Indication Signal (AIS)
- Y-bit
- RAI (A-bit)
- Loss Multi Frame Alignment (LMFA)
- Timeslot 16 Alarm Indication Signal (TS16AIS.)
- Accumulating CERR and FEBE Error Counters.

Host Processor Responsibilities

- Enabling/Disabling the E1 Framer
- Enabling Transmit And Receive Self Alignment Mode
- Optionally Process Overhead bits in Manual Mode
- Determining AIS condition and requesting the ZipWirePlus device to send framed AIS.
- Generating E1 RFSLIP and RUSLIP alarms
- Configuring the ZipWirePlus device for Fractional E1 with selectable insertion of programmable idle codes for fractional operation and/or channel banking

9.1.2 E1 Framer Overview

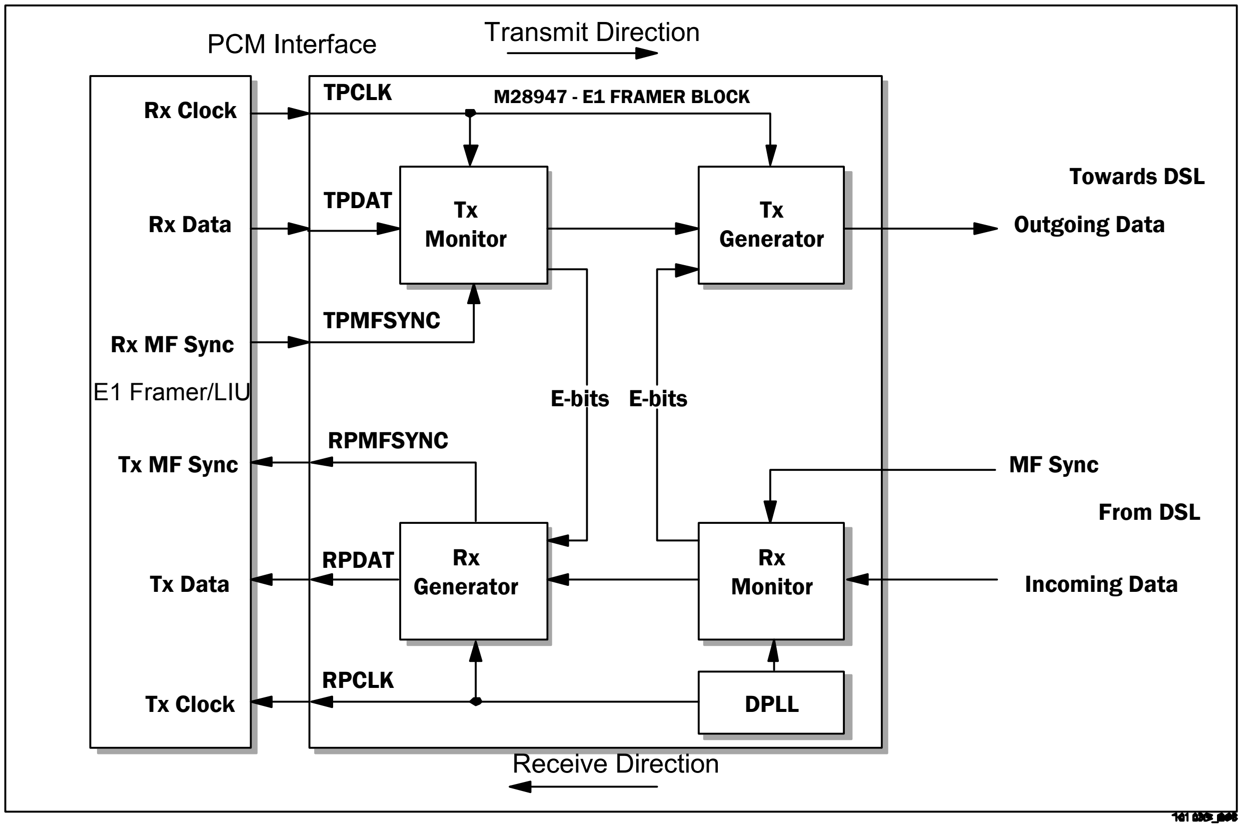
Figure 9-1 illustrates the E1 Framer data path and terminology. The E1 Framer functionality monitors and manipulates the overhead bits found in timeslot 0 of the E1 frame including CRC-4, E-bits, A-bits, and spare (Sa) bits. In addition, the block can either automatically lock onto the incoming multiframe boundary or accept the multiframe from an external source.

The E1 Framer functionality can be performed in both the transmit and receive directions. The transmit direction is defined as the data path from the PCM interface towards the DSL interface. The receive direction is defined as the data path from the DSL interface towards the PCM interface. Each direction consists of a monitor and generator block. The E1 Framer functionality is identical in each direction, but each direction is independently controlled.

In general, there are three possible ways to control the various overhead bits: transparent, automatic, and manual. Refer to each overhead bit definition to determine which modes apply.

- Transparent—the generator block outputs the overhead bits unaltered from the monitor block.
- Automatic—the generator block will automatically determine and output the appropriate overhead bits value. The overhead bits are updated every multiframe sync.
- Manual—the generator block outputs the overhead bits based on an API command received from the host processor. The generator block will continuously output the new value starting on the subsequent multiframe sync.

Figure 9-1 E1 Framer Overview



9.1.2.1 Multiframe Alignment

Table 9-1 lists the ITU–T CEPT Frame Format Time Slot 0-Bit definition for the E1 multiframe. The E1 multiframe is split into 2 sub-multiframes. The CRC-4 is calculated on a sub-multiframe boundary and inserted into the next sub-multiframe boundary.

The grayed bits define the Multiframe alignment pattern.

Table 9-1 ITU-T CEPT Multiframe Structure

Sub Multiframe	Frame #	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
1	0	C1	0	0	1	1	0	1	1
	1	0	1	A	Sa4	Sa5	Sa6	Sa7	Sa8
	2	C2	0	0	1	1	0	1	1
	3	0	1	A	Sa4	Sa5	Sa6	Sa7	Sa8
	4	C3	0	0	1	1	0	1	1
	5	1	1	A	Sa4	Sa5	Sa6	Sa7	Sa8
	6	C4	0	0	1	1	0	1	1
	7	0	1	A	Sa4	Sa5	Sa6	Sa7	Sa8
2	8	C1	0	0	1	1	0	1	1
	9	1	1	A	Sa4	Sa5	Sa6	Sa7	Sa8
	10	C2	0	0	1	1	0	1	1
	11	1	1	A	Sa4	Sa5	Sa6	Sa7	Sa8
	12	C3	0	0	1	1	0	1	1
	13	E1	1	A	Sa4	Sa5	Sa6	Sa7	Sa8
	14	C4	0	0	1	1	0	1	1
	15	E2	1	A	Sa4	Sa5	Sa6	Sa7	Sa8

NOTE 1: E1, E2 = CRC4 error indication bits (SMF block with error: E = 0, SMF block without error: E=1)
NOTE 2: Sa4 -Sa8 = Spare bits
NOTE 3: C1-C4 = CRC-4 bits
NOTE 4: A = RAI
Please see ETS 300 011-1: March 1998, Section 5.5.4.1 and Table 4

9.1.2.2 Transmit Direction

For applications, where the ZipWirePlus device is connected to an external E1 Framer(which provides the 2 ms multiframe signal), the internal Transmit Sync Detector can be disabled. In this configuration TPMFSYNC input is used to align to the E1 multiframe.

For applications, where the external multiframe does not exist, the internal Transmit Sync Detector should be enabled. This configuration is done with the E1_PRA_CONFIG(0x44) API command.

9.1.2.3 Receive Direction

The Receive Sync Detector should be enabled only when the E1 data in the DSL Frame is not multiframe aligned in the DSL frame (Unaligned D2048S mode). This scenario is very unlikely since in D2048S mode that E1 Framer functionality must be used in the Transmit path at the Far end. Therefore the E1 multiframe must be aligned in the DSL frame.

9.1.2.4 Loss of Multiframe Alignment

Once the Transmit Sync Detector has been enabled, the M28947 continuously searches for multiframe alignment sync pattern. If multiframe alignment sync pattern cannot be found, it declares loss of CRC-4 multiframe alignment state. During a loss of CRC-4 multiframe alignment state:

The CRC-4 Error Counter is halted.

The CRC-4 Error Monitoring operation for errored seconds and severely errored seconds is halted.

The received E-bit monitoring is halted.

All receive Sa6 byte monitoring and counters are halted.

Optionally, Remote Alarm bit is set to 1 and may be automatically transmitted to the line.

Optionally E-bit is set to zero and may be automatically transmitted to the line. _E1_PRA_ALARM_STATUS (0xC6) API command shall be used to monitor LMFA Alarm as defined in Section 15.16.11.

9.1.3 CRC-4 Processing

CRC-4 Generator

The CRC-4 generator can be configured to transparent or automatic mode. In transparent mode, the CRC-4 is output unaltered from the monitor block. In automatic mode, the CRC-4 is re-calculated based on the E1 data stream. The CRC-4 is calculated on a sub-multiframe boundary and inserted into the next sub-multiframe boundary.

CRC-4 Monitoring

The CRC-4 monitoring block computes the CRC on the incoming E1 data stream and checks it with the received CRC-4. The monitoring block uses the E-bit to convey the result of the CRC check. Please refer to Section 9.1.4 for details.

In all cases, the monitor block maintains a read/clear 16-bit CRC-4 error counter. The error counter wraps around on full count. The CRC-4 error counter is always available when E1 Framer Functionality is enabled.

9.1.4 CRC-4 Error Indication Bit (E-bit) Processing

The CRC-4 Error indication bits are used to indicate to the far-end unit that a CRC-4 error was detected in the incoming data stream. The E1 and E2 error indicator bits indicate the status of the previously received multiframe CRC-4.

The E1 bit indicates the previous sub-multiframe 1 status while the E2 bit indicates the previous sub-multiframe 2 status. The E bit will be set to 1 when no CRC-4 error is detected and will be set to 0 when a CRC-4 error is detected. Both E bits will be set to 0 until both basic and multiframe alignment are established.

9.1.4.1 CRC-4 Error Bit (E-bit) Generator

The E-bit generator can be configured to transparent, manual, or automatic mode.

In transparent mode, the E-bit is generated unaltered from the monitor block in the same direction. For example, the transmit direction generator E-bit will match the transmit direction monitor's E-bit.

In manual mode, the E-bit is determined from the host processor via an API command. The _E1_PRA_TX_GEN_VALUES (0x45) API command is used to configure the E-bit in the transmit direction. The _E1_PRA_RX_GEN_VALUES (0x46) API command is used to configure the E-bit in the receive direction.

In automatic mode, the E-bit is determined based on the opposite direction monitor's CRC-4 result and output on the subsequent frame. For example, the transmit direction generator E-bit will be set to 0 if the receive direction's monitor detected a CRC-4 error.

Table 9-2 Automatic Mode E-bit Examples

Monitor's CRC-4 Check	Generator's E-Bit Value
Error	0
No error	1

9.1.4.2 CRC-4 Error Bit (E-bit) Monitor

The E-bit monitor block checks the value of the E-bits regardless of mode and CRC-4 error state.

In all cases, the monitor block maintains a read/clear 16-bit E-bit error counter. The error counter wraps around on full count. The E-bit error counter is always available when E1 Framer Functionality is enabled. The _E1_PRA_ERROR_CTRS (0xC4) API command is used to read the TX/RX error counters.

9.1.5 Remote Alarm Bit (A-bit) Processing

The Remote Alarm Bit (A) indication bits are used to indicate an alarm condition to the far-end unit. The A-bit will be set to 0 in normal operation and will be set to 1 in an alarm condition.

The A-bit generator and monitor can be configured to transparent or manual mode. In transparent mode, the generator outputs the A-bit value unaltered from the monitor block. In manual mode, the generator A-bit value is determined from the host processor via an API command. The monitor block will notify the host processor whenever there is a change in the received A-bit pattern and all 8 A-bits are identical.

The A-bit monitored value is available in all modes.

_E1_PRA_ALARM_STATUS (0xC6) API command shall be used to monitor RAI Alarm as defined in Section 15.16.11.

9.1.6 Spare Bit (Sa4 to Sa8) Processing

The Spare Bits (Sa) are provided for user-defined messages. For each Sa byte, a single control bit controls the Sa generator and monitor modes for each direction.

9.1.6.1 Sa4, Sa5, Sa7, Sa8 Processing

The Spare Bits (Sa) are provided for user-defined messages. The ZipWirePlus device provides independent configuration control for the Sa bytes.

For each multiframe, all 8 bits of a particular Sa4 and Sa5 bytes are made available to the host application. However for the Sa7 and Sa8 bytes only the lower nibble is made available to the host application. The higher nibble of the byte made available to the host is a duplicate of the lower nibble.

The Sa generator and monitor can be configured to transparent or manual mode. In transparent mode, the generator outputs the Sa value unaltered from the monitor block. In manual mode, the generator Sa value is determined from the host processor via an API command. The monitor block will notify the host processor whenever there is a change in the received Sa pattern.

The Sa bits monitored values are available in all modes.

Please note that the two nibbles comprising the Sa7 or Sa8 byte have the same value, i.e. 0x00, 0x11, 0x44. A value of 0x45 would be illegal.

9.1.6.2 Sa6 Processing

The Sa6 byte is split into 2 nibbles per multiframe. For each multiframe, all 8 bits of a particular Sa6 byte are made available to the host application.

The Sa6 generator can be configured to transparent, manual, or automatic mode. In transparent mode, the generator outputs the Sa6 value unaltered from the monitor block. In manual mode, the generator Sa6 value is determined from the host processor via an API command. The monitor block will notify the host processor whenever there is a change in the received Sa6 pattern and both nibbles match.

The ZipWirePlus firmware does not do any Sa6 automatic mode processing. The host code should configure the Sa6 bits (as well as the A-bits) depending on the network application and G.704 and ETS 300 233 standards.

9.1.7 Inject CRC-4 Errors

The ZipWirePlus provides the ability to inject a CRC-4 error in either the transmit and/or receive directions. This feature is useful in debugging (verifying) the CRC-4 monitoring and E-Bit processing capabilities.

9.1.8 TS16 Channel Associated Signaling (CAS) Multiframe Alignment

The M28947 supports CEPT channel associated signaling (CAS) signaling mode.

The channel associated signaling (CAS) mode utilizes time slot 16 of the FAS and not FAS(nFAS) frames. The CAS format is a multiframe consisting of 16 frames where frame 0 of the multiframe contains the multiframe alignment pattern of four zeros in bits 1 through 4. Table 9-3 illustrates the CAS multiframe structure of time slot 16. The entire content in timeslot 16 of Frame 0 of Signaling Multi-Frame is '0000XYXX'.

'Y' is for remote Signaling Multi-Frame alarm indication and 'X's are extra bits. The three X bits in frame 0 of the time slot 16 multiframes can be used as a 0.5 kbits/s data link to and from the remote end. The codeword 'ABCD' are the signaling bits for different timeslots.

Alignment of the transmitted line CAS multiframe to the CRC-4 multiframe is arbitrary. The receive line CAS alignment is defined by the received data.

Table 9-3 ITU CEPT Time Slot 16 Channel Associated Signaling multiframe Structure

	Frame Number	Bits							
		1	2	3	4	5	6	7	8
	0	0	0	0	0	X_0	Ym	X_1	X_2
Time Slot 16 Channel Associated Signaling Multiframe	1	A1	B1	C1	D1	A17	B17	C17	D17
	2	A2	B2	C2	D2	A18	B18	C18	D18
	3	A3	B3	C3	D3	A19	B19	C19	D19
	4	A4	B4	C4	D4	A20	B20	C20	D20
	5	A5	B5	C5	D5	A21	B21	C21	D21
	6	A6	B6	C6	D6	A22	B22	C22	D22
	7	A7	B7	C7	D7	A23	B23	C23	D23
	8	A8	B8	C8	D8	A24	B24	C24	D24
	9	A9	B9	C9	D9	A25	B25	C25	D25
	10	A10	B10	C10	D10	A26	B26	C26	D26
	11	A11	B11	C11	D11	A27	B27	C27	D27
	12	A12	B12	C12	D12	A28	B28	C28	D28
	13	A13	B13	C13	D13	A29	B29	C29	D29
	14	A14	B14	C14	D14	A30	B30	C30	D30
	15	A15	B15	C15	D15	A31	B31	C31	D31

Notes:
Frame 0 bits 1-4 define the time slot 16 multiframe alignments.
X0-X2 = time slot 16 spare bits.
Ym = time slot 16 remote multiframe alarm (RMA) bit (1 = alarm condition).

CEPT Loss of Time Slot 16 Multiframe Alignment

As defined in ITU Rec. G.732 Section 5.2, time slot 16 signaling multiframe is assumed lost when two consecutive time slot 16 multiframe 4-bit all-zero patterns (Grayed in Table 2) are received with an error. In addition, if all the bits in timeslot 16 are set to zero for two multiframes the time slot 16 multi-frames are assumed to be lost.

This state is reported through the _E1_PRA_ALARM_STATUS (0xC6) API command Section 15.16.11. Once CAS Multiframe Alignment is enabled using API command _E1_PRA_CONFIG (0x44), the ZipWirePlus will initiate a search for the time slot 16 multiframe alignments.

During a loss of time slot 16 multiframe alignment state:

1. The updating of the signaling data is halted.
2. The received remote multiframe alarm indication status bit is forced to the binary 0 state.
3. Optionally, the transmit framer can transmit the time slot 16 signaling remote multiframe alarm.
4. Optionally, the receive framer can transmit the alarm indication signal (AIS) in the system transmit time slot 16 data.

Y bit

It is the Remote Multi-Frame Alarm Indication bit. . It is updated at the first bit of the next CAS Signaling Multi-Frame and is held during out of CAS Signaling Multi-Frame state.

_E1_PRA_ALARM_STATUS (0xC6) API command shall be used to monitor Y bit Alarm as defined in Section 15.16.11.

CEPT Loss of Time Slot 16 Multiframe Alignment Recovery Algorithm

The time slot 16 multiframe alignment recovery algorithm is implemented as described in ITU Recommendation G.732 Section 5.2. The recommendation states that if a condition of assumed frame alignment has been achieved, time slot 16 multiframe alignment is deemed to have occurred when the 4-bit time slot 16 multiframe pattern of 0000 is found in time slot 16 for the first time and the preceding time slot 16 contained at least one bit in the binary 1 state.

9.1.9 Detection and Monitoring of Alarm Indication Signal (AIS)

AIS pattern consists of a continuous bit stream of ones.

A one indicates that the receive framer is currently receiving an AIS pattern in time slot 16 from its remote line end. This unframed AIS is detected only in CEPT mode when signaling is enabled. The detection of the alarm indication signal (AIS) in timeslot 16 is according to ITU-T G.775. This bit is set if the incoming TS16 contains less than 4 zeros in each of two consecutive TS16 multiframe periods. This bit will be cleared if two consecutive received CAS multiframe periods contains more than 3 zeros or the multiframe pattern was found in each of them. This bit will be cleared if TS0 synchronization is lost.

This is implemented using integrated DSL BER meter. _E1_PRA_ALARM_STATUS (0xC6) API command shall be used to monitor this Alarm as defined in Section 15.16.11.

9.1.10 E1 Framer Alarms

The _E1_PRA_ALARM_STATUS (0xC6) API is used to retrieve the status of alarms. This command has two output bytes. First byte indicates Receive Alarms status and Second byte indicate Transmit Alarm status.

9.1.10.1 RLOF Alarm

RLOF Alarm indicate Loss of FAS and NFAS pattern. Whenever FAS Pattern is not found in every 8th frame, ZipWirePlus indicate this alarm. This function is not fully standard compliant.

9.1.10.2 AIS Alarm

If received signal is all ones, this alarm will be set.

9.1.10.3 Y Alarm

This Alarm indicates ‘Y’ Bit in TS16 byte 0. If this bit is one, then this alarm is set, and if zero this alarm has been cleared.

9.1.10.4 RAI Alarm

This bit shows the status of NFAS “A” bit. This bit will be set when A-bit = 1 and be cleared when A-bit is zero.

9.1.10.5 LMFA Alarm

This bit indicates Loss of MultiFrame Alignment. When either of Rx and Tx Sync Detector goes out of sync this bit will be set.

9.1.10.6 TS16AIS Alarm

If all ones are received in timeslot 16, then this alarm will be set. If any byte contains zero, this bit will not be set

9.2 API Commands

The following API commands support E1 Framing Feature.

COMMAND	OPCODE	IN LENGTH	OUT LENGTH	DESCRIPTION
_E1_PRA_CONFIG	0x44	5	N/A	Configure the E1 Framer
_E1_PRA_TX_GEN_VALUES	0x45	Variable (6 or 22)	N/A	Program the transmit path manual values
_E1_PRA_RX_GEN_VALUES	0x46	Variable (6 or 22)	N/A	Program the receive path manual values
_E1_PRA_INJECT_CRC_ERROR	0x47	2	N/A	Inject continuous or a finite number of CRC-4 errors in the specified direction
_E1_PRA_TX_CODE	0x48	1	N/A	Transmit framed AIS, unframed AUXP or normal payload in the specified direction
_E1_PRA_TX_MON_CHANGE	0xC0	1	2	Indicate a change to a transmit path monitor value
_E1_PRA_RX_MON_CHANGE	0xC1	1	2	Indicate a change to a receive path monitor value
_E1_PRA_TX_MON_VALUES	0xC2	1	Variable (6 or 22)	Return the transmit path manual values
_E1_PRA_RX_MON_VALUES	0xC3	1	Variable (6 or 22)	Return the receive path manual values
_E1_PRA_ERROR_CTRS	0xC4	1	8	Return the E1 Framer error counters
_E1_PRA_MF_STAT	0xC5	1	1	Return the E1 Framer multiframe alignment status.
_E1_PRA_ALARM_STATUS	0xC6	1	2	Return the E1 Framer Alarm's status.

9.3 ZipWirePlus E1 Framer Implementation Details

9.3.1 E1 Framer and Host Port DPRAM

The E1 Framer functionality uses the Dual Port RAM to pass monitored overhead information from both Tx and Rx monitors to the host processor and to read what overhead information (set by the host processor through API's) should be generated in both directions. A bit field is used to indicate what monitored overhead changed in both monitored directions.

Figure 9-2 shows the structure for the Dual Port RAM.

Figure 9-2 Structure for Dual Port RAM for E1 Framer

Dual Port RAM for E1 Framer (offset 0x1e9)			
Byte	**Content**		**Description**
Tx Generator Values			
1	Tx Path Value		See the bit-field description below.
	Bit	**Content**	**Description**
	7:3	Reserved	
	2	A-bit Generator Value	1-bit field specifying the manual A-Bit value
	1	E2-bit Generator Value	1-bit field specifying the manual E2-Bit value
	0	E1-bit Generator Value	1-bit field specifying the manual E1-Bit value
2	Sa4 Value		1-byte field specifying the manual Sa4 value
3	Sa5 Value		1-byte field specifying the manual Sa5 value
4	Sa6 Value		1-byte field specifying the manual Sa6 value
5	Sa7 Value		1-byte field specifying the manual Sa7 value
6	Sa8 Value		1-byte field specifying the manual Sa8 value
7-22	TS16 byte 1-16		16 bytes of Timeslot 16 of Frame 0-15.
Rx Generator Values			
23	Rx Path Value		See the bit-field description below.
	Bit	**Content**	**Description**
	7:3	Reserved	
	2	A-bit Generator Value	1-bit field specifying the manual A-Bit value
	1	E2-bit Generator Value	1-bit field specifying the manual E2-Bit value
	0	E1-bit Generator Value	1-bit field specifying the manual E1-Bit value
24	Sa4 Value		1-byte field specifying the manual Sa4 value
25	Sa5 Value		1-byte field specifying the manual Sa5 value
26	Sa6 Value		1-byte field specifying the manual Sa6 value
27	Sa7 Value		1-byte field specifying the manual Sa7 value
28	Sa8 Value		1-byte field specifying the manual Sa8 value

Dual Port RAM for E1 Framer (offset 0x1e9)			
29-44	TS16 byte 1-16		16 bytes of Timeslot 16 of Frame 0-15.
Tx Monitor Values			
45	Tx Path Value		See the bit-field description below.
	Bit	**Content**	**Description**
	7:3	Reserved	
	2	A-bit Generator Value	1-bit field specifying the manual A-Bit value
	1	E2-bit Generator Value	1-bit field specifying the manual E2-Bit value
	0	E1-bit Generator Value	1-bit field specifying the manual E1-Bit value
46	Sa4 Value		1-byte field specifying the manual Sa4 value
47	Sa5 Value		1-byte field specifying the manual Sa5 value
48	Sa6 Value		1-byte field specifying the manual Sa6 value
49	Sa7 Value		1-byte field specifying the manual Sa7 value
50	Sa8 Value		1-byte field specifying the manual Sa8 value
51-66	TS16 byte 1-16		16 bytes of Timeslot 16 of Frame 0-15.
Rx Monitor Values			
67	Rx Path Value		See the bit-field description below.
	Bit	**Content**	**Description**
	7:3	Reserved	
	2	A-bit Generator Value	1-bit field specifying the manual A-Bit value
	1	E2-bit Generator Value	1-bit field specifying the manual E2-Bit value
	0	E1-bit Generator Value	1-bit field specifying the manual E1-Bit value
68	Sa4 Value		1-byte field specifying the manual Sa4 value
69	Sa5 Value		1-byte field specifying the manual Sa5 value
70	Sa6 Value		1-byte field specifying the manual Sa6 value
71	Sa7 Value		1-byte field specifying the manual Sa7 value
72	Sa8 Value		1-byte field specifying the manual Sa8 value

Dual Port RAM for E1 Framer (offset 0x1e9)			
73-88	TS16 byte 1-16		16 bytes of Timeslot 16 of Frame 0-15.
Tx Monitor Change			
89	Tx Path Change		See bit-field description below.
	Bit	**Content**	**Description**
	7	Sa8 Monitor Change	0 = No 1 = Yes
	6	Sa7 Monitor Change	0 = No 1 = Yes
	5	Sa6 Monitor Change	0 = No 1 = Yes
	4	Sa5 Monitor Change	0 = No 1 = Yes
	3	Sa4 Monitor Change	0 = No 1 = Yes
	2	A-bit Monitor Change	0 = No 1 = Yes
	1	E-bit Monitor Change	0 = No 1 = Yes
	0	MF Sync Status	0 = No 1 = Yes
90	Tx TS16 CAS Indication		See bit-field description below.
	Bit	**Content**	**Description**
	7-1	Reserved	
	0	Tx TS16 CAS Indication	1 = This bit indicate CAS Alignment has been detected 0 = This bit indicate CAS Alignment has not detected
Rx Monitor Change			
91	Rx Path Change		See bit-field description below.
	Bit	**Content**	**Description**
	7	Sa8 Monitor Change	0 = No 1 = Yes
	6	Sa7 Monitor Change	0 = No 1 = Yes
	5	Sa6 Monitor Change	0 = No 1 = Yes

Dual Port RAM for E1 Framer (offset 0x1e9)			
	4	Sa5 Monitor Change	0 = No 1 = Yes
	3	Sa4 Monitor Change	0 = No 1 = Yes
	2	A-bit Monitor Change	0 = No 1 = Yes
	1	E-bit Monitor Change	0 = No 1 = Yes
	0	MF Sync Status	0 = No 1 = Yes
92	Rx TS16 CAS Indication		See bit-field description below.
	Bit	**Content**	**Description**
	7-1	Reserved	
	0	Rx TS16 CAS Indication	1 = This bit indicate CAS Alignment has been detected 0 = This bit indicate CAS Alignment has not detected

9.4 Operational Behavior

This section provides information on the operational behavior of E1 Framer feature. Refer to Figure 9-3 for references to HTU-C and HTU-R, Rx generator, Tx generator, Rx monitor and Tx monitor.

9.4.1 Manual Mode

If any of the overhead bytes are set to manual mode in the generator direction (Tx or Rx), the associated CRC4 should be set to recalculate in the same direction to avoid CRC4 errors in the downstream or upstream network element.

For instance, if the Sa4 mode is set to manual in the Tx generator direction on HTU-C, the Tx generator CRC4 mode should be set to re-calculate on HTU-C to avoid CRC4 errors on the Rx monitor on HTU-R.

9.4.2 Error Gating

The overhead processing of Sa and A-bits are disabled, if a DSL error (currently only Loss of Sync Word) is received at the Rx monitor. It is enabled if the DSL error clears.

The overhead processing is disabled, if a CRC4 error is received at the Rx or Tx monitor. It is enabled if the CRC4 error clears.

9.4.3 E1 Framer Error Counters

The E1 Framer error counters, Rx and Tx CRC4 and E-bit, are read on clear.

9.4.4 AUXP

The unframed AUXP pattern (“10101010…”) is transmitted by the Rx generator or Tx generator as desired via the _E1_PRA_TX_CODE (0x48) API command. This API uses the PRBS generation functionality of the ZipWirePlus to transmit the AUXP pattern. Any previous PRBS generation configuration will be lost. The desired PRBS API sequence must be re-issued after shutting off AUXP transmission in order to configure the ZipWirePlus device to the previous PRBS configuration.

NOTE: Disable E1 Framer before switching payload to unframed AUXP, otherwise if any overhead or CRC4 is set to manual mode and E1 Framer is enabled, it will corrupt unframed AUXP data stream.

9.4.5 Framed AIS

Monitored overhead bit processing is active when generating framed AIS. When framed AIS is transmitted, the CRC-4 generation mode should be set to automatic to avoid downstream CRC-4 errors.

9.4.6 Frame Alignment

Software will report extensive CRC-4 errors when there is no multi-frame alignment.

9.4.7 Sa6 Processing

The host processor is responsible for monitoring and updating the receive and transmit Sa6 byte values according to the ETS 300 233 standard.

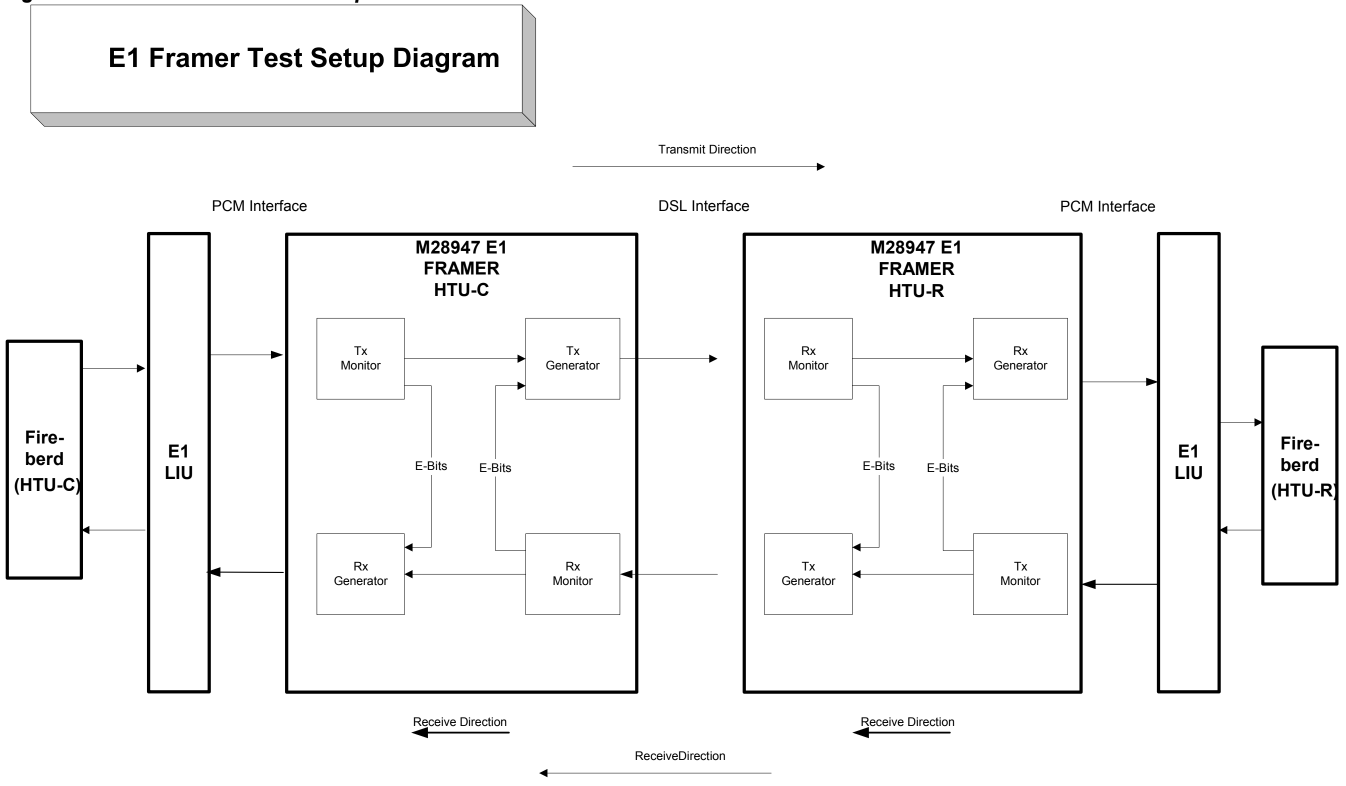
The Sa6 nibbles must be set to the same value, i.e. 0x00, 0x11, 0x44, etc. when using the generator API. A value of 0x45 would be illegal.

The ZipWirePlus firmware will report changes in the Sa6 byte as it does for all the overhead bytes. The ZipWirePlus firmware examines the entire byte, not the nibbles individually. For instance, a change of 0x11 to 0x22 will be reported via an unsolicited interrupt to the host processor. A change from 0x45 to 0x45 would not be reported to the host processor.

9.5 Test Setup

The following diagram illustrates the E1 Framer test setup.

Figure 9-3 E1 Framer Test Setup



10.0 HDSL1 Feature

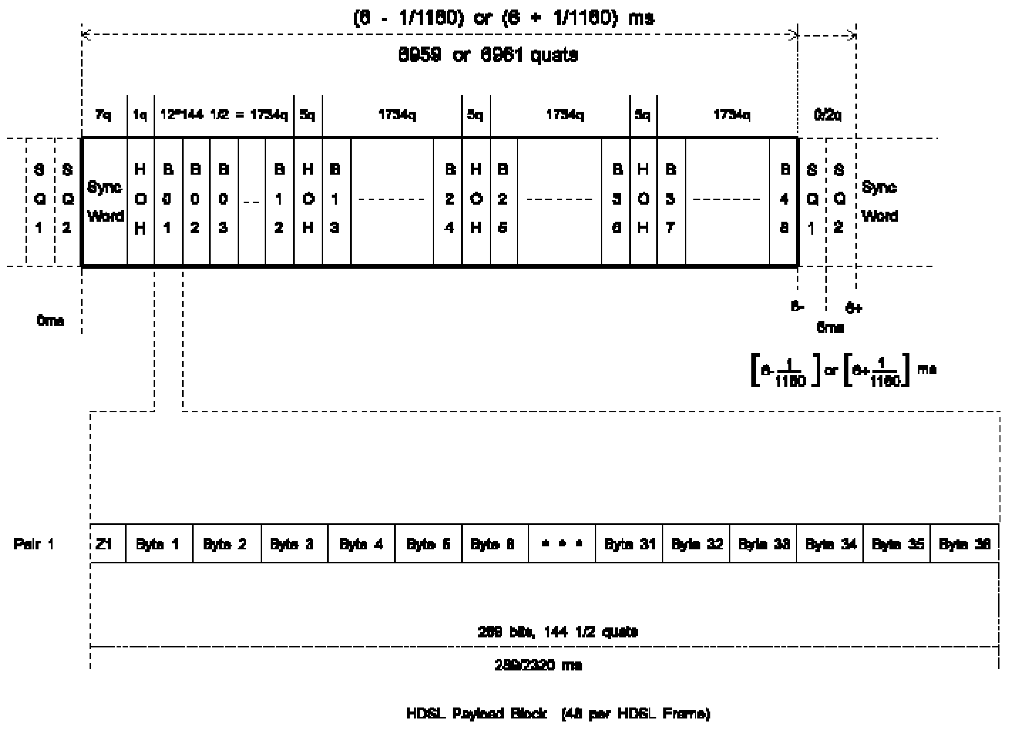
The HDSL1 feature is available on the M28950 device only. This device supports HDSL1 mode in addition to G.SHDSL mode. This section provides an overview of the APIs, provided by the ZipWirePlus firmware, for HDSL1 Feature.

10.1 Overview

10.1.1 HDSL1 Frame Structure

Each HDSL1 frame is nominally 6 ms in length and consists of 48 payload blocks. Each block contains one overhead Z-bit and application specific number of payload bytes. Groups of 12 payload blocks are separated by an ordered set of overhead bits including indicator bits and EOC bits. The start of the frame is identified by a 14-bit synchronization word. For more information, refer to ETS 101 135 (1998-02) Technical Specification document. The ZipWirePlus API commands will enable the host to write the overhead bits in the transmitted frames and read those bits in the receiver side.

Figure 10-1 HDSL 1 Frame Structure



Symbol	Name, function
B01 to B48	HDSL system payload blocks
Byte n	Byte n from core frame (n = 1 to 144)
HOH	HDSL overhead (sw, eoc, crc,...)
quat	Quaternary symbol
SQ1, SQ2	Stuff quats
Sync word	7-symbol Barker codes, "double Barker" $\rightarrow$ 14 bits
"6-", "6+"	6 - 1/1160 ms, 6+1/1160 ms
Z_n	Additional overhead bits (Z-bits)
n	Indicating number of payload block (n = 1 to 48)

Figure 9: Frame structure of the one pair system

10.1.2 Features

- ♦ Supports training with the RS8973 device.
- ♦ Supports communication with the RS8953 device using the 0x72 syncword.
- ♦ Provides access to z-bits, EOC bits and indicator bits via API commands.
- ♦ Provides error counters – CRC, FEBE.

10.1.3 Limitations

AutoBaud post activation is not supported. The HDSL1 and Autobaud should NOT be used together.

In HDSL1, during startup, the signal S0 and S1 signals are framed in HDSL1 frame. The M28950 does not transmit framed S0 and S1; it transmits unframed S0 and S1. The M28950 receiver can detect both framed and unframed S0 and S1 to establish the data link. This limitation does not affect the inter-operability of M28950 to RS8973/8953 that sends framed S0/S1.

This is a single pair implementation. Multi-pair operation is not supported.

PID support is for only one loop. The ZipWirePlus implementation does not support multiple loops.The PID sets the first 8 Zbits to the value of 0x01. It is left to user to set the value back to 0xFF after the far-end PID has completed. Or alternatively the PID can be disabled and PID will be set to a default value of 0xFF.

EOC is a clear channel implementation. Interpretation or message formatting needs to be done in the Host.

10.2 HDSL 1 Overhead bits Processing.

Each HDSL1 frame contains 96 overhead bits including indicator, EOC and Z bits.

The host can deliver the overhead bits to the ZipWirePlus by using API commands. The overhead bits will be transmitted in the next HDSL1 frame and subsequent frames until the next overhead bits are received from the host.

On the receiver side, the overhead bits are extracted from the incoming HDSL1 frames and can be delivered to the host upon request. Setting the Tx Overhead Bits

10.2.1.1 Indicator bits

The host can set indicator bits by sending _DSL_WRITE_IND_BITS (0x60) API command. The bits received from the host will be set on the next frame boundary (typically 6ms) and will continue to be sent on all subsequent frames until the command is issued again.

10.2.1.2 Z-bits

The host can set the six bytes of transmitted Z-bits by issuing _DSL_WRITE_ZBITS (0x63) API command. The Z-bits will be set on the next frame boundary (typically 6ms) and will continue to be sent on all subsequent frames until the command is issued again.

The Z-bits will be internally double-buffered. The Host will initially write two blocks of Z-bits, then a single block at each 6 ms interrupt. This approach ensures a continuous Z-bit stream to be transmitted.

10.2.1.3 EOC bits

The host can set a block of 13 EOC bits through _DSL_WRITE_EOC (0x64) API command. The bits received from the host will be set on the next frame boundary (typically 6ms) and will continue to be sent on all subsequent frames until the command is issued again. The Host will initially write two blocks of EOC bits, then a single block at each 6 ms interrupt. This approach ensures a continuous EOC bit stream to be transmitted.

10.2.2 Accessing the Rx Overhead bits

10.2.2.1 Indicator bits

In the receiver side, the host can request Rx indicator bits by sending _DSL_READ_IND_BITS (0xD0) API command. In response, the ZipWirePlus delivers the indicator bits extracted from the most recent DSL frame.

10.2.2.2 Z-bits

There are two mechanisms for accessing the Z-bits received from the incoming DSL frames.

Polling :- The host sends _DSL_READ_IND_BITS (0xD0) API command and in return the ZipWirePlus delivers the Z-bits received from the most recent DSL frame. This is the default mechanism.

Interrrupts :- The host will request that the Z-bits being sent to the host via unsolicited interrupts. Upon receiving a new HDSL1 frame and extracting the 6 bytes of Z-bits, the ZipWirePlus sends the Z-bits to the host through an unsolicited API command (_DSL_READ_Z_BITS) nominally every 6 ms. To set up this mechanism, the host shall follow the steps defined below

The host should invoke _DSL_INTR_HOST_MASK (0x50) API command to enable unsolicited interrupts due to API commands.

The host sends _DSL_INTR_API_SUBMASK (0x50) API command (0x51) to the ZipWirePlus device and writes 1 to Byte 1, `0xD1` (the opcode of _DSL_READ_ZBITS command) to Byte 2 and 1 to Byte 3 of the incoming parameters of _DSL_INTR_API_SUBMASK (0x51) API command to enable the unsolicited interrupts for _DSL_READ_ZBITS (0xD1) API command.

10.2.2.3 EOC bits

There are two mechanisms for accessing the EOC bits received from the incoming DSL frames.

Polling :- The host sends _EOC_RX_GET_MSG (0xB1) API command and in return the ZipWirePlus delivers the EOC bits to the host processor. This is the default mechanism.

Interrrupts :- The host processor will request that the EOC bits being sent to the host upon receiving a new HDSL1 frame via the unsolicited interrupt mechnism mentioned above.The ZipWirePlus sends the EOC bits to the host through an unsolicited API command nominally every 6 ms. To set up this mechanism, the host shall follow the steps defined below

The host should invoke _DSL_INTR_HOST_MASK (0x50) API command to enable unsolicited interrupts due to API commands.

The host sends _DSL_INTR_API_SUBMASK (0x51) API command (0x51) to the ZipWirePlus device and writes 1 to Byte 1, 0xB1 (the opcode of _EOC_RX_GET_MSG (0xB1) API command and in return the ZipWirePlus delivers the EOC bits to command) to Byte 2 and 1 to Byte 3 of the incoming parameters of _DSL_INTR_API_SUBMASK (0x51) API command to enable the unsolicited interrupts for _EOC_RX_GET_MSG (0xB1) API command. If the host processor can request that both EOC and Z-bits to be delivered concurrently by unsolicited interrupts mechanisms. This reduces the number of unsolicited API commands sent to the host every 6 ms. To receive both EOC and Z bits through _EOC_RX_GET_MSG (0xB1) command, the host should write 3 to Byte 3 of the incoming parameters of _DSL_INTR_API_SUBMASK (0x51) API command. The indicator bits can also be delivered along with EOC and Z bits. To request it, the host should write 7 to Byte 3 of incoming parameters of _DSL_INTR_API_SUBMASK (0x51) API command.

10.3 HDSL1 APIs

Table 10-1 lists the API commands supporting HDSL1 Feature.

Table 10-1 HDSL1 API Commands

Command	Control Status	Opcode	Description
_DSL_WRITE_IND_BITS	Control	0x60	Indicator bits to be transmitted out the next DSL frame.
_DSL_READ_IND_BITS	Status	0xD0	Indicator bits received from the incoming DSL frame
_DSL_WRITE_ZBITS	Control	0x63	Z-bits to be transmitted out the next DSL frame.
_DSL_READ_ZBITS	Status	0xD1	Z-bits received from the incoming DSL frame
_ DSL_WRITE_EOC	Control	0x64	EOC bits to be transmitted out the next DSL frame.
_EOC_RX_GET_MSG	Status	0xB1	EOC bits received from the incoming DSL frame
_DSL_AUTO_IND	Control	0x62	Enable Auto-Update of Indicator Bits
_DSL_INTR_HOST_MASK	Control	0x50	Enable/Disable Unsolicited Interrupts0x51
_DSL_INTR_API_SUBMASK	Control	0x51	Enable/Disable unsolicited Interrupts for API Commands
_DSL_CONFIG_PID	Control	0x2B	Configures the Pair Identification Functionality
_DSL_FR_2B1Q_CONFIG	Control	0x65	Configures the 2B1Q Mode

11.0 IDSL NT Feature

11.1 Introduction

The ZipWirePlus device supports the IDSL NT mode in addition to G.shdsl, HDSL1 and HDSL2 standards. This document provides an overview of the IDSL feature. This section guides the user on how to configure and manage the ZipWirePlus device for an IDSL Application.

The API commands described in this document should be used by the host to setup the IDSL activation. The host will be able to set the PCM and DSL rate. In addition, the host can write the indicator bits in the transmitted frames and read those bits in the receiver side. The host is responsible for sending and receiving the EOC messages. On the NT side, the Rx EOC messages extracted from the DSL frames. Valid messages are delivered to the host where the interpretation is done by the host processors.

11.2 Overview

11.2.1 IDSL Features

- Fully T1.601-1999 and ETSI TS 101 080 V1.3.1. standard compliant
- Supports NT operation Only.
- Supports data rate of 64, 128 & 144 Kbps with line rate of 80Kbaud (160Kbps)
- Supports full termination of IDSL superframe M bits and synchronization
- Fully interoperable with existing IDSL 2B1Q Transceivers
- Fully compliant with the IDSL jitter tolerance as specified in ETSI TS 101 080 V1.3.1
- Fully compliant with the IDSL PSD Masks (for both LT and NT) as specified in T1.601-1999.

11.2.2 References

- ANSI T1.601-1999, Integrated Services Digital Network (ISDN) – Basic Access Interface for Use on Metallic Loops for Application on the Network Side of the NT (Layer 1 Specification)
- ETSI TS 101 080 V1.3.1, Transmission and Multiplexing(TM), Integrated Services Digital Network (ISDN) basic rate access, Digital Transmission system on metallic local lines.

11.3 IDSL Overhead Bits

Each 12 ms IDSL superframe contains 36 overhead bits including indicator and EOC bits and excluding CRC bits. The host can deliver the overhead bits to the ZipWirePlus device by using the ZipWirePlus API commands. The overhead bits will be transmitted in the next IDSL frame and subsequent frames until the next overhead bits are received from the host. On the receiver side, the overhead bits are extracted from the incoming IDSL frames and monitored. If a change of any of these bits is detected, the host will be notified by an unsolicited interrupt enabled by the host.

11.3.1 Setting the Transmit Overhead Bits

11.3.1.1 Indicator Bits

The host can set the indicator bits (**act**, **ps1**, **ps2**, **ntm**, **cso**, **sai, uoa**, and **aib**) by sending _DSL_WRITE_IND_BITS (0x60) API command. The ZipWirePlus device is responsible of generating **crc, febe** and **dea** bits accordingly. The bits received from the host will be set on the next frame boundary (typically 12 ms) and will continue to be sent on all subsequent frames until the command is issued again.

11.3.1.2 EOC bits

Activation of an EOC function is a one-way operation. The LT sends the message and the NT initiates the action if it supports the requested function.

11.3.2 Accessing the Receive Overhead Bits

11.3.2.1 Indicator Bits

In the receiver side, the host can request Rx indicator bits by sending _DSL_READ_IND_BITS (0xD0) API command. In return the ZipWirePlus device delivers the indicator bits extracted from the most recent IDSL frame.

The ZipWirePlus device is capable of monitoring the Rx indicator bits, detecting a change in any IDSL indicator bits and reporting it to the host. To enable this feature the following steps should be taken.

- The host should send DSL_INTR_HOST_MASK API command (0x50) to enable unsolicited interrupts due to API commands.
- To turn on a specific unsolicited API command, the host sends _DSL_INTR_API_SUBMASK API command (0x51) to the ZipWirePlus device. To receive the indicator bits (whenever a change is detected), the host writes 0x01 to Byte 1, 0xD0 (the opcode of _DSL_READ_IND_BITS command) to Byte 2 and 0x01 to Byte 3 to enable the interrupt for _DSL_READ_IND_BITS API command [Refer to Section **Error! Reference source not found.**].
- To allow activation of the interrupt signal on the change of a subset of the indicator bits, the host can use Bits 1-7 of Byte 3 to define a mask for the indicator bits

11.3.2.2 EOC bits

The EOC message must be received at least three times by the NT before it can be processed. Upon receiving the EOC messages, the NT echoes back the EOC messages toward the LT. Meanwhile it checks for validity of the message. After receiving the third message NT either echoes it back as an affirmative reply or sends Unable To Comply message to the LT (if the Rx EOC message is not valid or doesn't contain one of the supported functions). If the EOC message is valid, the host will be notified via the unsolicited interrupt mechanism and the processing is done by the host preocessor. To activate the interrupt signal, the followings steps need to be taken:

- The host should send DSL_INTR_HOST_MASK (0x50) API command to enable unsolicited interrupts due to API commands.
- To turn on a specific unsolicited API command, the host sends _DSL_INTR_API_SUBMASK (0x51) API command to the ZipWirePlus device and writes 0x01 to Byte 1, 0xB1 (the opcode of _EOC_RX_GET_MSG command) to Byte 2 and 0x01 to Byte 3 to enable the interrupt for _EOC_RX_GET_MSG API command [Refer to Section **Error! Reference source not found.**].

The EOC frames received by NT are handled by the ZipWirePlus device. It checks the address field of the frame, if the address is other than its own (000 or 111), the NT sends Hold State message to the LT. Otherwise, it initiates the action after receiving three identical and consecutive EOC frames. In addition, the NT will echo the received EOC frames toward the LT. If the frame contains one of the supported EOC functions, it notifies the host with the EOC message. The host will decide how to respond to the requested function. The supported EOC functions are specified by the host through _DSL_WRITE_EOC (0x64) API command.

11.4 IDSL APIs

Table 11-1 shows the complete list of IDSL API commands.

Command	Type	Opcode	Description
_DSL_SYSTEM_CONFIG	Control	0x06	Operation Mode
_DSL_MULTI_RATE_CONFIG	Control	0x1B	PCM data rate
_DSL_WRITE_IND_BITS	Control	0x60	DSL Framer Write Indicator Bits
_DSL_READ_IND_BITS	Status	0xD0	DSL Framer Read Indicator Bits
_ DSL_WRITE_EOC_BITS	Control	0x64	DSL Framer Write EOC Bits
_EOC_RX_GET_MSG	Status	0xB1	DSL Framer Read EOC Bits
_DSL_INTR_HOST_MASK	Control	0x50	Unsolicited Interrupts Mask
_DSL_INTR_API_SUBMASK	Control	0x51	Unsolicited Interrupts API Command Sub Mask

Table 11-1 IDSL API Commands

12.0 Diagnostics and Test Modes

The ZipWirePlus operational code supports several diagnostics and test modes to help in development, debugging, troubleshooting, and manufacturing of the ZipWirePlus modem. Section 12.1 provides information about ZipWirePlus operational error counters. Section 12.2 provides information on loopback options. Section 12.3 discusses the Echo Return Loss Enhancement (ERLE) test option used to validate the hybrid and AFE circuitry. Section 12.4 provides information on BER meters.

12.1 DSL Operational Error Monitoring and Handling

The ZipWirePlus firmware is responsible for processing the SYNC status, error status, DPLL, and stuffing generator requests every 6ms.

12.1.1 Sync Status Monitoring

The loop synchronization status is checked and updated every 6 ms. Use the _DSL_STATUS (0x85) API command to determine the status of the link.

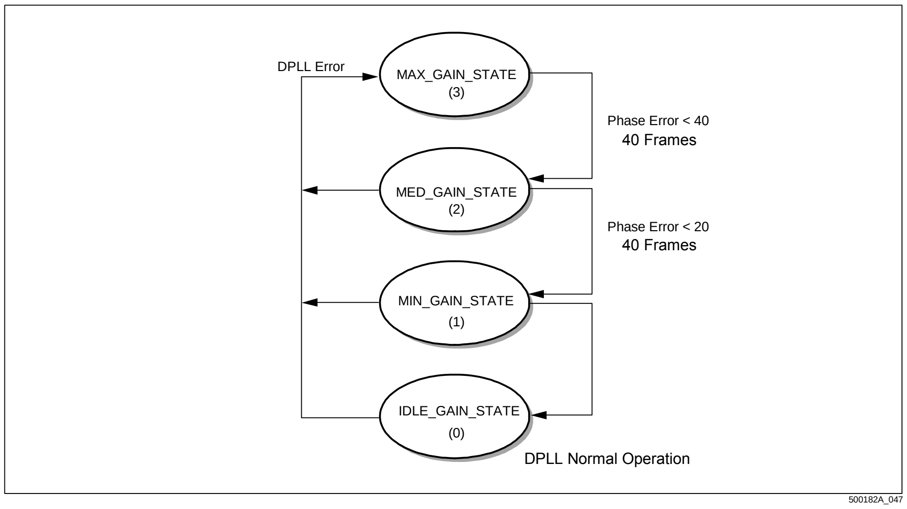
12.1.2 Operational Error Status Reporting

The ZipWirePlus firmware periodically updates the operational error counters. When an error occurs, the corresponding error counter increments. Use the _DSL_OPER_ERR_CTRS (0x9C) and _ATM_PHY_OPER_ERR_CTRS (0xB8) to determine the DSL and ATM operational error counters. The _DSL_CLEAR_ERROR_CTRS (0x40) resets all ZipWirePlus error counters.

12.1.3 DPLL Manager Error Monitoring and Handling

Figure 12-1 illustrates the DPLL state machine. The basic purpose of the DPLL state machine is to increase (open up) the DPLL bandwidth when an error occurs; this allows the DPLL to increase its capture range. As the DPLL locks onto the far-end PCM clock, the DPLL bandwidth is decreased to minimize the PCM RCLK jitter. The DPLL takes less than 1 second to stabilize after a DPLL error is detected. Once the DPLL stabilizes, the appropriate receiver FIFO resets are applied. The _DSL_DPLL_CLOCK_GEN (0x88) API command configures the DPLL clock. The _DSL_STAGE_NUMBER (0x8F) API command determines the state of the internal DPLL state machine.

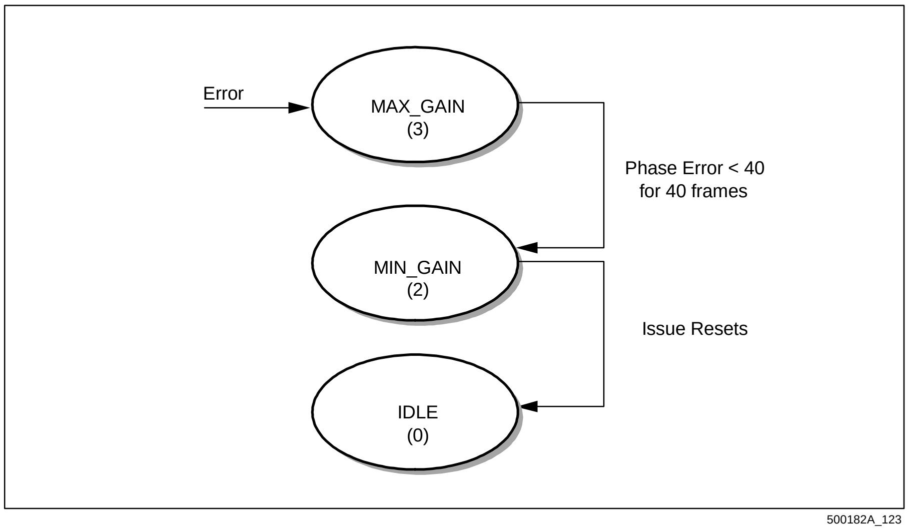
Figure 12-1 DPLL State Machine



12.1.4 Stuffing Manager Error Monitoring And Handling

Figure 12-2 illustrates the stuffing manager. The stuffing manager monitors the phase error between the PCM and DSL clocks. Once the stuffing phase stabilizes, the appropriate FIFO resets are applied. The _DSL_STAGE_NUMBER (0x8F) API command determines the state of the internal stuffing manager state machine.

Figure 12-2 Stuffing Manager State Diagram



12.1.5 Operational Error Monitoring API Commands

Table 12-1 lists the API commands used for operational error monitoring.]

Table 12-1 Operational Error Monitoring API Command Summary

COMMAND	OPCODE	DESCRIPTION
_DSL_STAGE_NUMBER	0x8F	Queries the stage numbers of various state machines
_DSL_CLEAR_ERROR_CTRS	0x40	Clears all ZipWirePlus error counters
_DSL_OPER_ERR_CTRS	0x9C	Queries the ZipWirePlus operational error counters
_ATM_PHY_OPER_ERR_CTRS	0xB8	Queries the ATM PHY operational error counters
_DSL_INJECT_CRC_ERROR	0x41	Injects CRC errors in DSL frames
_ATM_PHY_INJECT_HEC_ERROR	0x1F	Inject HEC errors in ATM cells
_DSL_DBANK	0x27	Sets DBANK patterns

12.2 Loopbacks

The ZipWirePlus family of devices provide loopback capabilities at multiple interfaces and can be configured for several loopback options.

12.2.1 Loopback Interfaces

In general loopbacks are supported at the following interfaces. The name of the interface as used throughout this section is shown in parentheses.

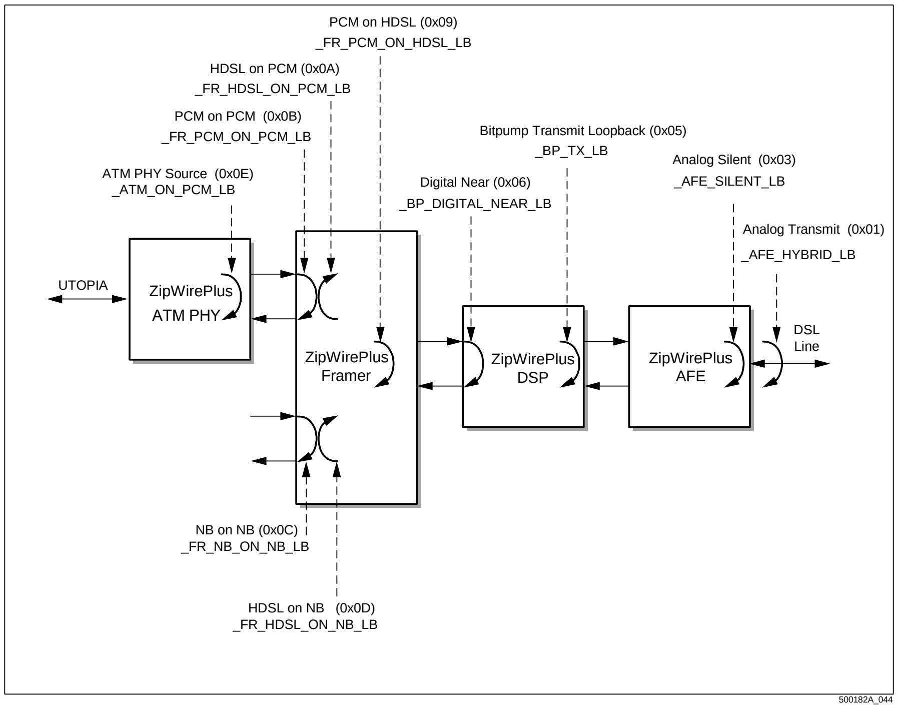
- PCM Interface–(PCM)
- Narrowband Interface–(NB)
- Framer to DSP Serial Interface (Internal)–(HDSL)
- Bit Pump to AFE Serial Interface–(BP TX)
- AFE Analog Transmit to AFE Analog Receive Interface (Internal)–(AFE_SILENT)
- Hybrid or DSL Line Interface (AFE_HYBRID)

NOTE: These interfaces are not available on all the devices. The M28950 does not support Narrowband Interface.

12.2.2 Loopback Configurations

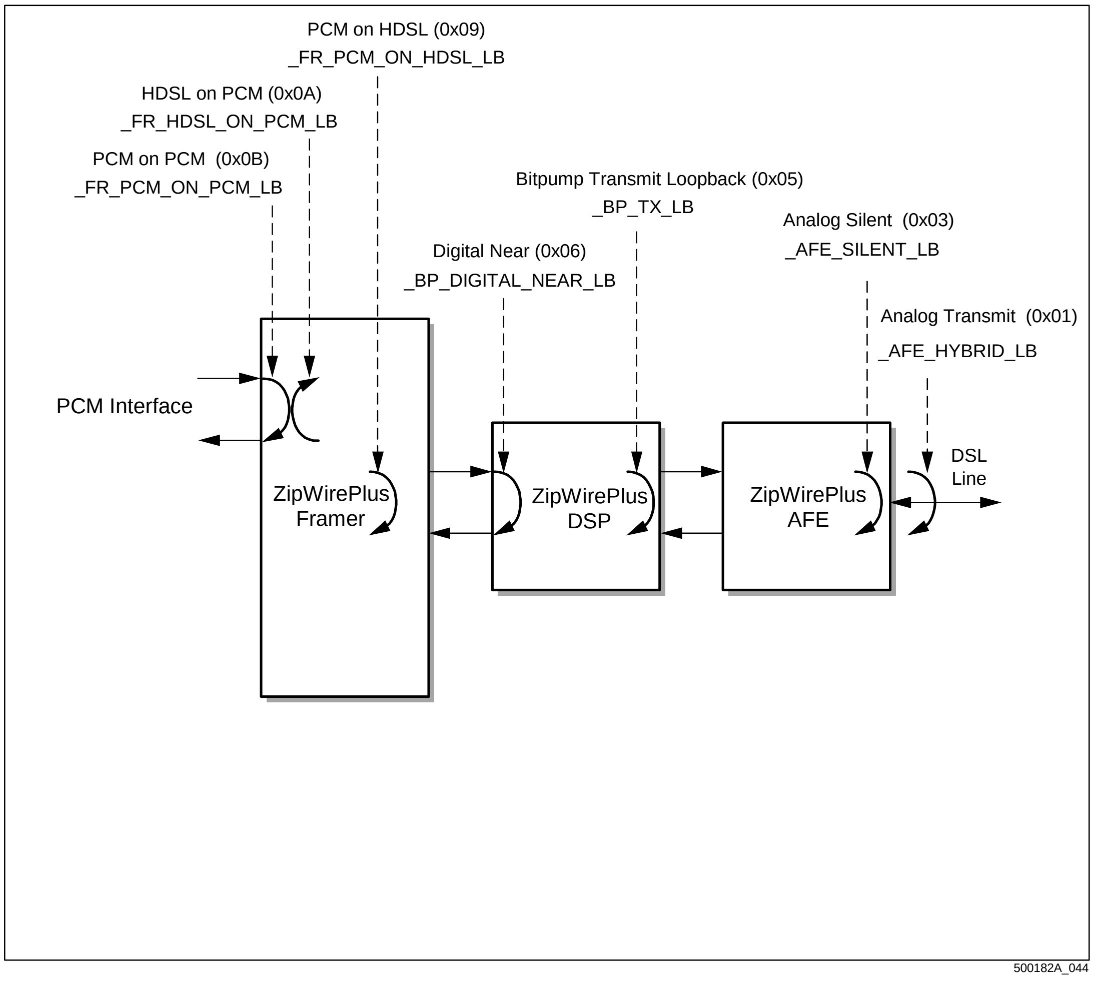
Figure 12-3 illustrates all loopback options for the ZipWirePlus chipset.

Figure 12-3 Loopbacks on M28945, M28946 & M28947 devices



In Figure 12-3 and Figure 12-4 a naming convention is used for identifying the loopback option. An example of a loopback option is PCM_ON_HDSL loopback. The loopback mode is also accompanied by the corresponding #define constant (_FR_PCM_ON_HDSL_LB) to be used with _DSL_LOOPBACK (0x09) API command. The name of a loopback option has two parts to it. For example PCM_ON_HDSL has two parts to its name: the first part is PCM and the second part is HDSL. The first part determines the type of data loopback is performed on. The second part determines the type of interface where the loopback is performed. In the example of PCM_ON_HDSL loopback mode, the loopback is performed on PCM data (first part) and the loopback is performed at the HDSL (second part) interface.

Figure 12-4 Loopbacks on M28950 device



12.2.3 Loopback API Options

Table 12-2 provides a detailed description of the different loopback options. The ZipWirePlus device can be configured in different loopback modes using the _DSL_LOOPBACK (0x09) API command. Loopback options are classified as being either destructive or non-destructive. Setting the ZipWirePlus for a destructive loopback option causes the DSL link to drop. Non-destructive loopbacks are maintained during retrains.

Table 12-2 Loopback Modes

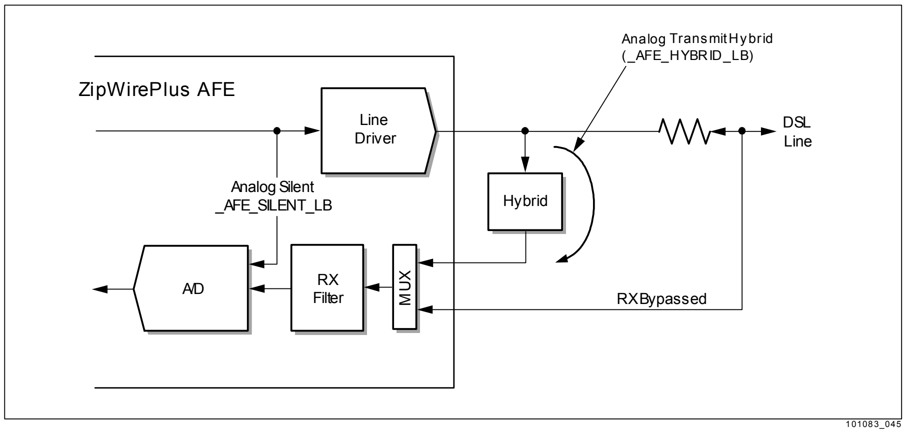
LOOPBACK OPTION	C CONSTANT	DESCRIPTION
_ATM_ON_PCM_LB	0x0E	The ATM PHY transmit data is looped back (ATM Interface), before the DSL framer interface, back to the ATM PHY receive section. (Not Valid on M28950)
_FR_PCM_ON_PCM_LB	0x0B	The clock, data, and sync signals are looped back internally (PCM interface) from the DSL framer PCM transmits inputs to the DSL framer PCM receive inputs.

LOOPBACK OPTION	C CONSTANT	DESCRIPTION
_FR_HDSL_ON_PCM_LB	0x0A	The clock, data, and sync signals are looped back internally (PCM Interface) before the DSL framer PCM receive inputs back into the DSL framer PCM transmit inputs.
_FR_PCM_ON_HDSL_LB	0x09	The data is looped back internally, before the DSL framer transmit section (HDSL Interface), back to the DSL framer receive section. The DSL framer scrambler and de-scrambler are set to use the same tap.
_FR_NB_ON_NB_LB	0x0C	The clock, data, and sync signals are looped back internally from the DSL framer and NB transmit inputs to the DSL framer NB receive inputs. (Not Valid on M28950)
_FR_HDSL_ON_NB_LB	0x0D	The clock, data, and sync signals are looped back internally before the DSL framer and NB receive inputs back to the DSL framer NB transmit inputs. (Not Valid on M28950)
_BP_DIGITAL_NEAR_LB	0x06	The data is looped back internally before the bit pump DSP transmit section is looped back to the DSL framer DSL receive section. When the DSL framer is present, the DSL framer scrambler and descrambler are set to use the same tap.
_BP_TX_LB	0x05	The data is looped back internally, before the bit pump to the AFE interface, back into the bit pump AFE serial inputs. The AFE line driver continues to output proper levels. The AFE receiver and hybrid inputs are bypassed.
_AFE_SILENT_LB	0x03	The data is looped back internally before the AFE line driver into the AFE A/D converter. The AFE receiver and hybrid inputs are bypassed. The AFE line driver is disabled.
_AFE_HYBRID_LB	0x01	The data is transmitted out the AFE line driver and looped back into the AFE hybrid input. The AFE receiver inputs are bypassed.

The _BP_DIGITAL_NEAR_LB (0x06) and _FR_PCM_ON_HDSL_LB (0x09) is automatically disabled (exited) when the modem trains. Issuing the _DSL_ACTIVATION (0x0B) clears all loopback and test modes.

Figure 12-5 illustrates the ZipWirePlus AFE and hybrid configurations for the _AFE_SILENT_LB and _AFE_HYBRID_LB loopback modes.

Figure 12-5 Detailed AFE Loopbacks



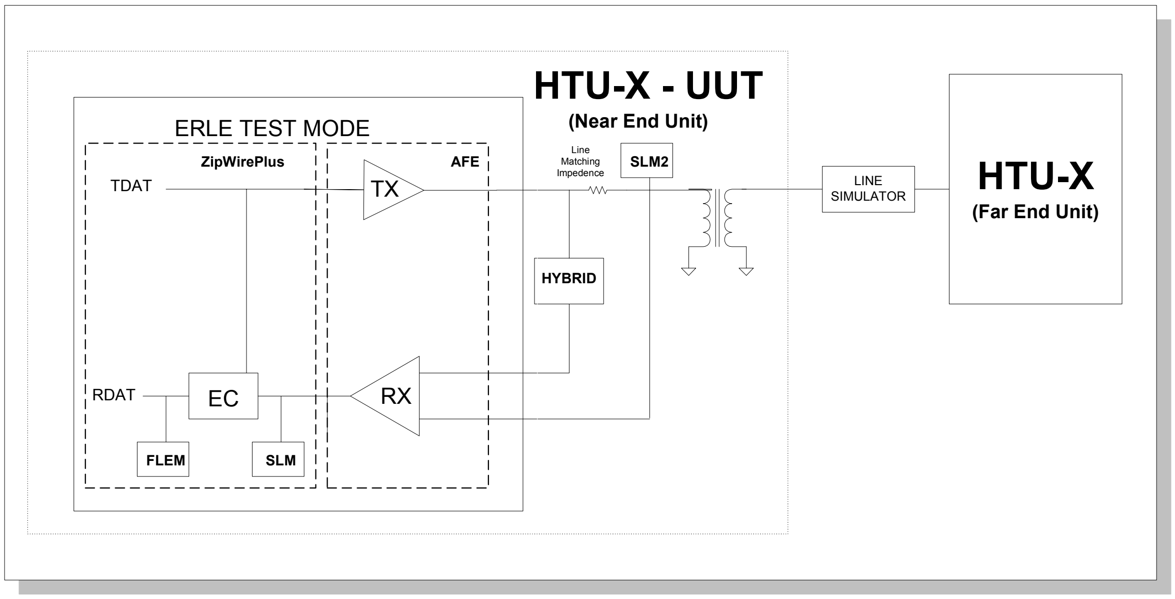
12.3 ERLE

Echo Return Loss Enhancement (ERLE) is a diagnostic tool to verify the integrity of the hardware's analog front end, transformer, surge protection, etc. The ERLE result determines the amount of echo canceled by the system. The echo is canceled initially by the external hybrid (analog ERLE) and then by the linear digital echo canceller in the bit pump (digital ERLE). This diagnostic tool can be used during development and manufacturing to identify problems with the AFE and Hybrid circuits.

12.3.1 Test Setup

Figure 12-6 illustrates the test setup needed for ERLE diagnostic mode. The Unit Under Test (UUT) is connected to a reference HTU-X (far-end unit) via a line simulator. The line length is set to ~ 10,000 ft (24 or 26 AWG). The far end unit's transmitter may be disabled or powered down for ERLE testing. All configurations for ERLE are done via software API commands.

Figure 12-6 ERLE Test Setup



12.3.2 ERLE TEST MODES

There are two major steps to the ERLE test modes:

1. Background Noise Test—Transmitter OFF
2. ERLE Test—Transmitter ON

12.3.2.1 Background Noise Test—Transmitter OFF

The background test measures the Signal Level Meter (SLM) while the transmitter is turned off. This effectively gives the amount of noise present at the input of the A/D converter. The noise result should be very small. A large noise result with the transmitter off usually indicates a DSL connection to a remote unit that is transmitting.

12.3.2.2 ERLE Test—Transmitter ON

The ERLE result determines the amount of echo canceled. The echo is canceled initially by the external hybrid followed by the Linear Echo Canceller (LEC) in the bit pump. The Analog ERLE (AERLE) determines how much echo is canceled by the external hybrid. The SLM measures the input to the A/D converter. The FELM measures the output of the Digital Echo Canceller (DEC). The FELM value should be small because there is no far-end transmitter. The SLM/FELM ratio gives a quality measurement of the digital echo cancellation; the larger the ratio, the better the performance of the digital echo cancellation. A large SLM or FELM with the transmitter on could indicate that the DSL is not properly terminated. The SLM2 measures the TX signal power when bypassing the analog hybrid. The SLM2/SLM ratio gives a quality measurement of the analog echo cancellation. The following equations show how DERLE and AERLE are computed from SLM, SLM2, and FELM.

DERLE = 20 * log (SLM / FELM)
AERLE = 20 * log (SLM2 / SLM)

12.3.3 ERLE API Command Summary

Table 12-3 lists API command used for configuring the ZipWirePlus device for ERLE test modes.

Table 12-3 ERLE API Command Summary

COMMAND	OPCODE	DESCRIPTION
_BP_ERLE_TEST_MODE	0x18	Activates the ERLE test mode.
_BP_ERLE_RESULTS	0x93	Gets ERLE test mode results.
_DSL_TEST_MODE	0x0D	Configures the device in special test modes. Use this command with the _EXIT_TEST_MODE option to abort an ERLE test before completion.

12.3.4 ERLE Procedure

1. Connect your board under test via a line simulator of ~10,000 ft (24 or 26 gauge) and terminate the far-end with its transmitter turned off or the unit powered off.
2. Download and configure the modem as you would to run end-to-end except do not issue the _DSL_ACTIVATE (0x0B) command.
3. Issue the _BP_ERLE_TEST_MODE (0x18) with a parameter of 0x00.
4. Wait for ERLE to complete by polling the _DSL_STATUS (Status 1 byte, upper 2 bits) until it reaches Normal Operation. ERLE takes ~ 1 second.
5. Get the ERLE results by issuing the _BP_ERLE_RESULTS (0x93) and apply the DERLE and AERLE formulas as specified in Section 12.3.2.2.
6. Analyze results.

12.4 BER Meter

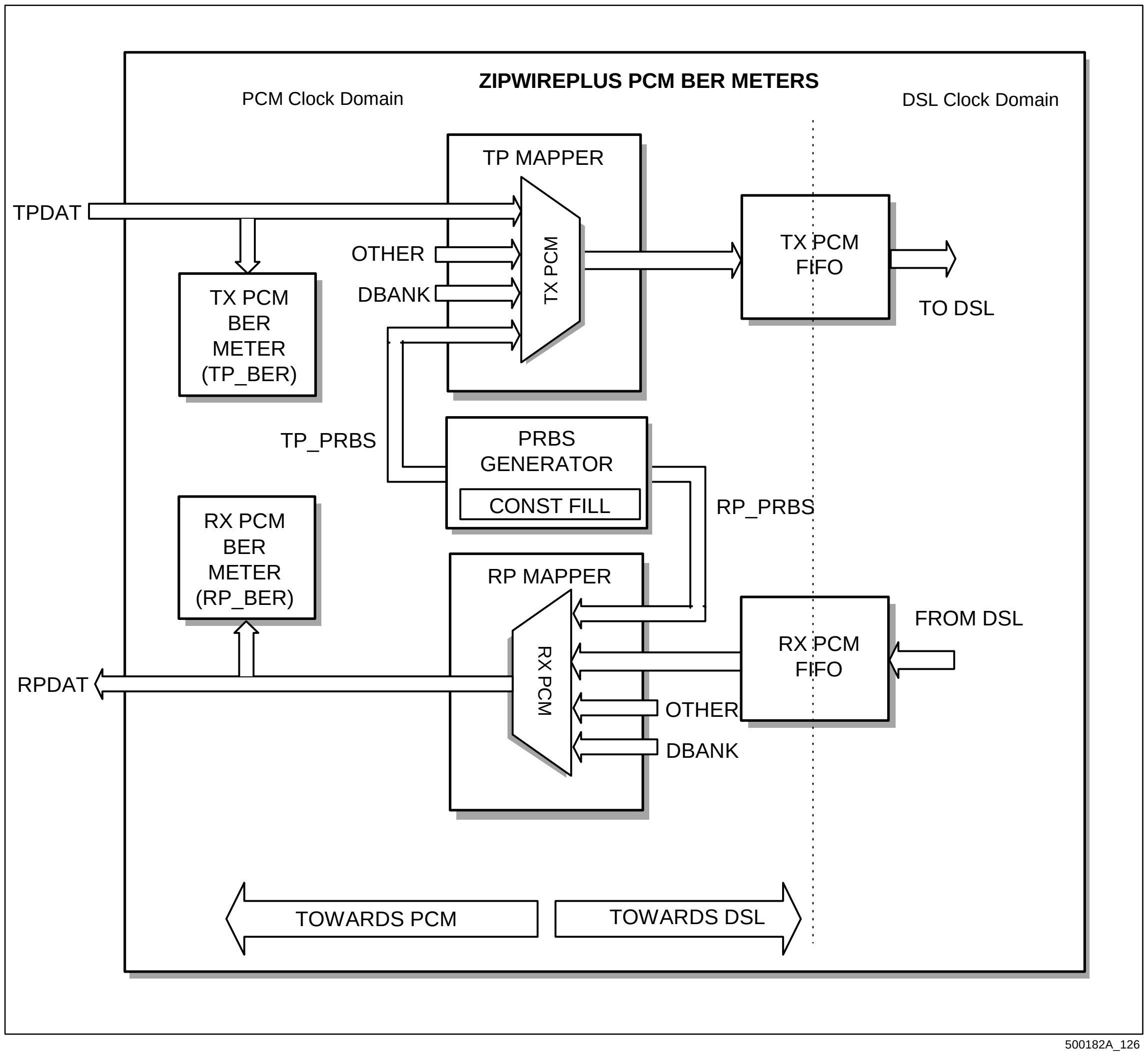
The DSL framer has four PRBS/BER meter modules, supporting BER measurement towards the DSL, PCM, and NB sides. The internal BER module can be used to test the DSL link, PCM, or NB interface data path without an external BER tester. A BER module is a combination of a PRBS generator and a BER meter. The PRBS generator is shared by both the TX and RX BER modules.

Figure 12-7 illustrates the PCM TX and RX BER modules. The Narrowband TX and RX BER modules (on M28945, M28946 & M28947) are identical to the PCM BER modules.

TP_PRBS and RP_BER function as PCM (NB) TX BER module towards the HDSL side. The RP_PRBS and TP_BER function as PCM (NB) RX BER module towards the PCM (NB) side.

The PRBS sequence can override TPDAT and RPDAT on a per time slot basis, and generate any framed or unframed test pattern. The OH (overhead) bits are left unchanged. Time slots can then be examined for test patterns on either a per time slot basis or on the entire frame. The PRBS sequence is programmable using the _DSL_PRBS_CONFIGURE (0x25) API command. This API command can also be used to invert the TP_BER and RP_BER sequence. Optionally, a constant value on a per time slot basis can override TPDAT and RPDAT instead of a PRBS sequence.

Figure 12-7 ZipWirePlus Framer PCM BER Meter



For proper operation of the BER modules, both PRBS generator and BER meter resets must be issued in each BER measurement process. To avoid initialization bit errors, the PRBS generator reset must be issued before the BER meter reset is issued. The PRBS generator is reset by issuing the _DSL_FR_TP_RESET (0x4E) API command. The TX PCM BER meter (TP_BER) is reset using the _DSL_TP_BER_STATE (0x23) API command. The RX PCM BER (RP_BER) meter is reset using the _DSL_RP_BER_STATE (0x24) API command.

After the BER module is reset, it starts the qualification phase. During the qualification phase the BER module is initialized and synchronized for data alignment. The BER error counters are not updated during the qualification phase. After successful completion of the qualification phase, the BER meter automatically starts the measurement phase. The results of the BER meter are valid only after the successful completion of the measurement phase. The status of the RX PCM BER (RP_BER) meter is obtained using the _DSL_RP_BER_RESULTS (0x8D) API command. The BER test interval (BER_SCALE) selected by the _DSL_PRBS_CONFIGURE (0x25) API command determines the average BER.

Section 12.4.1 lists the API commands used to configure the ZipWirePlus device for BER testing.

12.4.1 BER Meter API Commands

API commands common to both PCM and NB BER testing is covered in Section 12.4.1.1. PCM BER Meter API commands are covered in Section 12.4.1.2. Narrowband BER Meter API commands are covered in Section 12.4.1.3.

12.4.1.1 PCM and NB Common BER Meter API Commands

Table 12-4 lists a summary of BER Meter API commands common to PCM and NB.

Table 12-4 Common BER Meter API Commands

COMMAND	OPCODE	DESCRIPTION
_DSL_PRBS_CONFIGURE	0x25	Configures the PRBS generator for a selected pattern.
_DSL_CONST_FILL	0x26	Sets the fill pattern to be used when BER data is configured to be sourced from a fixed pattern.

12.4.1.2 PCM BER Meter API Commands

Table 12-5 lists a summary of commands used in PCM BER testing.

Table 12-5 PCM BER Meter API Commands

COMMAND	OPCODE	DESCRIPTION
_DSL_FR_TX_RESET	0x4E	Resets the TX PCM PRBS generator and TX PCM BER meter (TP_BER).
_DSL_FR_RX_RESET	0x4F	Resets the RX PCM PRBS generator and RX PCM BER meter (RP_BER).
_DSL_TP_BER_STATE	0x23	Enables/Disables the DSL framer transmit PCM BER meter.
_DSL_RP_BER_STATE	0x24	Enables/Disables the DSL framer receive PCM BER meter.
_DSL_TP_BER_RESULTS	0x8C	Gets DSL framer transmit PCM BER meter status.
_DSL_RP_BER_RESULTS	0x8D	Gets DSL framer receive PCM BER meter status.
_DSL_TP_MAPPER_VALUE	0x30	For transmit PCM frame, determines which time slots are to be enabled for BER monitoring and selects the data source for these time slots.
_DSL_TP_MAPPER_WRITE	0x31	Writes the transmit PCM mapper values to the device.
_DSL_TP_MAPPER_READ	0xA3	Reads the transmit PCM mapper settings from the device.
_DSL_RP_MAPPER_VALUE	0x32	For receive PCM frame, determines which time slots are to be enabled for BER monitoring and selects the data source for these time slots.
_DSL_RP_MAPPER_WRITE	0x33	Writes the receive PCM mapper values to the device.
_DSL_RP_MAPPER_READ	0xA4	Reads the receive PCM mapper settings from the device.

12.4.1.3 Narrowband BER Meter API Commands

Table 12-6 lists a summary of commands used in Narrowband (NB) BER testing.

Table 12-6 Narrowband BER Meter API Commands

COMMAND	OPCODE	DESCRIPTION
_DSL_FR_TNB_RESET	0x4B	Resets the TX NB PCM PRBS generator and TX Narrowband PCM BER meter (TNB_BER).
_DSL_FR_RNB_RESET	0x4C	Resets the RX NB PCM PRBS generator and RX NB PCM BER meter (RNB_BER).
_DSL_TNB_BER_STATE	0x21	Enable/Disable the DSL framer transmit NB PCM BER meter.
_DSL_RNB_BER_STATE	0x22	Enable/Disable the DSL framer receive NB PCM BER meter.
_DSL_TNB_BER_RESULTS	0x91	Get DSL framer transmit NB PCM BER meter status.
_DSL_RNB_BER_RESULTS	0x92	Get DSL framer receive NB PCM BER meter status.
_DSL_TNB_MAPPER_VALUE	0x38	For transmit NB PCM frame, it determines which time slots are to be enabled for BER monitoring and selects the data source for these time slots.
_DSL_TNB_MAPPER_WRITE	0x39	Writes the transmit NB PCM mapper values to the device.
_DSL_TNB_MAPPER_READ	0xA7	Reads the transmit NB PCM mapper settings from the device.
_DSL_RNB_MAPPER_VALUE	0x3A	For receive NB PCM frame, it determines which time slots are to be enabled for BER monitoring and selects the data source for these time slots.
_DSL_RNB_MAPPER_WRITE	0x3B	Writes the receive NB PCM mapper values to the device.
_DSL_RNB_MAPPER_READ	0xA8	Reads the receive NB PCM mapper settings from the device.

12.4.2 BER Meter Configuration Sequence

- Configure the PCM or NB mapper to enable the BER meter and set the data source option to PRBS on the desired time slots.
- Configure the PRBS generator by issuing the _DSL_PRBS_CONFIGURE (0x25) API command.
- Enable the BER meter by issuing the appropriate BER_STATE API command.
- Poll the appropriate BER status command until the measurement phase is completed or failed.
- Read the appropriate BER meter results.

12.4.2.1 PCM BER Meter-API Sequencing Example

Table 12-7 provides an example for using the PCM BER meter. The ZipWirePlus modems are configured for G.shdsl mode. In this application, the PCM and DSL typically operate at the same data rate (minus DSL overhead). For this example, the PCM rate is set to 1536 kbps using the _DSL_MULTI_RATE_CONFIG (0x1B) API command. On the HTU-C, the DPLL operates in open loop mode. On the HTU-R, the DPLL is operating in closed loop mode. The HTU-R is also configured in a loopback mode with the _FR_HDSL_ON_PCM_LB option selected. For both the HTU-C and HTU-R, the transmit and receive PCM clocks are sourced from the DPLL.

Table 12-7 PCM BER Meter API Sequencing Example

Description		Opcode (HEX)	Parameter(s) [HEX]
System Enable		01	01 (HTU-C) 09 (HTU-R)
System Configuration		06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)		1B	18 00 18 18 01 00 00 00
Training Mode		03	12 00 (G.shdsl training / coded 16-PAM /Symmetric PSD) [default]
PCM Clock Source		12	0A (TPCLK = DPLL; RPCLK = DPLL) [default]
PCM Configuration		10	03 (HTU-C; DPLL Open; Asynchronous) 01 (HTU-R; DPLL Closed; Asynchronous) [Default handles HTU-C vs. HTU-R]
Activate (go)		0B	02 (fixed startup time-out, enable Act Request)
HTU-R	Loopback	09	0x0A (DSL framer DSL on PCM loopback)
Note: Wait until link is up before BER meter configuration:			
Framer set state machine		4A	00 (disable framer interrupt)
PRBS Configure		25	C8 (BER Scale: 2^{21}; PRBS Invert: Not Inverted; PRBS Source: Random Pattern; PRBS Data Pattern: 2^{23} - 1.)
TX PCM Mapper Value (HTU-C)		30	00 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 (For first 24 time slots: Data Source = PRBS, BER = Disable)
TX PCM Mapper Write (HTU-C)		31	00
RX PCM Mapper Value (HTU-C)		32	00 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 (For first 24 time slots: Data Source = RX PCM FIFO, BER = Enable)
RX PCM Mapper Write (HTU-C)		33	00
Framer TX PCM Reset		4E	01 (TP_PRBS_RESET = 1)
Framer Set State Machine		4A	01 (enable framer interrupt and re-run Framer State Machine)
Wait for Framer to Stabilize:			
Receive PCM BER State		24	01 (Enable)
Receive PCM BER Results		8D	00, Wait until Byte# 1 reads 0x05. Use bytes 2-3 to calculate average BER and bytes 4-5 for elapsed time.

12.4.2.2 NB BER Meter - API Sequencing Example

Table 12-7 provides an example for using the NB BER meter. The ZipWirePlus modems are configured for G.shdsl mode. In this application, the NB and DSL typically operate at the same data rate (minus DSL overhead). For this example, the NB rate is set to 1536kbps using the _DSL_NB_MULTI_RATE_CONFIG (0x1A) API command. The dsl data rate is set to be the same by _DSL_MULTI_RATE_CONFIG (0x1B). The HTU-R is also configured in a loopback mode with the _FR_HDSL_ON_NB_LB option selected.

Table 12-8 NB BER Meter API Sequencing Example

Description		Opcode (HEX)	Parameter(s) [HEX]
Configure the Modems for a PCM+NB Application (with NB and PCM rate set to 1536 kbps), Activate the Modems			
HTU-R	Loopback	09	0x0D (_FR_HDSL_ON_NB_LB loopback)
Note: Wait until link is up before BER meter configuration:			
Framer set state machine		4A	00 (disable framer interrupt)
PRBS Configure		25	C8 (BER Scale: 2^{21}; PRBS Invert: Not Inverted; PRBS Source: Random Pattern; PRBS Data Pattern: 2^{23}-1.)
TX NB Mapper Value (HTU-C)		38	00 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 (For first 24 time slots: Data Source = PRBS, BER = Disable)
TX NB Mapper Write (HTU-C)		39	00
RX NB Mapper Value (HTU-C)		3A	00 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 08 (For first 24 time slots: Data Source = RX PCM FIFO, BER = Enable)
RX NB Mapper Write (HTU-C)		3B	00
Framer TX NB PRBS Reset		4B	01 (TNB_PRBS_RESET = 1)
Framer Set State Machine		4A	01 (enable framer interrupt and re-run Framer State Machine)
Wait for Framer to Stabilize:			
Receive NB BER State		22	01 (Enable)
Receive NB BER Results		92	00, Wait until Byte# 1 reads 0x05. Use bytes 2-3 to calculate average BER and bytes 4-5 for elapsed time.

13.0 Microprocessor Communication Channel Protocol

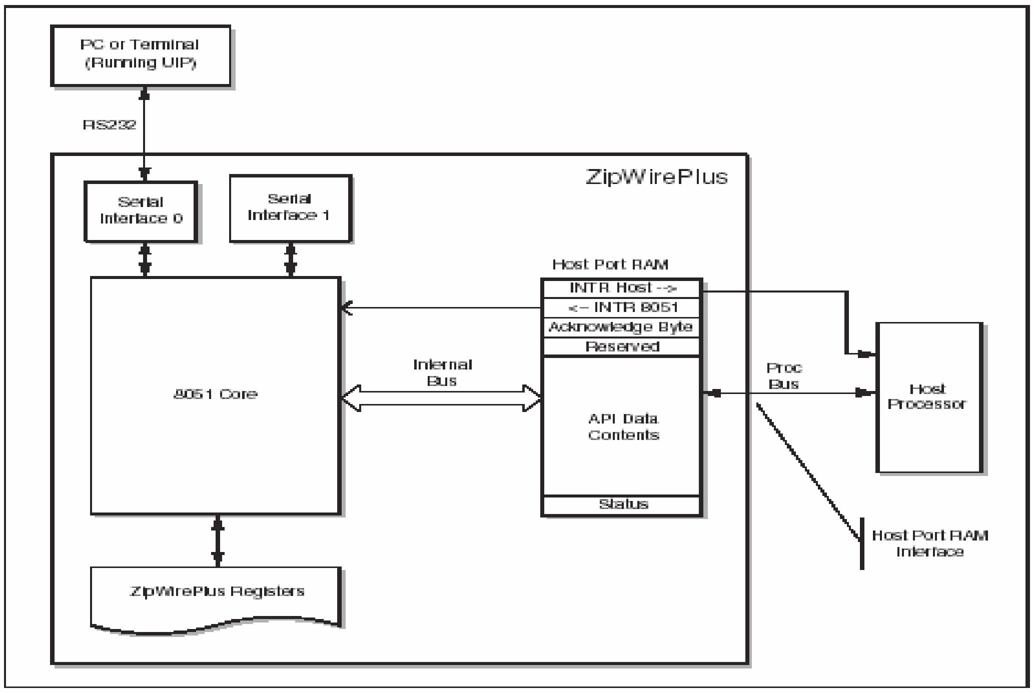
This section describes the microprocessor programming interface and communication protocol used by an external host processor to communicate with the ZipWirePlus device.

Application Programmer Interface (API) commands pass from an external processor to the ZipWirePlus embedded (8051) microprocessor via the following interfaces. Figure 13-1 illustrates the communication channel interfaces.

- Host Port RAM interface
- RS232 Serial Interface (UIP only)

The host port RAM interface is the primary channel used to communicate with the 8051. The RS232 serial interface is provided for use with the UIP(TestExec) and is intended for debug and development purposes only. Both the host and RS232 communication channels use the host port RAM for local storage and, therefore, only one communication channel can be used at a time.

Figure 13-1 ZipWirePlus Microprocessor (8051) Communication Channels



13.1 Peer-to-Peer Communication Protocol

A peer-to-peer protocol (which allows either the host or 8051 to initiate a message) is implemented. The host initiates API message transfer using the microprocessor communication channel protocol (host port RAM or RS232 serial Interface). The ZipWirePlus initiates API message transfer using the unsolicited interrupt protocol. One major advantage with the peer-to-peer protocol is that the 8051 processor can generate an asynchronous interrupt if a fatal error is detected. The ZipWirePlus status registers are also updated when a fatal error is detected. The host processor is expected to periodically poll the ZipWirePlus device for system and operational errors such as line quality, CRC errors, etc.

The peer-to-peer protocol is based on a request-response state machine where either the host processor or the ZipWirePlus device can be the requestor or initiator. The initiator cannot generate another request until the previous request has been serviced (responded to). The ZipWirePlus device can initiate a message before responding to a pending request from the host.

13.2 API Message Time-Out

The 8051 processor is guaranteed to locally process and respond to a status API command within 15 ms. The 8051 processor will process and respond to a control API command within 125 ms. When using the serial RS232 interface, each byte takes ~100 µs when running at 115 kpbs (87 µs to transfer the data plus 10 µs for the microprocessor to process). Therefore, the serial RS232 interface can add up to ~7 ms to complete certain API commands such as performance history dumps.

13.3 API Message Structure

The incoming and outgoing API message structures are similar for the host port RAM interface and RS232 serial interface communication channels. In general, API messages have a header section and a data section. The header section is five bytes and is of fixed length. The data section is of variable length and can go upto 75 bytes. The incoming message structure is described in Section 13.3.1 and the outgoing message structure is described in Section 13.3.2.

13.3.1 Incoming Message Structure

Table 13-1 illustrates the incoming message structure from the host processor to the 8051 processor. The incoming message is a variable-byte structure that consists of a header section and a data section. The header section is a fixed five-byte length and the data section is of variable length.

1. Header: Destination, Opcode, Reserved Byte, Length, and Header Checksum.
2. Data: Data Parameters and Data Checksum.

This message structure is the same for control and status request commands.

Table 13-1 Incoming Message Structure From the Host Processor

Header Section					Data Section	
Destination	Opcode	Reserved	Length	CS	Data Parameter	CS
1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	(Length + 1) Bytes	1 Byte

13.3.2 Outgoing Message Structure

Table 13-2 illustrates the outgoing message structure from the 8051 processor to the host processor. The outgoing message structure includes a header section and a data section. The third byte of the outgoing message header section is the acknowledge status byte.

For control commands, only the header section is sent from the 8051 processor to the host processor. For status request commands, both the header section and data section are sent from the 8051 processor to the host processor.

Table 13-2 Outgoing Message Structure From the 8051 Processor

Header Section					Data Section	
Destination	Opcode	ACK Status	Length	CS	Data Parameter	CS
1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	(Length + 1) Bytes	1 Byte

Control Command Response or Invalid Message						
Status Command Response						

13.3.3 Header Section

This section describes the header section bytes.

13.3.3.1 Header Section–Destination Field

The destination field selects the device to which the command is targeted. For the ZipWirePlus device this byte must be set to 0xF0.

13.3.3.2 Header Section–Opcode Field

The opcode field selects the specific control command or status request to be executed. The most significant bit (bit 7) determines if the opcode is a control or status request as shown in Table 13-3. See Chapter 15.0 for opcode values and descriptions.

Table 13-3 API Opcode Type

Bit # 7	Opcode Range (HEX)	Opcode Range (DEC)	API Type
0	0x00–0x7F	0–127	Control Command
1	0x80–0xFF	128–255	Status Request

13.3.3.3 Header Section—Reserved Byte or ACK Status

For incoming messages from the host processor, the third byte of the header is reserved for future expansion and to be compatible with the outgoing message. For incoming messages, the reserved byte must be set to 0x00.

For outgoing message this field contains the Acknowledge Status (ACK Status) byte.

Acknowledge Status Byte

For each control and status request message received and processed from the host processor, the 8051 processor generates an acknowledge status byte. This is used by the host uses for error checking. Table 13-4 shows the bit definition for the Acknowledge Status register.

Table 13-4 Acknowledge Status Register (Interrupt Source Register)

Bit 7	Bit 6	Bit 5	Bit 4:0
Unsolicited Interrupt	Reserve d	API Response	Acknowledge Status

Unsolicited Interrupt

The Unsolicited Interrupt bit (bit 7) of the Acknowledge Status register is set when the API outgoing message is caused by an unsolicited interrupt. To determine the source of the unsolicited interrupts, read the STATUS_8 byte. This is located at offset 0xC37 in the Host Port RAM or can be read by invoking the _DSL_STATUS (0x85) API command.

API Response

The API Response bit (bit 5) is set when the 8051 processor responds to a host API command. The API Response bit is also set when the 8051 processor triggers a boot ROM wake-up (_ACK_BOOT_WAKE_UP) or operational code wake-up interrupt (_ACK_OPER_WAKE_UP).

NOTE: The API response and wake-up are mutually exclusive events and share the same bit field.

Acknowledge Status

The Acknowledge Status field (bits 4:0) provides the status code of the current API command (see Table 13-5).

Table 13-5 Acknowledge Status Codes

Status	Description	Value	#define (as defined in DSL_API.H)
Not Completed	Current API command has not been completed or no API command was sent to this device. This is used when multiple INTR_HOST interrupts are ORed together to the host processor.	0x00	_ACK_NOT_COMPLETE
API Successful	No error. Command was successfully completed. Any status data is valid.	0x01	_ACK_PASS
Busy	8051 busy, unable to process command. The host should resend command after approximately 500 ms.	0x02	_ACK_BUSY
Not Applicable	The API command is not applicable to the current H/W or S/W configuration.	0x03	_ACK_NOT_APPLICABLE
Invalid Destination	An invalid destination was specified.	0x04	_ACK_INVALID_DEST
Invalid Opcode	An invalid opcode was specified.	0x05	_ACK_INVALID_OPCODE
Invalid Length	An invalid length was specified.	0x06	_ACK_INVALID_LENGTH
Invalid Data	One or more invalid data parameters detected.	0x07	_ACK_INVALID_DATA
Invalid Checksum	One or more invalid Checksum parameters specified.	0x08	_ACK_INVALID_CHKSUM
No Result	The 8051 was unable to complete the specified status request. No results are returned.	0x09	_ACK_NO_RESULT
Not Available	The ZipWirePlus command is not available for this image or the link is not up (EOC)	0x0A	_ACK_NOT_AVAILABLE
Boot ROM Wake-up	The 8051 boot ROM has successfully initialized and is awaiting the host download. This code is only set when the PRAM download is selected from host port RAM.	0x0D	_ACK_BOOT_WAKE_UP
Operational Code Wake-up	The 8051 has successfully reached the operational code.	0x0E	_ACK_OPER_WAKE_UP
Reserved	These codes are reserved for future expansion of the 8051 code.	0x0B, 0x0C, 0x0F–0x1F	—

13.3.3.4 Header Section-Message Length Field

The message length field is a value from 0–74 to provide the number of bytes in the data parameters field (1–75 bytes). The number of bytes in the data field is always at least 1, so a 0 length specifies that one byte is in the data field; thus, the length is equal to the number of bytes in the data parameter plus one.

13.3.3.5 Header Section-Checksum

This byte must carry a one-byte checksum of the header when issuing commands to the boot code or when used with the RS232 communication channel. This byte is ignored by the ZipWirePlus embedded processor when using the host port communication channel. The header checksum is computed as described in the following Header API Checksum Function paragraph.

Header API Checksum Function

The host computes the header checksum value as shown below. The header checksum is sent as the fifth byte of the header section. The checksum byte is not used in the checksum computation. The checksum value is calculated using the following formula:

$$CS = (Byte\#1) \oplus (Byte\#2) \oplus (Byte\#3) \oplus (Byte\#4) \oplus (0xAA)$$

Where ⊕ denotes a bit-wise exclusive-OR operation, and 0xAA is the binary byte 10101010. The same rule is used by the 8051 processor to calculate the header checksum byte of the outgoing message sent to the host processor.

When the ZipWirePlus is executing the boot code, the outgoing message header checksum value is calculated as shown below. The API response (bit 5 of the Acknowledge Status register) is set when the ZipWirePlus is responding to an API command.

$$CS = (Byte\#1) \oplus (Byte\#2) \oplus (Byte\#3) \oplus (Byte\#4) \oplus (0xAA) \mid API\_RESPONSE$$

NOTE: The checksum is required when using the host port RAM interface to maintain backwards compatibility with legacy products.

13.3.4 Data Section

This section describes the data section of the incoming or outgoing message.

13.3.4.1 Data Section—Data Parameter Field

For incoming commands, the data parameter field provides additional data (or parameters) for the given API command. Because the incoming length is 0-based, every API command has at least one byte for the data parameter. In commands where there is no need for additional data, 0x00 should be placed as the data byte to ensure future compatibility.

NOTE: For invalid parameters, the software typically uses the default values in addition to returning the code for Invalid Data (_ACK_INVALID_DATA) for Acknowledge Status field.

For outgoing status request commands, the data parameter field provides the results for the given API command.

13.3.4.2 Data Section-Checksum

This byte must carry a one-byte checksum of the data section when issuing commands to the boot code or when used with the RS232 communication channel. For incoming messages, this byte is ignored by the ZipWirePlus embedded processor when using the host port communication channel. The data checksum is computed as described in the following Data Section API Checksum Function paragraph.

Data Section API Checksum Function

The host processor shall compute the data checksum value as shown below. The data checksum value is sent as the last byte of the message. The checksum byte is not used in the checksum computation. The checksum value is calculated using the following formula:

$$CS = (Byte\#1) \oplus (Byte\#2) \oplus (Byte\#3) \oplus .... \oplus (Byte\# N) \oplus (0xAA)$$

Where ⊕ denotes a bit-wise exclusive-OR operation, and 0xAA is the binary byte 10101010. The same rule is used by the 8051 processor to calculate the data checksum byte of the outgoing message sent to the host processor.

NOTE: The checksum is required when using the host port RAM interface to maintain backwards compatibility with legacy products.

13.4 Host Port RAM Interface Protocol

The host port RAM is a dual port RAM so both processors can read and write to the RAM. Registers 0x000 (INTR_HOST) and 0x001 (INTR_8051) are special registers that behave uniquely. Table 13-6 lists the mapping for the host port RAM and provides a brief description of the register function.

Table 13-6 Host Port RAM Mapping

Register	Label	Description
0x000	INTR_HOST	Writing to this register generates an interrupt to the host processor. Reading this register clears the interrupt.
0x001	INTR_8051	Writing to this register generates an interrupt to the 8051. Reading this register clears the interrupt. The host processor only writes to this register, while the 8051 only reads this register.
0x002	Host Port RAM MAP Version	Host Port Version ID. The host software would use this ID to determine the host port RAM address mapping and protocols.
Incoming API Message Structure		
0x003	API In Destination	Incoming API destination field.
0x004	API In Opcode	Incoming API opcode Field.
0x005	Reserved	Reserved. Must be set to 0x00.
0x006	API In Length	Incoming API data length. Specifies the number of API data bytes for incoming API commands. The length field is 0-based. A 0x00 represents 1 byte; the maximum is 75 bytes (value of 74).
0x007	Reserved	Reserved for RS232 protocol. Value ignored when connected to host port RAM.
0x008–0x052	API In Data	Incoming API data—up to 75 bytes.
0x053	API In Data Checksum	Incoming API message data checksum.
Outgoing API Message Structure		
0x054	API Out Destination	Outgoing API destination field.
0x055	API Out Opcode	Outgoing API opcode Field.
0x056	API Out Status Acknowledge	Acknowledge byte. When the INTR_HOST is detected, the host processor queries this byte to determine what further action is needed.
0x057	API Out Length	Outgoing API Data Length. Specifies the number of API Data bytes for outgoing API commands. A 0 represents 1 byte; the maximum is 75 bytes (value of 74). For control commands and un-successful status acknowledges, this value is set to 0.
0x058	Reserved	Reserved for RS232 protocol. Value ignored when connected to host port RAM.
0x059–0x0A3	API Out Data	Outgoing API data—up to 75 bytes.
0x0A4	Reserved	Reserved for RS232 protocol. Value ignored when connected to host port RAM.

Register	Label	Description
0x0A5–0x1E8	Reserved	—
0x1E9-0x244	E1 Framer Block	Please refer to Section 9.3.1 for details on the structure of this block. This is valid for M28947 device only.
0x245-0x3BF	Reserved	
Status Information		
0x3C0	STATUS_1	See _DSL_STATUS (0x85) API Command (STATUS_1 byte)
0x3C1	STATUS_2	See _DSL_STATUS (0x85) API Command (STATUS_2 byte)
0x3C2	STATUS_3	See _DSL_STATUS (0x85) API Command (STATUS_3 byte)
0x3C3	STATUS_4	See _DSL_STATUS (0x85) API Command (STATUS_4 byte)
0x3C4	STATUS_5	See _DSL_STATUS (0x85) API Command (STATUS_5 byte)
0x3C5	STATUS_6	See _DSL_STATUS (0x85) API Command (STATUS_6 byte)
0x3C6	STATUS_7	See _DSL_STATUS (0x85) API Command (STATUS_7 byte)
0x3C7	STATUS_8	See _DSL_STATUS (0x85) API Command (STATUS_8 byte)
0x3C8–0x3FF	Reserved	Reserved

13.4.1 INTR_HOST and INTR_8051 Registers

When register 0x000 or 0x001 is written to, an interrupt is generated. When register 0x000 or 0x001 is read, the interrupt is cleared. The hardware interrupts are active-low. The INTR_HOST (0x000) and INTR_8051 (0x001) signal the other processor that data is available in the host port RAM.

13.4.2 Host Port Acknowledge Register

Register 0x056 (Acknowledge Status) is the API acknowledge status byte. Whenever the host processor detects the INTR_HOST interrupt, the host must query the acknowledge byte to determine the course of action. The acknowledge status byte can be viewed as an Interrupt Service Register (ISR).

The all-0s (_ACK_NOT_COMPLETE) value is used in a multiple-device environment where the INTR_HOST interrupt lines are ORed together to the host processor. When the INTR_HOST is detected, the host processor polls each device's acknowledge byte to determine where the interrupt was generated.

The host processor is responsible for clearing the acknowledge register before exiting its INTR_HOST interrupt handler. This ensures the acknowledge byte is 0x00 (_ACK_NOT_COMPLETE) in the event that another device generates a separate INTR_HOST request.

13.4.3 Host Port Status Registers

Registers 0x3C0–0x3C7 contain the STATUS_N bytes for the ZipWirePlus device. Reading this register directly eliminates the inefficiency of the API message protocol when the host processor must query for common status during normal operation. Each device has up to eight status bytes. The host processor should never write to the STATUS_N registers because the 8051 processor uses this RAM as the storage location for the status information. Writing to these registers could corrupt the status information and cause unpredictable behavior. Reading the STATUS_N register returns the same result as the _DSL_STATUS (0x85) API command.

13.4.4 Host Port RAM Interface Sequence Of Events

The host processor writes to the host port RAM based on the desired API command, then writes 0xFE to the INTR_8051 to signal the 8051 that an API command is ready. The 8051 interrupt handler sets a flag to process the API command. The API command is processed in the main thread when the 8051 processor has time available. After the 8051 processor processes the API command and writes the results into the host port RAM, the 8051 processor writes to the INTR_HOST register to acknowledge to the host that the API command is complete. The host processor can now read the acknowledge status register and any valid status information. The host processor then must write a 0x00 into the ACK_STATUS register and the INTR_HOST register. The host processor then reads from INTR_HOST to clear the interrupt. This completes the API command.

From the time the host processor writes to the INTR_8051 and until the INTR_HOST is detected, the host processor must not write to the host port RAM incoming API fields. Writing to the host port RAM could corrupt the current API command. Additionally, the host processor only reads the host port RAM outgoing API fields after the INTR_HOST is detected. Reading the host port RAM before the INTR_HOST is detected does not corrupt any data, but the contents of the data will be invalid. The host processor writes a 0x00 to the INTR_HOST then subsequently reads the INTR_HOST register to clear the interrupt only after the API status results are read and the acknowledge byte is written to 0x00.

Table 13-7 illustrates the host port RAM sequence of events. The <Processor>-Main indicates this task is accomplished in the main thread while the <Processor>-ISR indicates this task is accomplished in the interrupt service handler.

NOTE: Host implementation is application and processor specific and may differ depending on the host processor environment.

Table 13-7 Host Port RAM Message Protocol Events

Processor	Action
Host-Main	Write API content registers
Host-Main	Write to INTR_8051 register to initiate message (0x001) to 0xFE
8051	Detect INTR_8051
8051-ISR	Read API content registers
8051-ISR	Write INTR_8051 to 0x00
8051-ISR	Clear INTR_8051 interrupt by reading from the INTR_8051 register
8051-Main	Perform API task
8051-Main	If status request, write results into host port RAM
8051-Main	Write INTR_HOST register to initiate acknowledge (0x000) to 0xFE
8051-Main	Write acknowledge status byte
Host	Detect INTR_HOST
Host-ISR	Read acknowledge
Host-ISR	If status request, read API data length and data registers to processes results
Host-ISR	Write INTR_HOST to 0x00
Host-ISR	Clear INTR_HOST interrupt by reading from the INTR_HOST register
Host-ISR	Write the acknowledge status byte register to 0x00
Host-Main	Host is clear to send another message

In multitasking environments, the host semaphore prevents the host from sending any additional commands until the current command is completed. If the communication channel is unavailable to send another command, the host processor could put the task to sleep and switch to process other tasks. When the INTR_HOST is detected and the data is processed, the task can be awakened and processed.

13.4.5 Host Processor Polling Method

In non-multitasking environments, simply poll and wait until INTR_HOST is written to. This is accomplished by polling the INTR_HOST register and looking for a 0xFE.

13.4.6 Multi-Device System

In a multi-ZipWirePlus device system (see ZipWirePlus data sheet), the INTR_HOST interrupts lines would be wire-ORed together. The host processor must be able to process and send back acknowledgements, at the same time, from multiple ZipWirePlus devices.

The _ACK_NOT_COMPLETE acknowledge status code (see Table 13-5) is critical in multi-ZipWirePlus device systems. When the host processor detects the INTR_HOST interrupt, the host processor polls the acknowledge status request of all the devices to determine which device generated the interrupt.

13.5 Unsolicited Interrupt

13.5.1 Unsolicited Interrupt Overview

The host port RAM interface allows the 8051 to generate unsolicited interrupts to the host processor when certain events occur, as summarized in the following list:

- Reaching Normal Operation—this includes successful end-to-end training as well as certain loopbacks and test modes. This is also called link up state.
- Deactivating—dropping the link. This is called link down state.
- EOC Events
- Receive an incoming message
- Successfully transmitted message
- EOC error—such as receive queue overflow, invalid receive CRC, etc.

Other Features

- Each event can be masked.
- The 8051 will hold off generating a new interrupt until the host processor has completed its interrupt handler (by clearing the Acknowledge Status register).
- The host is required to issue successive API commands at least 100 ms apart. This is required because unsolicited interrupts cannot tolerate continuous input API stream.

13.5.2 Interrupt Sources

The cause of an unsolicited interrupt can be determined by looking at the STATUS_8 byte (Offset: 0x3C7) of the dual port RAM. Table 13-8 shows the bit definition for STATUS_8 byte. In general, unsolicited interrupts can be generated due to change in state of the Activation State Manager or due to API commands issued by the 8051 to the host processor.

Figure 13-2 Unsolicited Interrupt Sources

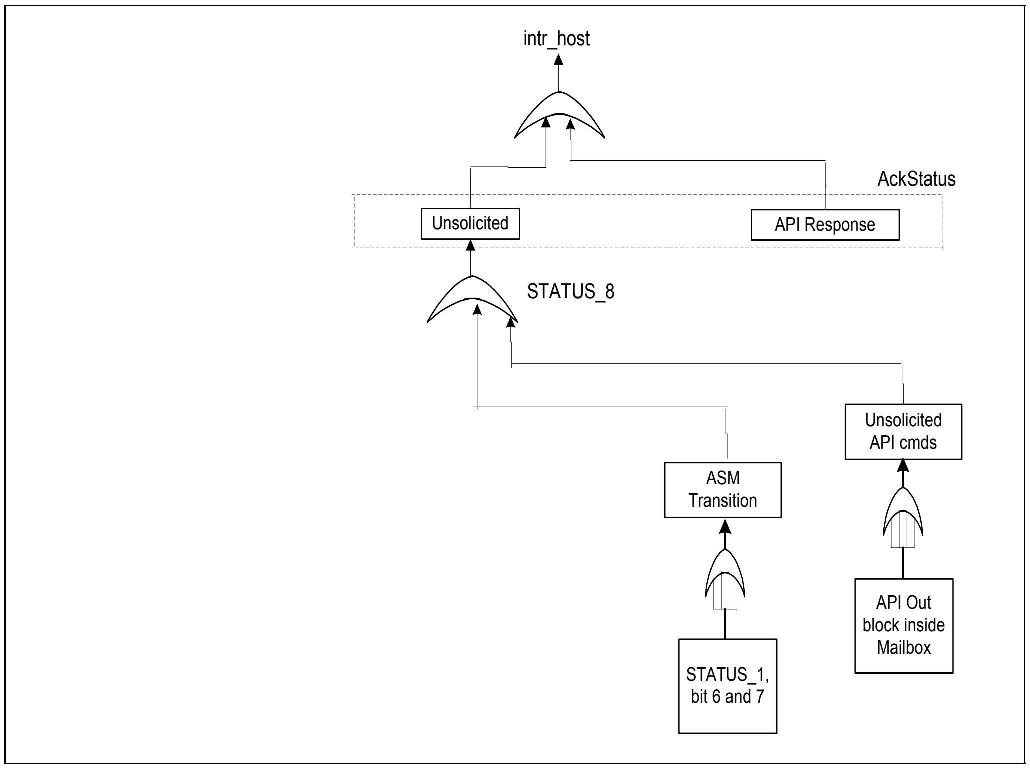


Table 13-8 STATUS_8: Unsolicited Interrupt Source Bit Definition

Bit 7:4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	Dying Gasp	Reserved	ASM Transition	Unsolicited API Command

Unsolicited API Commands

This bit is used to indicate an unsolicited interrupt due to an API command from the 8051 to the host processor. This bit is set for all API commands that generate an unsolicited interrupt.

The following API commands can generate an unsolicited interrupt.

Received a EOC message API -- _EOC_RX_GET_MSG (0xB1)

Read Z-bits API -- _DSL_READ_ZBITS (0xD1) (available in M28950 only)

E1 framer Transmit Path Change - _E1_PRA_TX_MON_CHANGE (0xC0) (available in M28947 only)

E1 framer Receive Path Change - _E1_PRA_RX_MON_CHANGE (0xC1) (available in M28947 only)

ASM Transition

This bit is set (1) when the ASM transitions into the Active State (normal operation) or when the ASM transitions from the Pending State to the Deactivated State. The host processor reads the STATUS_1 to determine the link-up or link-down status. This bit is also set when certain loopbacks or test modes are completed, such as ERLE or Analog Loopback.

In summary, this bit provides link-down to link-up transition or link-up to link-down transition.

Dying Gasp

This bit is set to indicate that the HTU-R has a power failure. When HTU-R has a power failure, HTU-R sends power status indicator bits (active low) to HTU-C. HTU-C will report dying gasp unsolicited interrupt to the host.

13.5.3 Events and the Service Routines for Unsolicited Interrupts

Table 13-9 Unsolicited Interrupt Events and Service Routines

Events	8051 Action	Interrupt Service Routine (Host)
LINK UP/DOWN	Set status_8 bit 1 Write to status_1 Generate interrupt to Host	Read status_1, bit 6 and 7 Clear interrupt Write 0x00 to acknowledge status byte
_EOC_RX_GET_MSG (0xB1)	Set status_8 bit 0 Write to API Out (with opcode 0xB1) Generate interrupt to Host	Read information from API Out Clear interrupt Write 0x00 to acknowledge status byte
_DSL_READ_ZBITS (0xD1)	Set status_8 bit 0 Write to API Out (with opcode 0xD1) Generate interrupt to Host	Read information from API Out Clear interrupt Write 0x00 to acknowledge status byte
_E1_PRA_TX_MON_CHANGE (0xC0)	Set status_8 bit 0 Write to API Out (with opcode 0xC0) Write to Host Port RAM (offset 0x1E9) Generate interrupt to Host	Read Opcode from API Out Read information from Host Port RAM (offset 0x1E9) Clear interrupt Write 0x00 to acknowledge status byte
_E1_PRA_RX_MON_CHANGE (0xC1)	Set status_8 bit 0 Write to API Out (with opcode 0xC1) Write to Host Port RAM (offset 0x1E9) Generate interrupt to Host	Read Opcode from API Out Read information from Host Port RAM (offset 0x1E9) Clear interrupt Write 0x00 to acknowledge status byte

NOTE: The Unsolicited Mechanism for E1 Framer Feature is different from the rest of the APIs. For the APIs, _E1_PRA_TX_MON_CHANGE (0xC0) and _E1_PRA_RX_MON_CHANGE (0xC1), the host on receiving the Unsolicited interrupt will only read the Opcode of the API from API Out. It will read the data from the E1 Framer block in Host Port RAM (Offset 0x1E9). These APIs are supported on M28947 device only.

13.5.4 Protocol Event Sequence for Unsolicited Interrupts

Table 13-10 Unsolicited Interrupt Protocol Sequence for Link Up/Down Events

Processor	Action
8051	Detect changes in Activation State Manager, update STATUS_1
8051	Set STATUS_8 bit 1 to 1
8051	Write INTR_HOST register to 0xFE to generate an interrupt
8051	Write Acknowledge Status register with bit 7 set to indicate unsolicited interrupt
Host	Detect INTR_HOST
Host	Read ACK Status register to determine if it is an unsolicited interrupt
Host	Read STATUS_8, and find out bit 1 is 1

Processor	Action
Host	Read STATUS_1 and process results
Host	Clear and write INTR_HOST to 0x00
Host	Write the ACK status register to 0x00

Table 13-11 Unsolicited Interrupt Protocol Sequence for API Commands

Processor	Action
8051	Detect any changes in Rx EOC or and other API command events
8051	Set STATUS_8 bit 0 to 1
8051	Fill API Out block with related information and set Opcode to the appropriate API value
8051	Write INTR_HOST register to 0xFE to generate an interrupt
8051	Write Acknowledge Status register with bit 7 set to indicate unsolicited interrupt
Host	Detect INTR_HOST
Host	Read ACK Status register to determine if it is an unsolicited interrupt
Host	Read STATUS_8, and find out bit 0 is 1
Host	Read API Out Opcode, data to process results
Host	Clear and write INTR_HOST to 0x00
Host	Write the ACK status register to 0x00

Table 13-12 Unsolicited Interrupt Protocol Sequence for E1 Framer API Commands

Processor	Action
8051	Detect any changes in Rx EOC or and other API command events
8051	Set STATUS_8 bit 0 to 1
8051	Fill API Out block with the Opcode
8051	Fill E1 Framer block(offset:0x1E9) with related information
8051	Write INTR_HOST register to 0xFE to generate an interrupt
8051	Write Acknowledge Status register with bit 7 set to indicate unsolicited interrupt
Host	Detect INTR_HOST
Host	Read ACK Status register to determine if it is an unsolicited interrupt
Host	Read STATUS_8, and find out bit 0 is 1
Host	Read API Out Opcode, data to process results
Host	Read data from the E1 Framer block in Host Port Ram(Offset:0x1E9)

Processor	Action
Host	Clear and write INTR_HOST to 0x00
Host	Write the ACK status register to 0x00

13.5.5 Interrupt Mask/Enable Support

There are two levels for interrupt control. Using the _DSL_INTR_HOST_MASK (0x50) API command, unsolicited interrupts due to API commands can be enabled or disabled globally. This API is also used to enable or disable unsolicited interrupts due to Activation State Machine transitions and Dying Gasp event.

The _DSL_INTR_API_SUBMASK (0x51) API command is used by the host to selectively enable or disable certain events from generating an unsolicited interrupt. This is accomplished by enabling or disabling the API commands that are allowed to generate an unsolicited interrupt.

13.6 RS232 Serial Interface Protocol

The RS232 serial interface protocol is an asynchronous serial channel that allows an external PC (or terminal) to communicate with the ZipWirePlus chipset via an RS232 connector.

13.6.1 Host Processor to 8051 Processor Message Structure

All RS232 messages sent from the host processor to the 8051 processor are contained in a variable byte structure that must be at least seven-bytes long, as shown in Table 13-13. The message structure contains a variable byte structure that consists of two sections (the header and the data) with the following fields:

1. Header: Destination, Opcode, a Reserved Byte, Length, and Checksum
2. Data: Data Parameters and Checksum.

The header and data sections contain a required checksum to guarantee the message packet transfer. See Section 13.3.3.5 and Section 13.3.4.2 for details regarding the checksum computation.

NOTE: The upper four-bits of the destination field are set to all 1s, which implies the destination field has the following format: 0xF<destination>. This is used to indicate the start of a message.

Table 13-13 Host Processor To 8051 Processor RS232 Message Structure

Header Section					Data Section	
Destination	Opcode	Reserved	Length	CS	Data Parameter	CS
1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	(Length + 1) Bytes	1 Byte

13.6.2 RS232 Acknowledge Message Structure

The 8051 processor acknowledges all valid RS232 messages (control and status requests) with a five-byte long acknowledge message, as shown in Table 13-14.

Table 13-14 RS232 Acknowledge Response Message Structure

Acknowledge Response Packet				
Destination	Opcode	ACK Status	Length	CS
1 Byte	1 Byte	1 Byte	1 Byte	1 Byte

The checksum value is calculated according to the formula described in Section 13.3.3.5.

13.6.3 ZipWirePlus 8051 Processor to Host Processor Status Message Structure

For status request commands, the 8051 processor sends the variable length results after the acknowledge response. The data results are only sent if the acknowledge status byte was successful. Table 13-15 illustrates the message structure format.

The destination and opcode fields match what the 8051 received. The number of bytes is equal to the length plus 1, i.e., a length of 0 equals one byte and a length of 74 equals 75 bytes. The length does not include the checksum byte.

Table 13-15 8051 Processor to Host Processor RS232 Message Structure

Acknowledge Response Section (Header)					Data Section	
Destination	Opcode	ACK Status	Length	CS	Data Parameter(s)	CS
1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	(Length + 1) Bytes	1 Byte

Control Command Response or invalid message	
Status Command Response	

The host processor first reads the API data length to determine the number of the status bytes. The number of status bytes is then read. The final byte (checksum) is then read to complete the message.

13.6.4 RS232 Message Transfer Protocol

The application sends a command to any of the devices in the system by transmitting a message over the serial communication channel. Every command that is correctly received and decoded by the 8051 processor is acknowledged by sending a special acknowledge message back to the application. In response to a status request command, the 8051 processor also sends a status response message containing the information requested.

The 8051 processor is guaranteed to acknowledge a received message within a specified time (see Section 13.2). The host processor retransmits messages that are not acknowledged within this time limit. The host processor should send no new messages before the previous one was acknowledged unless the time limit has been exceeded. When the 8051 processor receives a status request command, it responds (after acknowledging the command) by sending a status message to the host processor.

The following examples illustrate the 8051 processor sequence of events. In the examples, all commands are sent to device number 0 (destination = 0).

Example 1: Set the _DSL_SYSTEM_CONFIG (0x01) to 0x01 (In Service, HTU-C).

Table 13-16 Example 1—Incoming RS232 Message

Header Section					Data Section	
Destination	Opcode	Reserved	Length	CS	Data 0	CS
0xF0	0x01	0x00	0x00	0x5B	0x01	0xAB

Table 13-17 Outgoing RS232 Message

Header Section				
Destination	Opcode	ACK	Length	CS
0xF0	0x01	0x21	0x00	0x7A

Example 2: Query the _DSL_STATUS (0x85); assume all return bytes are 0.

Table 13-18 Example 2—Incoming RS232 Message

Header Section					Data Section	
Destination	Opcode	Reserved	Length	CS	Data 0	CS
0xF0	0x85	0x00	0x00	0xDF	0x00	0xAA

Table 13-19 Example 2—Outgoing RS232 Message

Header Section					Data Section			
Destination	Opcode	ACK	Length	CS	Data 0	...	Data 7	CS
0xF0	0x85	0x21	0x07	0xF9	0x00	...	0x00	0xAA

Example 3: Processor Busy during query the _DSL_STATUS (0x85). No data results are returned.

Table 13-20 Example 3—Incoming RS232 Message

Header Section					Data Section	
Destination	Opcode	Reserved	Length	CS	Data 0	CS
0xF0	0x85	0x00	0x00	0xDF	0x00	0xAA

Table 13-21 Example 3—Outgoing RS232 Message

Header Section				
Destination	Opcode	ACK	Length	CS
0xF0	0x85	0x22	0x00	0xFD

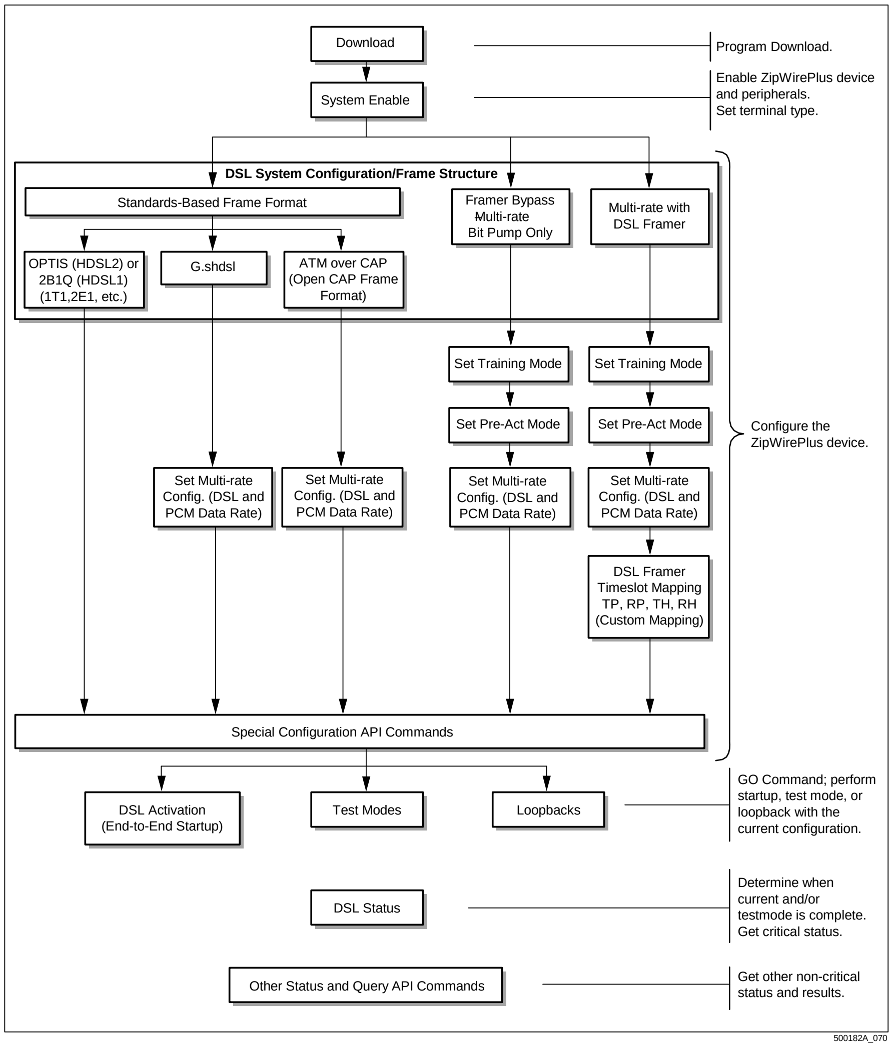
14.0 ZipWirePlus API Configuration

The ZipWirePlus firmware is extremely flexible and provides an extensive set of API commands. This chapter simplifies the API command usage by grouping and sequencing APIs commands used in different applications.

14.1 API Command Sequencing Overview

Figure 14-1 illustrates the API command sequence required to configure the ZipWirePlus devices for different applications.

Figure 14-1 API Command Sequencing


Download
System Enable
Program Download.
Enable ZipWirePlus device and peripherals. Set terminal type.
DSL System Configuration/Frame Structure
Standards-Based Frame Format
Framer Bypass ~~Multi-rate~~ Bit Pump Only
Multi-rate with DSL Framer
OPTIS (HDSL2) or 2B1Q (HDSL1) (1T1,2E1, etc.)
G.shdsl
ATM over CAP (Open CAP Frame Format)
Set Training Mode
Set Training Mode
Set Pre-Act Mode
Set Pre-Act Mode
Set Multi-rate Config. (DSL and PCM Data Rate)
Set Multi-rate Config. (DSL and PCM Data Rate)
Set Multi-rate Config. (DSL and PCM Data Rate)
Set Multi-rate Config. (DSL and PCM Data Rate)
DSL Framer Timeslot Mapping TP, RP, TH, RH (Custom Mapping)
Configure the ZipWirePlus device.
Special Configuration API Commands
DSL Activation (End-to-End Startup)
Test Modes
Loopbacks
GO Command; perform startup, test mode, or loopback with the current configuration.
DSL Status
Determine when current and/or testmode is complete. Get critical status.
Other Status and Query API Commands
Get other non-critical status and results.
500182A_070

14.1.1 Special Sequencing Order Requirements

Only the Reset, System Enable, System Configuration, Multirate, and Narrowband Multirate API commands have to be in a particular order and place. These commands must be done first and in the order listed in Table 14-1. No other API commands have ordering requirements.

Table 14-1 Special Case API Sequencing Requirements

Description	Opcode (HEX)	Parameter(s) [HEX]
Reset	00	00 (not necessary)
System Enable	01	01 (HTU-C) 09 (HTU-R)
System Configuration	06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)	1B	N 00 N N 01 I 00 00

Description	Opcode (HEX)	Parameter(s) [HEX]
Other API commands	N/A	N/A

14.1.2 Multirate Configuration

The multirate configuration API commands allow the application to customize the DSL and PCM data rates. The multirate configuration has no interaction with other devices, except that the devices share a common PCM bus. More specifically, the ZipWirePlus devices do not handle loop reversal or switching of the master loop. The multirate configuration API commands are described in Table 14-2.

Table 14-2 Multirate Configuration API Commands

Parameter	Description
Number of PCM Time Slots	Total number of PCM time slots available. A value of 1 implies 1 time slot. The low byte is programmed first. X = 1…128
Number of DSL Time Slots	Total number of DSL time slots available. A value of 1 implies 1 time slot. X = 1…89
Number of Occupied PCM Time Slots	Total number of occupied time slots available. A value of 1 implies 1 time slot. X = 1…MIN (Total PCM, Total DSL)
Start PCM Time Slot Location	Indicates the location of the first time slot to extract from the PCM bus. The first PCM time slot is numbered 1.
Number of i-bits	Total number of i-bits available. X = 0…7
Mapping Format	0 = Block mapping 1 = Interleave mapping
Interleave Ratio	Determines the interleave ratio on the PCM bus. An interleave ratio of 0x01 implies that all PCM timeslots will be used. A value of 0x02 implies using every other time slot. A value of 0x00 is invalid. This value is only applicable when the Mapping Mode is set to the Interleave Mapping option.

From this API command, the ZipWirePlus operational code can determine the data rate and mapping information as follows:

- DSL Data Rate: (Total number of DSL time slots x 64 k) + 8 k + (i x 8 k); where 8 k is the fixed overhead data and the (i x 8 k) is the additional overhead. For HDSL2 (OPTIS) and HDSL1, the additional overhead (i) is always 1. For G.shdsl, the additional overhead can be from 0-7.
- PCM Data Rate: (Total number of PCM time slots x 64 k)
- DSL Mapping: The number of occupied time slots are mapped into the first N payload bytes of the DSL channel. If the total number of DSL time slots exceeds the number of occupied time slots, the unused DSL time slots are filled with DBANK1 contents.
- PCM Mapping: Setting the mapping format parameter to block mapping causes consecutive time slots on the PCM bus to be used for transfer. Setting the mapping format parameter to interleave mapping will cause every M^{th} time slot to be used for transfer where M is specified through the interleave ratio parameter. The interleave ratio parameter is ignored when the mapping format is set to block mapping.

14.2 Basic API Sequencing Application Examples

This section provides the basic API sequencing and control commands required to configure the ZipWirePlus.

14.2.1 Minimum API Commands For End-to-End Data Transfer

Table 14-3 lists the minimum API commands required to configure the ZipWirePlus for end-to-end data transfer. These assumptions are made for this configuration:

- ZipWirePlus operational code has its default setting
- Targeted for EVM connected to Fireberd LAB module
- DSL data rate = 1552 kbps
- ATM disabled
- ZipWirePlus provides master PCM clock–where both TPCLK and RPCLK are sourced from DPLL
- HTU-C DPLL runs in open loop mode
- HTU-R DPLL runs in closed loop mode
- PCM data rate tracks DSL track (excluding DSL overhead)

Table 14-3 Minimum API Sequencing Requirements

Description	Opcode (HEX)	Parameter(s) [HEX]
System Enable	01	01 (HTU-C) 09 (HTU-R)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)

14.2.2 General API Sequence

The following sequence of API commands should be used to activate the ZipWirePlus from a power-on or reset as shown in Figure 14-1. This assumes the PRAM (Program RAM) contents have already been downloaded to the ZipWirePlus device.

Issue the _DSL_SYSTEM_ENABLE (0x01) command; set the appropriate terminal type.

1. Issue the _DSL_SYSTEM_CONFIG (0x06) command.
2. Depending on the system configuration, issue the appropriate multirate config, training mode, etc.
3. Issue any other application specific API commands.
4. Issue the _DSL_ACTIVATION (0x0B) command; enable the Activation Request State.
5. Repeatedly issue the _DSL_STATUS (0x85) command to determine when the modem has successfully trained.

14.2.3 Deactivating the ZipWirePlus System

The following sequence of API commands should be used to deactivate the ZipWirePlus and to have it remain in an idle state (transmitter off).

1. Disable ASM in the _DSL_ACTIVATION (0x0B) command; the parameter should be set to 0x00.
2. Issue the _DSL_FORCE_DEACTIVATE (0x0C) command; the parameter should be set to 0x01.

This is useful to detect a higher-level error and to disable any retraining in order to perform diagnostics or troubleshooting of the system. The _DSL_SYSTEM_ENABLE (0x01) API command set to 0x00 (Out-of-service) could also be used; however, that would tri-state clocks and might make the system unsuitable for debugging.

14.2.4 Retrain the ZipWirePlus Modems

Issuing the _DSL_FORCE_DEACTIVATE (0x0C) command while the ASM is enabled forces the modems to retrain.

14.2.5 Setting the ZipWirePlus System to an Out-of-Service State

Issue the _DSL_SYSTEM_ENABLE (0x01) command; the parameter should be 0x00.

14.3 Advanced API Sequencing Application Examples

14.3.1 G.shdsl EVM to Lab Module Transport

Table 14-4 G.shdsl EVM to Lab Module API Sequencing Example

Description	Opcode (HEX)	Parameter(s) [HEX]
System Enable	01	01 (HTU-C) 09 (HTU-R)
System Configuration	06	63 (G.shdsl Multirate mode)

Multirate Setting (set data rates)	1B	NPCM, 00, NDSL, NOCC, 01, i,, 00, 00
		Note: NPCM = Number of PCM Timeslots
		NDSL = Number of DSL Timeslots
		NOCC = Number of occupied PCM Timeslots

Training Mode (Default)	03	12 00 (G.shdsl Training / Coded 16-PAM / Symmetric PSD)
PCM Clock Source (Default)	12	0A (TPCLK = DPLL; RPCLK = DPLL)
PCM Configuration	10	03 (HTU-C; DPLL Open; Asynchronous) 01 (HTU-R; DPLL Closed; Asynchronous)
Multi-frame Length (Default)	05	2F 2F (6mS)
Pre-Activation Configuration	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 00 (MS Sender = HTU-C) 00 (TPS_TC = Do not modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00 (Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)

Activate (go)	0B	02 (fixed startup time-out, enable Act Request)

14.3.2 G.shdsl ATM Byte Alignment Application–Using UTOPIA

Table 14-5 lists the API sequencing for an ATM application. The following assumptions are made.

- ZipWirePlus device uses UTOPIA 16-bit Level 2.
- Framer must run in Synchronous mode.
- ATM must be ATM byte-aligned in the transmit direction.
- ZipWirePlus device provides master PCM clock–where both TPCLK and RPCLK are sourced from DPLL.
- HTU-C DPLL runs in open loop mode.
- HTU-R DPLL runs in closed loop mode.
- Multi-frame Length API must not be programmed. The internal driver will set it properly.
- PCM data rate will track DSL track (excluding DSL overhead).
- ZipWirePlus device operating as HTU-R is configured to send G.hs mode select.
- i bit in multirate must be set to zero for byte alignment
- This configuration is valid for M28945, M28946 and M28947 devices only.

Table 14-5 G.shdsl ATM Byte Alignment API Sequencing Example

Description	Opcode (HEX)	Parameter(s) [HEX]
System Enable	01	01 (HTU-C) 09 (HTU-R)
System Configuration	06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)	1B	NPCM, 00, NDSL, NOCC, 01, i,, 00, 00 Note: NPCM = Number of PCM Timeslots NDSL = Number of DSL Timeslots NOCC = Number of occupied PCM Timeslots
Clock Configuration	04	20 (Synchronous Mode) 01 00
ATM PHY Interface Mode (Gen Mux)	1E	1F 01 (ATM UTOPIA, General Purpose Mode, byte aligned)
ATM PHY UTOPIA Configuration	1D	11 (16-bit; Level 2) [default]
ATM PHY Configuration	20	00 (UTOPIA address 0) 0D (Enable scr/descr, HEC coset, and Auto HEC)
Training Mode	03	12 00 (G.shdsl Training / Coded 16-PAM / Symmetric PSD)
PCM Clock Source	12	0A (TPCLK = DPLL; RPCLK = DPLL)
PCM Configuration	10	02 (HTU-C; TPMFSYNC input: DPLL Open; Synchronous) 00 (HTU-R; TPMFSYNC input: DPLL Closed; Synchronous)
Multi-frame Length	05	Do not issue

Description	Opcode (HEX)	Parameter(s) [HEX]
Pre-Activation Configuration	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don’t care) 00 (MS Sender–HTU-C) 10 (TPS_TC = ATM) 00 (Data Rate Source =List) 01 (Annex A) 00 (Range i-Bit: don’t care) 00 (Range Min-N: don’t care) 00 (Range Max-N: don’t care) 00 (reserved)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)

14.3.3 G.shdsl ATM Non-Byte Alignment Application–Using UTOPIA

Table 14-6 provides an example for G.shdsl mode operating in an ATM application. In this application, the PCM and DSL typically operate at the same data rate (minus DSL overhead). On the HTU-C, the DPLL operates in open loop. On the HTU-R, the DPLL operates in closed loop mode. For both the HTU-C and HTU-R, the transmit and receive PCM clocks are sourced from the DPLL and operate in PCM Float (asynchronous alignment mode). This configuration is valid for M28945, M28946 and M28947 devices only.

Table 14-6 G.shdsl ATM Non-Byte Alignment–API Sequencing Example

Description	Opcode (HEX)	Parameter(s) [HEX]
System Enable	01	01 (HTU-C) 09 (HTU-R)
System Configuration	06	63 (G.shdsl, Multirate mode)

Multirate Setting (set data rates)	1B	NPCM, 00, NDSL, NOCC, 01, i,, 00, 00
		Note: NPCM = Number of PCM Timeslots
		NDSL = Number of DSL Timeslots
		NOCC = Number of occupied PCM Timeslots

ATM PHY Interface Mode (Gen Mux)	1E	1F (ATM UTOPIA, General Purpose mode) 00 (Byte align disabled)
ATM PHY UTOPIA Configuration	1D	11 (16-bit; Level 2) [default]
Training Mode	03	12 00 (G.shdsl Training / Coded 16-PAM / Symmetric PSD) [default]
PCM Clock Source	12	0A (TPCLK = DPLL; RPCLK = DPLL) [default]
PCM Configuration	10	03 (HTU-C; DPLL Open; Asynchronous) 01 (HTU-R; DPLL Closed; Asynchronous)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)

14.3.4 G.shdsl ATM–Using ATM Serial Interface

NOTE: This configuration is valid for M28945, M28946 and M28947 devices only.

Table 14-7 ATM Serial Interface Mode API Sequencing Example

Description	Opcode (HEX)	Parameter(s) [HEX]
System Enable	01	01 (HTU-C) 09 (HTU-R)
System Configuration	06	63 (G.shdsl Multirate mode)

Multirate Setting (set data rates)	1B	NPCM, 00, NDSL, NOCC, 01, i,, 00, 00
		Note: NPCM = Number of PCM Timeslots
		NDSL = Number of DSL Timeslots
		NOCC = Number of occupied PCM Timeslots

ATM PHY Interface Mode (Gen Mux)	1E	3F (ATM_SIF, General Purpose mode) 00 (Byte align disabled)
ATM PHY UTOPIA Configuration	0x1D	00 (8-bit; Level 1)
Training Mode	03	12 00(G.shdsl Training, Coded 16-PAM, Symmetric PSD)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)

14.3.5 2B1Q Mode With Auto Baud Pre-activation and ATM Enabled

- This configuration should work against the RS8973 EVM.
- Framer must be in framer transparent Mode.
- ATM can optionally be enabled (running in General Purpose mode)
- ATM does not need to be byte aligned.

Table 14-8 2B1Q Mode With Auto Baud Pre-activation Sequencing Example

Description	Opcode (HEX)	Parameter(s) [HEX]
System Enable	01	01 (HTU-C) 09 (HTU-R)
System Configuration	06	65 (Multirate, framer transparent mode)

Multirate Setting (set data rates)	1B	NPCM, 00, NDSL, NOCC, 01, i,, 00, 00
		Note: NPCM = Number of PCM Timeslots
		NDSL = Number of DSL Timeslots
		NOCC = Number of occupied PCM Timeslots

Description	Opcode (HEX)	Parameter(s) [HEX]
Pre-activation Data Rate List	15	07 (Num entries–0 based) 02 (N), 02 (i) = 144 kbps 04 (N), 02 (i) = 272 kbps 06 (N), 02 (i) = 400 kbps 08 (N), 02 (i) = 528 kbps 0C (N), 02 (i) = 784 kbps 12 (N), 02 (i) = 1168 kbps 18 (N), 02 (i) = 1552 kbps 24 (N), 02 (i) = 2320 kbps These are the data rates supported by the CX8973.
Pre-activation User Info	14	0x01 (ATM), 0x01 (SSS Scrambling, Hec Coset Disabled)
ATM PHY Configuration	20	00 (address 0) 09 (Disable HEC coset)
ATM PHY Interface Mode (Gen Mux)	1E	1F (ATM UTOPIA, General Purpose mode) 00 (ATM Byte alignment disabled)
PCM Clock Source	12	0A (TPCLK = DPLL; RPCLK = DPLL)
PCM Configuration	10	01 (DPLL Closed; Asynchronous)
Training Mode	03	60 00(2B1Q Training / Uncoded 4-PAM)
Pre-Activation Configuration	0F	01 (Mode = Auto Baud) 01 (Line Probe = enabled) 00 (N/A to Auto Baud) 00 (N/A to Auto Baud) 00 (N/A to Auto Baud) 10 (TPS_TC = ATM) [00 (List) for HTU-C or 02 (All) for HTU-R] 00 (N/A to Auto Baud) 00 (N/A to Auto Baud) 00 (N/A to Auto Baud) 00 (N/A to Auto Baud) 00 (N/A to Auto Baud)
Activate (go)	0B	22 (Variable Act Time, enable Act Request)

14.3.5.1 RS8973 EVM Configuration

When Auto Baud Pre-activation is disabled, Dip Switch # 7 and 6 set the data rate

00 = 1168kbps
01 = 2320kbps
10 = 784kbps
11 = 1552kbps

The rest of the dip switches should be in the down position.

NOTE: See to the 8973 EVM documentation for complete details.

14.3.6 Framed PCM Plus Unframed Narrowband Example

In this example, the ZipWirePlus device has the following configuration:

- NB Clock = 1024 khz (Nnb = NOCCnb = 16 timeslots)
- The Narrowband Data is unframed it has no Frame Sync.
- DSL Data Rate = 2056 kbps (Ndsl = 32 timeslots + 8kbps overhead)
- PCM Clock = 2048 khz (Npcm = 32 timeslots)
- Occupied PCM = 1024 kbps (Nocc = 16 timeslots)
- PCM Sync = 6 ms
- The PCM Application is the master of the clock and it provides the clock to the ZipWirePlus device on HTU-C and HTU-R
- The NB Application clock **should have** the same reference as the PCM Application.
- On both HTU-C & HTU-R, PCM operates in aligned mode and TPMFSYNC is an input.
- The PCM is mapped before the Narrowband within the DSL frame.

NOTE: This configuration is valid for M28945, M28946 and M28947 devices only.

Table 14-9 PCM + Narrowband API Sequencing Example

Description	Opcode (HEX)	Parameter(s) [HEX]
System Enable	01	01 (HTU-C) 09 (HTU-R)
System Configuration	06	63 (G.shdsl Multirate mode)

PCM Multirate Setting	1B	NPCM, 00, NDSL, NOCC, 01, 00, 00, 00
		Note: NPCM = Number of PCM Timeslots
		NDSL = Number of DSL Timeslots
		NOCC = Number of occupied PCM Timeslots

PCM Configuration	10	80 (PCM+NB, TPMFSYNC = Input, DPLL Closed; Aligned)
Narrowband Multirate Setting	1A	Nnb 00 NOCCnb 01 00 00 (PCM before NB) Note: Nnb = Number of narrowband TS NOCCnb = Number of occupied narrowband TS
Pre-Activation Configuration	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 00 (MS Sender–HTU-C) 00 (TPS_TC = Do Not Modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00 (Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)
PCM Clock Source	12	08 (HTU-C: TPCLK = pin; RPCLK = DPLL)

Description	Opcode (HEX)	Parameter(s) [HEX]
Narrowband Configuration	28	80 (Byte #1: TNBMFSYNC is derived from TPMFSYNC internally, TNBMFSYNC = Input, DPLL Closed; Aligned) 08 (Byte #2: TNBCLK = pin; RNBCLK = DPLL) 2F (Byte #3: 6mS TNBMFSYNC) 2F (Byte #4: 6mS RNBMFSYNC)
Multi-frame Length	05	2F 2F (6mS)
PCM Water Level	2C	00 F0 00 F0 00 (Please note that the Water Level is unique for each combination of PCM and NB Rate)
NB Water Level	2D	00 20 00 7F 00 (Please note that the Water Level is unique for each combination of PCM and NB Rate)
G.hs API Commands	XX	As desired
Training Mode	03	12 00(G.shdsl Training / Coded 16-PAM / Symmetric PSD)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)

14.3.7 Common Sync Application Setup

Common sync is a setting where the transmit and receive data use the same sync signal. The signal should be the transmit sync pulse (TPMFSYNC). This pulse can either be configured as an input or output (see DSL_FR_PCM_CONFIG 0x10 API), and the receive sync pulse (RPMFSYNC) should be ignored. When in common sync mode it is important to note that the alignment of the receive frame to the transmit pulse can be done by using the ZipWirePlus internal slip buffer. This slip buffer only allows for a 1 frame buffer, therefore when in common sync, the TPMFSYNC and RPMFSYNC both need to be 1 frame(value 0) or 125us (_DSL_PCM_MF_LEN 0x05 API). Once in common sync mode, the receive data will have to be shifted to align to the TPMFSYNC pulse. The amount of the shift depends on if the TPMFSYNC is an input or an output.

When the TPMFSYNC is an:

OUTPUT, the RP_FRM_OFST (_DSL_RP_FRM_OFST 0x2F API) should be the (number of PCM timeslots times 8) minus 3. ((Nx8)-3) An example 32 timeslots [32x8 = 256 – 3 = 253(0xFD)]

INPUT, the RP_FRM_OFST (_DSL_RP_FRM_OFST 0x2F API) should be the (number of PCM timeslots times 8) minus 6. ((Nx8)-6) An example 32 timeslots [32x8 = 256 – 6 = 250(0xFA)]

In this mode since a single clock is used, both the TPCLK and RPCLK (_DSL_PCM_CLK_CONF 0x12 API) have to be from the same source. For example, in the case where the ZipWirePlus is the master of the transmit/receive clock source, both the TPCLK and RPCLK have to sourced from the PCM_DPLL. In the case where the ZipWirePlus is the slave of the transmit/receive clock source, both TPCLK and RPCLK have to be sourced from TPCLK input pin.

In common sync mode the TPMFSYNC (both in Input and Output modes) signifies bit ZERO of the PCM frame.

14.3.8 Custom Mapping Example

In this example the:

- ♦ PCM is configured for N*64 data rates
- ♦ The used time slots are(3,8,18,30 and 31).
- ♦ The same time slots are used in both the TX and RX directions.
- ♦ The PCM bus is tri-stated when there is no data to transmit in the receive direction.
- ♦ In the TX direction, the unused time slot data is ignored. In the RX direction, the unused time slots are mapped from DBANK_1.

Table 14-10 Custom Mapping API Sequence Example

Description	Opcode (HEX)	Parameter(s) [HEX]		
System Enable	01	01 (HTU-C)		
System Config	06	0x63 (G.shdsl Multirate mode)		
DSL Framer PCM Configuration	10	0x01 (PCM Only, DPLL Mode = Closed, PCM Float= Asynchronous).		
Multipair Configuration	0x19	0x01, Master, Bused mode. (this is required only if RPDAT needs to be tri-stated on unused timeslots in single pair application).		
Mutli-Rate Setting (set data rates)	1B	0x20 0x00 0x20 0x20 0x01 0x00 0x00 0x01		
Training Mode	03	12 00(G.shdsl Training / Coded 16-PAM / Symmetric PSD)		
PCM Clock Source	12	08 (TPCLK = pin ; RPCLK = DPLL)		
Multi-frame Length	05	0x2F 0x2F		
Activate (go)	0B	02		
Wait for Link Activation				
Set DSL Framer State Machine	4A	00 (Disable Framer Interrupts)		
Transmit PCM Mapper Value	30	**Byte #**	**Value**	**Comments**
		1	0x00	Address
		2	0x40	TS 0-2, Disregard Data
		3	0x01	TS 3, Data from TPDAT
		4	0x60	TS 4-7, Disregard Data
		5	0x01	TS 8, Data from TPDAT
		6	0xE0	TS 9-16, Disregard Data
		7	0x00	TS 17, Disregard Data
		8	0x01	TS 18, Data from TPDAT
		9	0xE0	TS 19-26, Disregard Data
		10	0x40	TS 27-29, Disregard Data
		11	0x21	TS 30-31, Data from TPDAT
Transmit PCM Mapper Write	31	00		
Receive PCM Mapper Value	32	**Byte #**	**Value**	**Comments**
		1	0x00	Address
		2	0x42	TS 0-2, Data from DBANK_1
		3	0x00	TS 3, Data from RX FIFO
		4	0x62	TS 4-7, Data from DBANK_1
		5	0x00	TS 8, Data from RX FIFO

Description	Opcode (HEX)	Parameter(s) [HEX]		
		6	0xE2	TS 9-16, Data from DBANK_1
		7	0x02	TS 17, Data from DBANK_1
		8	0x00	TS 18, Data from RX FIFO
		9	0xE2	TS 19-26, Data from DBANK_1
		10	0x42	TS 27-29, Data from DBANK_1
		11	0x20	TS 30-31, Data from RX FIFO
Receive PCM Mapper Write	33	00		
Set DSL Framer State Machine	4A	01 (Enable Framer Interrupts)		

14.3.9 Internal E1 Framer Example

In this example the:

- PCM is configured to E1 data rate 2048 Kbps.
- The HTU-C and HTU-R PCM interface are configured as identical, so the path from HTU-C to HTU-R and HTU-R to HTU-C are independent.
- The PCM transmit interface need to be provided with external 2ms E1 multiframe sync.
- It is recommended that CRC-4 mode is set to automatic mode (recalculate CRC-4).

Table 14-11 E1 Framer Sequence Example

Description	Opcode (HEX)	Parameter(s) [HEX]
System Enable	01	01 (HTU-C) 09 (HTU-R)
System Configuration	06	63 (G.shdsl Multirate mode)
PCM Multirate Setting	1B	20 00 20 20 01 00 00 01
PCM Configuration	10	00 (PCM Only, TPMFSYNC = Input, DPLL Closed; Aligned)
PCM Clock Source	12	08 (TPCLK = pin; RPCLK = DPLL)
Multi-frame Length	05	0F 0F (2mS)
G.hs API Commands	XX	As desired
Training Mode	03	12 00(G.shdsl Training / Coded 16-PAM / Symmetric PSD)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)
Wait for link is up and framer are stabilized:		

Description	Opcode (HEX)	Parameter(s) [HEX]
E1 Framer Configurations	44	07 (Enable E1 Framer; Select external sync) 05 (TX Path Configure: A bit: transparent mode E bit: automatic mode CRC-4: automatic mode) 00(TX Path Configure: SA4, SA5, SA6, SA7, SA8: transparent mode) 05 (RX Path Configure: A bit: transparent mode E bit: automatic mode CRC-4: automatic mode) 00(RX Path Configure: SA4, SA5, SA6, SA7, SA8: transparent mode)
Poll E1 Framer Error Counters periodically	C4	00 (look for transmit and receive direction’s CRC-4 errors and E bit errors)

14.3.10 2E1 Multi-Pair Cascade Mode Example

2E1 Multi-Pair Cascade mode example description:

- ♦ 2 pair point to point E1
- ♦ Cascade Mode
- ♦ HTU-R and HTU-C Sides include one master and one slave with any of the M28945/46/47 devices.
- ♦ MS8370 E1 framer connected through PCM port on HTU-R and HTU-C.When the dsl link comes up, the software sets pcm/dsl mapping as a default block or interleave mapping, which does not comply with 2E1 mapping standard. Custom mapping must be performed after the link is up.

Table 14-12 2E1 Multi-Pair Cascade Mode Sequence Example

Description	Opcode (HEX)	Parameter(s) [HEX]
System Enable	01	01 (HTU-C) 09 (HTU-R)
System Config	06	0x63 (G.shdsl, Multirate mode)
DSL Framer PCM Configuration	10	0x00 (PCM Only, TPMFSYNC input, DPLL Mode = Closed, PCM Aligned).
Multipair Configuration	19	0x02 (Master, PCM Cascade) 0x12 (Slave, PCM Cascade)
DSL PCM Clock Configure	12	0x08 (Master : RPCLK = DPLL, TPCLK = input pin) 0x04 (Slave: RPCLK = PEXTCLK, TPCLK = input pin)
Mutli-Rate Setting (set data rates)	1B	0x20 0x00 0x12 0x12 0x01 0x00 0x00 0x01 (blocking mapping)
Training Mode	03	12 00(G.shdsl Training / Coded 16-PAM / Symmetric PSD)
DSL Preactivation Configure	0F	0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00 (Do not configure PCM, Annex B)
Multi-frame Length	05	0x2F 0x2F (6ms for MS 8970 framer)

Description	Opcode (HEX)	Parameter(s) [HEX]		
Activate (go)	0B	02		
Wait for Link Activation				
MASTER: TS 0,1,3,5,7,9,11,13,15,16,18,20,22,24,26,28,30,DBANK1				
Set DSL Framer State Machine	4A	0x00 (Disable Framer Interrupts)		
Transmit PCM Mapper Value (Master)	30	**Byte #**	**Value**	**Comments**
		1	0x00	Starting Address
		2	0x11	TS 0-1, Data from TPDAT
		3	0x00	TS 2, Disregard Data
		4	0x01	TS 3, Data from TPDAT
		5	0x00	TS 4, Disregard Data
		6	0x01	TS 5, Data from TPDAT
		7	0x00	TS 6, Disregard Data
		8	0x01	TS 7, Data from TPDAT
		9	0x00	TS 8, Disregard Data
		10	0x01	TS 9, Data from TPDAT
		11	0x00	TS 10, Disregard Data
		12	0x01	TS 11, Data from TPDAT
		13	0x00	TS 12, Disregard Data
		14	0x01	TS 13, Data from TPDAT
		15	0x00	TS 14, Disregard Data
		16	0x11	TS 15-16, Data from TPDAT
		17	0x00	TS 17, Disregard Data
		18	0x01	TS 18, Data from TPDAT
		19	0x00	TS 19, Disregard Data
		20	0x01	TS 20, Data from TPDAT
		21	0x00	TS 21, Disregard Data
		22	0x01	TS 22, Data from TPDAT
		23	0x00	TS 23, Disregard Data
		24	0x01	TS 24, Data from TPDAT
		25	0x00	TS 25, Disregard Data
		26	0x01	TS 26, Data from TPDAT

Description	Opcode (HEX)	Parameter(s) [HEX]		
		27	0x00	TS 27, Disregard Data
		28	0x01	TS 28, Data from TPDAT
		29	0x00	TS 29, Disregard Data
		30	0x01	TS 30, Data from TPDAT
		31	0x00	TS 31, Disregard Data
Transmit PCM Mapper Write (Master)	31	00		
Receive PCM Mapper Value (Master)	32	Byte #	Value	Comments
		1	0x00	Starting Address
		2	0x10	TS 0-1, Data from RX PCM FIFO
		3	0x05	TS 2, Data From RPEXDAT (Slave Loop)
		4	0x00	TS 3, Data from RX PCM FIFO
		5	0x05	TS 4, Data From RPEXDAT (Slave Loop)
		6	0x00	TS 5, Data from RX PCM FIFO
		7	0x05	TS 6, Data From RPEXDAT (Slave Loop)
		8	0x00	TS 7, Data from RX PCM FIFO
		9	0x05	TS 8, Data From RPEXDAT (Slave Loop)
		10	0x00	TS 9, Data from RX PCM FIFO
		11	0x05	TS 10, Data From RPEXDAT (Slave Loop)
		12	0x00	TS 11, Data from RX PCM FIFO
		13	0x05	TS 12, Data From RPEXDAT (Slave Loop)
		14	0x00	TS 13, Data from RX PCM FIFO
		15	0x05	TS 14, Data From RPEXDAT (Slave Loop)
		16	0x20	TS 15-16, Data from RX PCM FIFO
		17	0x05	TS 17, Data From RPEXDAT (Slave Loop)
		18	0x00	TS 18, Data from RX PCM FIFO
		19	0x05	TS 19, Data From RPEXDAT (Slave Loop)
		20	0x00	TS 20, Data from RX PCM FIFO
		21	0x05	TS 21, Data From RPEXDAT (Slave Loop)
		22	0x00	TS 22, Data from RX PCM FIFO
		23	0x05	TS 23, Data From RPEXDAT (Slave Loop)
		24	0x00	TS 24, Data from RX PCM FIFO

Description	Opcode (HEX)	Parameter(s) [HEX]		
		25	0x05	TS 25, Data From RPEXDAT (Slave Loop)
		26	0x00	TS 26, Data from RX PCM FIFO
		27	0x05	TS 27, Data From RPEXDAT (Slave Loop)
		28	0x00	TS 28, Data from RX PCM FIFO
		29	0x05	TS 29, Data From RPEXDAT (Slave Loop)
		30	0x00	TS 30, Data from RX PCM FIFO
		31	0x05	TS 31, Data From RPEXDAT (Slave Loop)
Receive PCM Mapper Write (Master)	33	00		
Slave–TS 0,2,4,6,8,10,12,14,16,17,19,21, 23, 25, 27, 29, 31, DBANK. DBANK will be inserted from TH_MAPPER				
Transmit PCM Mapper Value (Slave) Map 17 TS to TFIFO	30	Byte #	Value	Comments
		1	0x00	Starting Address
		2	0x01	TS 0, Data from TPDAT
		3	0x00	TS 1, Disregard Data
		4	0x01	TS 2, Data from TPDAT
		5	0x00	TS 3, Disregard Data
		6	0x01	TS 4, Data from TPDAT
		7	0x00	TS 5, Disregard Data
		8	0x01	TS 6, Data from TPDAT
		9	0x00	TS 7, Disregard Data
		10	0x01	TS 8, Data from TPDAT
		11	0x00	TS 9, Disregard Data
		12	0x01	TS 10, Data from TPDAT
		13	0x00	TS 11, Disregard Data
		14	0x01	TS 12, Data from TPDAT
		15	0x00	TS 13, Disregard Data
		16	0x01	TS 14, Data from TPDAT
		17	0x00	TS 15, Disregard Data
		18	0x11	TS 16-17, Data from TPDAT
		19	0x00	TS 18, Disregard Data
		20	0x01	TS 19, Data from TPDAT
		21	0x00	TS 20, Disregard Data

Description	Opcode (HEX)	Parameter(s) [HEX]		
		22	0x01	TS 21, Data from TPDAT
		23	0x00	TS 22, Disregard Data
		24	0x01	TS 23, Data from TPDAT
		25	0x00	TS 24, Disregard Data
		26	0x01	TS 25, Data from TPDAT
		27	0x00	TS 26, Disregard Data
		28	0x01	TS 27, Data from TPDAT
		29	0x00	TS 28, Disregard Data
		30	0x01	TS 29, Data from TPDAT
		31	0x00	TS 30, Disregard Data
		32	0x01	TS 31, Data from TPDAT
Transmit PCM Mapper Write (Slave)	31	00		
Receive PCM Mapper Value (Slave)	32	Byte #	Value	Comments
		1	0x00	Starting Address
		2	0x00	TS 0, Data from RX PCM FIFO
		3	0x02	TS 1, Data From DBANK_1
		4	0x00	TS 2, Data from RX PCM FIFO
		5	0x02	TS 3, Data From DBANK_1
		6	0x00	TS 4, Data from RX PCM FIFO
		7	0x02	TS 5, Data From DBANK_1
		8	0x00	TS 6, Data from RX PCM FIFO
		9	0x02	TS 7, Data From DBANK_1
		10	0x00	TS 8, Data from RX PCM FIFO
		11	0x02	TS 9, Data From DBANK_1
		12	0x00	TS 10, Data from RX PCM FIFO
		13	0x02	TS 11, Data From DBANK_1
		14	0x00	TS 12, Data from RX PCM FIFO
		15	0x02	TS 13, Data From DBANK_1
		16	0x00	TS 14, Data from RX PCM FIFO
		17	0x02	TS 15, Data From DBANK_1
		18	0x10	TS 16-17, Data from RX PCM FIFO

Description	Opcode (HEX)	Parameter(s) [HEX]		
		19	0x02	TS 18, Data From DBANK_1
		20	0x00	TS 19, Data from RX PCM FIFO
		21	0x02	TS 20, Data From DBANK_1
		22	0x00	TS 21, Data from RX PCM FIFO
		23	0x02	TS 22, Data From DBANK_1
		24	0x00	TS 23, Data from RX PCM FIFO
		25	0x02	TS 24, Data From DBANK_1
		26	0x00	TS 25, Data from RX PCM FIFO
		27	0x02	TS 26, Data From DBANK_1
		28	0x00	TS 27, Data from RX PCM FIFO
		29	0x02	TS 28, Data From DBANK_1
		30	0x00	TS 29, Data from RX PCM FIFO
		31	0x02	TS 30, Data From DBANK_1
		32	0x00	TS 31, Data from RX PCM FIFO
Receive PCM Mapper Write (Salve)	33	00		
Transmit DSL Mapper Value (Rewrite the 18th TS insert from DBANK1)	34	Byte #	Value	Comments
		1	0x08	Starting Address 8 (0-based), rewrite the 9th entry
		2	0x01	TS 17, Read data from PCM Transmit FIFO; TS 18, insert from DBANK1
Transmit DSL Mapper Write	35	00		
Receive DSL Mapper Value (Master/slave) (Discard DBANK1)	36	Byte #	Value	Comments
		1	0x08	Starting Address 8 (0-based), rewrite the 9th entry
		2	0x01	TS 17, write data to RFIFO; TS 18, discard DBANK1
Receive PCM Mapper Write (Master/slave)	37	00		
Set DSL Framer State Machine	4A	01 (Enable Framer Interrupts)		
Wait for the framer stabilized after rerun state machine				
Reset Master's Transmit Water level	4E	08		
Reset Master's Transmit FIFO	4E	10		
Reset Master's Receive Water level	4F	08		
Reset Master's Receive FIFO	4F	10		

Description	Opcode (HEX)	Parameter(s) [HEX]
Reset Slave's Transmit Water level	4E	08
Reset Slave's Transmit FIFO	4E	10
Reset Slave's Receive Water level	4F	08
Reset Slave's Receive FIFO	4F	10

14.3.11 Configuring for G.shdsl Unframed PCM 4-wire Application

Application is Master of Clock

In this example:

- ♦ The Application provides the clock to the ZipWireplus HTU-C and ZipWirePlus HTU-R
- ♦ Application transports 6 timeslots (384 Kbps) with 3 timeslots being transported on each Pair.
- ♦ The TPMFSYNC pin on the Master ZipWirePlus device is connected to the TPMFSYNC pin of the Slave ZipWirePlus device.

Table 14-13 Unframed PCM 4-wire mode with the Application being the master of the Clock

Description	Opcode (HEX)	Parameter(s) [HEX]
HTU-C Master		
System Enable	01	01 (HTU-C)
System Configuration	06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)	1B	06, 00, 03, 03, 01, 00, 80, 01 (PCM = 6, DSL = 3, Occupied PCM = 3, Start PCM = 1, Block Mapping)
Pre-Activation Configuration	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 00 (MS Sender–HTU-C) 00 (TPS_TC = Do Not Modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00 (Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)
Training Mode	03	12 00 (G.shdsl Training / Coded 16-PAM / Symmetric PSD)
PCM Clock Source	12	08 (TPCLK = TPCLK pin; RPCLK = DPLL)
PCM Configuration	10	21 (TPMFSYNC output: DPLL Closed; Asynchronous)
Multi-frame Length	05	2F 2F (6 ms)

Description	Opcode (HEX)	Parameter(s) [HEX]
PCM Water Level	2C	00 30 00 30 00 (Please note that the Water Level is unique for each PCM Rate)
Multi-Pair Configuration	19	02 (Master, PCM Cascade Mode)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)
HTU-C Slave		
System Enable	01	01 (HTU-C)
System Configuration	06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)	1B	06, 00, 03, 04, 01, 00, 80, 01 (PCM = 6, DSL = 3, Occupied PCM = 3, Start PCM = 4, Block Mapping)
Pre-Activation Configuration	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 00 (MS Sender–HTU-C) 00 (TPS_TC = Do Not Modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00 (Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)
Training Mode	03	12 00 (G.shdsl Training / Coded 16-PAM / Symmetric PSD)
PCM Clock Source	12	04 (TPCLK = TPCLK pin; RPCLK = PEXTCLK pin)
PCM Configuration	10	0A (TPMFSYNC input: DPLL Open Loop; Sync Aligned, PCM DPLL Ref = DSLSYNCI)
Multi-frame Length	05	2F 2F (6 ms)
PCM Water Level	2C	00 30 00 30 00 (Please note that the Water Level is unique for each PCM Rate)
Multi-Pair Configuration	19	12 (Slave, PCM Cascade Mode)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)
HTU-R Master		
System Enable	01	09 (HTU-R)
System Configuration	06	63 (G.shdsl Multirate mode)

Description	Opcode (HEX)	Parameter(s) [HEX]
Multirate Setting (set data rates)	1B	06, 00, 03, 03, 01, 00, 80, 01 (PCM = 6, DSL = 3, Occupied PCM = 3, Start PCM = 1, Block Mapping)
Pre-Activation Configuration	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 00 (MS Sender–HTU-C) 00 (TPS_TC = Do Not Modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00 (Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)
Training Mode	03	12 00 (G.shdsl Training / Coded 16-PAM / Symmetric PSD)
PCM Clock Source	12	08 (TPCLK = TPCLK pin; RPCLK = DPLL)
PCM Configuration	10	21 (TPMFSYNC output: DPLL Closed; Asynchronous)
Multi-frame Length	05	2F 2F (6 ms)
PCM Water Level	2C	00 30 00 30 00 (Please note that the Water Level is unique for each PCM Rate)
Multi-Pair Configuration	19	02 (Master, PCM Cascade Mode)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)
HTU-R Slave		
System Enable	01	09 (HTU-R)
System Configuration	06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)	1B	06, 00, 03, 04, 01, 00, 80, 01 (PCM = 6, DSL = 3, Occupied PCM = 3, Start PCM = 4, Block Mapping)

Description	Opcode (HEX)	Parameter(s) [HEX]
Pre-Activation Configuration	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 00 (MS Sender–HTU-C) 00 (TPS_TC = Do Not Modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00 (Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)
Training Mode	03	12 00 (G.shdsl Training / Coded 16-PAM / Symmetric PSD)
PCM Clock Source	12	04 (TPCLK = TPCLK pin; RPCLK = PEXTCLK pin)
PCM Configuration	10	0A (TPMFSYNC input: DPLL Open Loop; Sync Aligned, PCM DPLL Ref = DSLSYNCI)
Multi-frame Length	05	2F 2F (6 ms)
PCM Water Level	2C	00 30 00 30 00 (Please note that the Water Level is unique for each PCM Rate)
Multi-Pair Configuration	19	12 (Slave, PCM Cascade Mode)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)

ZipWirePlus device is Master of Clock

In this example:

- The Application is slave to the clock from the ZipWireplus HTU-C and ZipWirePlus HTU-R
- Application transports 6 timeslots (384 Kbps) with 3 timeslots being transported on each Pair.
- The TPMFSYNC pin on the Master ZipWirePlus device is connected to the TPMFSYNC pin of the Slave ZipWirePlus device.

Table 14-14 Unframed PCM 4-wire mode with the Application being the Slave to the Clock

Description	Opcode (HEX)	Parameter(s) [HEX]
HTU-C Master		
System Enable	01	01 (HTU-C)
System Configuration	06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)	1B	06, 00, 03, 03, 01, 00, 80, 01 (PCM = 6, DSL = 3, Occupied PCM = 3, Start PCM = 1, Block Mapping)

Description	Opcode (HEX)	Parameter(s) [HEX]
Pre-Activation Configuration	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 00 (MS Sender–HTU-C) 00 (TPS_TC = Do Not Modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00 (Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)
Training Mode	03	12 00 (G.shdsl Training / Coded 16-PAM / Symmetric PSD)
PCM Clock Source	12	0A (TPCLK = DPLL; RPCLK = DPLL)
PCM Configuration	10	23 (TPMFSYNC output: DPLL Open; Asynchronous)
Multi-frame Length	05	2F 2F (6 ms)
PCM Water Level	2C	00 30 00 30 00 (Please note that the Water Level is unique for each PCM Rate)
Multi-Pair Configuration	19	02 (Master, PCM Cascade Mode)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)
HTU-C Slave		
System Enable	01	01 (HTU-C)
System Configuration	06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)	1B	06, 00, 03, 04, 01, 00, 80, 01 (PCM = 6, DSL = 3, Occupied PCM = 3, Start PCM = 4, Block Mapping)
Pre-Activation Configuration	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 00 (MS Sender–HTU-C) 00 (TPS_TC = Do Not Modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00 (Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)

Description	Opcode (HEX)	Parameter(s) [HEX]
Training Mode	03	12 00 (G.shdsl Training / Coded 16-PAM / Symmetric PSD)
PCM Clock Source	12	04 (TPCLK = TPCLK pin; RPCLK = PEXTCLK pin)
PCM Configuration	10	0A (TPMFSYNC input: DPLL Open Loop; Sync Aligned, PCM DPLL Ref = DSLSYNCI)
Multi-frame Length	05	2F 2F (6 ms)
PCM Water Level	2C	00 30 00 30 00 (Please note that the Water Level is unique for each PCM Rate)
Multi-Pair Configuration	19	12 (Slave, PCM Cascade Mode)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)
HTU-R Master		
System Enable	01	09 (HTU-R)
System Configuration	06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)	1B	06, 00, 03, 03, 01, 00, 80, 01 (PCM = 6, DSL = 3, Occupied PCM = 3, Start PCM = 1, Block Mapping)
Pre-Activation Configuration	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don’t care) 00 (MS Sender–HTU-C) 00 (TPS_TC = Do Not Modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00 (Range i-Bit: don’t care) 00 (Range Min-N: don’t care) 00 (Range Max-N: don’t care) 00 (reserved)
Training Mode	03	12 00 (G.shdsl Training / Coded 16-PAM / Symmetric PSD)
PCM Clock Source	12	0A (TPCLK = DPLL pin; RPCLK = DPLL)
PCM Configuration	10	21 (TPMFSYNC output: DPLL Closed; Asynchronous)
Multi-frame Length	05	2F 2F (6 ms)
PCM Water Level	2C	00 30 00 30 00 (Please note that the Water Level is unique for each PCM Rate)
Multi-Pair Configuration	19	02 (Master, PCM Cascade Mode)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)
HTU-R Slave		

Description	Opcode (HEX)	Parameter(s) [HEX]
System Enable	01	09 (HTU-R)
System Configuration	06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)	1B	06, 00, 03, 04, 01, 00, 80, 01 (PCM = 6, DSL = 3, Occupied PCM = 3, Start PCM = 4, Block Mapping)
Pre-Activation Configuration	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 00 (MS Sender–HTU-C) 00 (TPS_TC = Do Not Modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00 (Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)
Training Mode	03	12 00 (G.shdsl Training / Coded 16-PAM / Symmetric PSD)
PCM Clock Source	12	04 (TPCLK = TPCLK pin; RPCLK = PEXTCLK pin)
PCM Configuration	10	0A (TPMFSYNC input: DPLL Open Loop; Sync Aligned, PCM DPLL Ref = DSLSYNCI)
Multi-frame Length	05	2F 2F (6 ms)
PCM Water Level	2C	00 30 00 30 00 (Please note that the Water Level is unique for each PCM Rate)
Multi-Pair Configuration	19	12 (Slave, PCM Cascade Mode)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)

14.3.12 Configuring for G.shdsl ATM 4-wire Application

The ZipWirePlus datasheet describes the ATM 4-wire Mode. The following is API Configuration Sequence for the G.shdsl ATM 4-wire mode. Please note that the Slave ZipWirePlus device ATM Interace is not used, it should be disabled.

Table 14-15 G.shdsl ATM 4-wire API Configuration Sequence

Description	Opcode (HEX)	Parameter(s) [HEX]
HTU-C Master		
System Enable	01	01 (HTU-C)
System Configuration	06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)	1B	20, 00, 10, 10, 01, 00, 80, 01 (PCM = 32, DSL = 16, Occupied PCM = 16, Start PCM = 1, Block Mapping)
Pre-Activation Configuration	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 00 (MS Sender–HTU-C) 00 (TPS_TC = Do Not Modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00 (Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)
Training Mode	03	12 00 (G.shdsl Training / Coded 16-PAM / Symmetric PSD)
PCM Clock Source	12	0A (TPCLK = DPLL; RPCLK = DPLL)
DSL Framer Configure	11	03 (AUX Enable)
Multi-frame Length	05	2F 2F (6 ms)
ATM PHY Interface Mode (Gen Mux)	1E	5F 01 (ATM UTOPIA, General Purpose Mode, byte aligned)
ATM PHY UTOPIA Configuration	1D	11 (16-bit; Level 2) [default]
ATM PHY Configuration	20	01 (UTOPIA address 1) 0D (Enable scr/descr, Enable HEC coset, and Auto HEC)
Multi-Pair Configuration	19	03 (Master, ATM Cascade Mode)
PCM Configuration	10	22 (TPMFSYNC output: DPLL Open; Sync Aligned; PCM DPLL Ref Select = Internal DSL Sync Detector) NOTE: this API must be executed before API 0x0B
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)
HTU-C Slave		
System Enable	01	01 (HTU-C)
System Configuration	06	63 (G.shdsl Multirate mode)

Description	Opcode (HEX)	Parameter(s) [HEX]
Multirate Setting (set data rates)	1B	20, 00, 10, 10, 11, 00, 80, 01 (PCM = 32, DSL = 16, Occupied PCM = 16, Start PCM = 17, Block Mapping)
Pre-Activation Configuration	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 00 (MS Sender–HTU-C) 00 (TPS_TC = Do Not Modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00 (Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)
Training Mode	03	12 00 (G.shdsl Training / Coded 16-PAM / Symmetric PSD)
PCM Clock Source	12	04 (TPCLK = TPCLK pin; RPCLK = PEXTCLK pin)
Multi-frame Length	05	2F 2F (6 ms)
ATM PHY Interface Mode (Gen Mux)	1E	00 01 (ATM DISABLE)
Multi-Pair Configuration	19	12 (Slave, PCM Cascade Mode)
PCM Configuration	10	0A (TPMFSYNC input: DPLL Open Loop; Sync Aligned, PCM DPLL Ref = DSLSYNCI input pin)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)
HTU-R Master		
System Enable	01	09 (HTU-R)
System Configuration	06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)	1B	20, 00, 10, 10, 01, 00, 80, 01 (PCM = 32, DSL = 16, Occupied PCM = 16, Start PCM = 1, Block Mapping)

Description	Opcode (HEX)	Parameter(s) [HEX]
Pre-Activation Configuration	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 00 (MS Sender–HTU-C) 00 (TPS_TC = Do Not Modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00 (Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)
Training Mode	03	12 00 (G.shdsl Training / Coded 16-PAM / Symmetric PSD)
PCM Clock Source	12	0A (TPCLK = DPLL; RPCLK = DPLL)
DSL Framer Configure	11	03 (AUX Enable)
Multi-frame Length	05	2F 2F (6 ms)
ATM PHY Interface Mode (Gen Mux)	1E	5F 01 (ATM UTOPIA, General Purpose Mode, byte aligned)
ATM PHY UTOPIA Configuration	1D	11 (16-bit; Level 2) [default]
ATM PHY Configuration	20	01 (UTOPIA address 1) 0D (Enable scr/descr, Enable HEC coset, and Auto HEC)
Multi-Pair Configuration	19	03 (Master, ATM Cascade Mode)
PCM Configuration	10	22 (TPMFSYNC output: DPLL opened; Sync Aligned;PCM DPLL Ref Select = Internal DSL Sync Detector) NOTE: this API must be executed before API 0x0B
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)
HTU-R Slave		
System Enable	01	09 (HTU-R)
System Configuration	06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)	1B	20, 00, 10, 10, 11, 00, 80, 01 (PCM = 32, DSL = 16, Occupied PCM = 16, Start PCM = 17, Block Mapping)

Description	Opcode (HEX)	Parameter(s) [HEX]
Pre-Activation Configuration	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 00 (MS Sender–HTU-C) 00 (TPS_TC = Do Not Modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00 (Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)
Training Mode	03	12 00 (G.shdsl Training / Coded 16-PAM / Symmetric PSD)
PCM Clock Source	12	04 (TPCLK = TPCLK pin; RPCLK = PEXTCLK pin)
Multi-frame Length	05	2F 2F (6 ms)
ATM PHY Interface Mode (Gen Mux)	1E	00 01 (ATM DISABLE)
Multi-Pair Configuration	19	12 (Slave, PCM Cascade Mode)
PCM Configuration	10	0A (TPMFSYNC input: DPLL Open Loop; Sync Aligned, PCM DPLL Ref = DSLSYNCI input pin)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)

14.3.13 Configuring for G.shdsl Proprietary Low Data Rates

The ZipWireplus device supports low data rates of 64 Kbps and 128 kbps using a proprietary 4 TCPAM coding. The list of APIs below setup the ZipWirePlus HTU-C and HTU-R for 64 Kbps Payload rate and a DSL rate of 72 kbps.

Table 14-16 G.shdsl Low Data Rate Configuration Sequence

Description	Opcode (HEX)	Parameter(s) [HEX]
System Enable	01	01 (HTU-C) 09 (HTU-R)
System Configuration	06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)	1B	01 00 01 01 01 00 00 00
Training Mode	03	32 00 (G.shdsl Training / Coded 4-PAM/ Symmetric PSD)
PCM Clock Source (Default)	12	0A (TPCLK = DPLL; RPCLK = DPLL)

Description	Opcode (HEX)	Parameter(s) [HEX]
PCM Configuration	10	03 (HTU-C; DPLL Open; Asynchronous) 01 (HTU-R; DPLL Closed; Asynchronous)
Multi-frame Length (Default)	05	2F 2F (6mS)
Pre-Activation Configuration (Default)	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 04 (MS Sender = No Remote Configure) 00 (TPS_TC = Do not modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00(Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)
Activate (go)	0B	02 (fixed startup time-out, enable Activation Request)

14.3.14 Configuring an IDSL NT Application

The M28945 and the M28950 devices support IDSL NT Mode. This API sequence configures the IDSL NT for user data rate of 128 kbps. In this configuration the ZipWirePlus device is the master of the clock. Please note that the ZipWireplus device will provide the clock to the Application through the RPCLK signal. The Application will transmit the data with respect to the RPCLK. The RPMFSYNC should be connected to the TPMFSYNC pin of the ZipWirePlus device.

Table 14-17 Configuring an IDSL NT Application

Description	Opcode (HEX)	Parameter(s) [HEX]
System Enable	01	09 (NT)
System Configuration	06	66 (IDSL mode)
PCM Multirate Setting	1B	02 00 02 02 01 00 00 01 (Payload Rate – 128 Kbps)
PCM Configuration	10	00 (PCM Only, DPLL Closed; Aligned)
PCM Clock Source	12	0A (TPCLK = DPLL; RPCLK = DPLL)
Training Mode	03	60 00(2B1Q Training / Uncoded 4-PAM)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)

14.3.15 Configuring an HDSL1 Application

This configuration is valid on M28950 device only.

In this section a list of all the APIs required to make the connection between the HTU-C and HTU-R in the HDSL1 mode. Please note that Autobaud is not in the HDSL1 standard, so when running in HDSL1 mode, the desired Pre-Act mode in _DSL_PREACTIVATION_CFG

(0x0F) API command should be set to “None” and not “Autobaud”. Also note the frame structure in _DSL_SYSTEM_CONFIG (0x06) should be HDSL1_FRAME_FORMAT and not TRANSPARENT_FRAME_FORMAT like SDSL/2B1Q uses.

Table 14-18 API Configuration sequence for a HDSL1 Application

Description	Opcode (HEX)	Parameter(s) [HEX]
Reset System	00	00
System Enable	01	01 (HTU-C) 09 (HTU-R)
System Configuration	06	61 (HDSL1 mode)

Multirate Setting (set data rates)	1B	NPCM 00 NDSL NOCC 01 00 00 00
		Note: NPCM = Number of PCM Timeslots (0x20)
		NDSL = Number of DSL Timeslots (0x24)
		NOCC = Number of occupied PCM Timeslots(0x20)

Training Mode	03	60 00(2B1Q Training, Coded 32-PAM)
PCM Clock Source	12	0A (TPCLK = DPLL; RPCLK = DPLL)
PCM Configuration	10	02 (HTU-C; DPLL Open; Sync Aligned) 00 (HTU-R; DPLL Closed; Sync Aligned)
Pre-Activation Configuration (Default)	0F	00 (Mode = None) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don’t care) 00 (MS Sender = No Remote Configure) 00 (TPS_TC = Do Not Modify PCM/ATM Interface) 00 (Data Rate Source =List) 02 (Annex B) 00(Range i-Bit: don’t care) 00 (Range Min-N: don’t care) 00 (Range Max-N: don’t care) 00 (reserved)
Activate (go)	0B	02 (fixed startup time-out, enable Act Request)

14.3.16 Configuring for Enhanced G.shdsl Higher Rates

Enhanced G.shdsl is an optional extension of the G.991.2 recommendation. It allows user data rates going up to 5696 Kbps. The Enhanced G.shdsl is specified in the Annex F of the G.991.2 recommendation.

The rates for single pair are given by nX64 + iX8 kbits/s. For 16-TCPAM, $3 \leq n \leq 60$ and $0 \leq i \leq 7$. For 16-TCPAM and n=60, the applicable value of i is 0. This corresponds to (payload) data rates from 192 kbit/s to 3840 kbit/s in increments of 8 kbit/s for 16-TCPAM. For 32-TCPAM, $12 \leq n \leq 89$ and $0 \leq i \leq 7$. For 32-TCPAM and n=89, the applicable value of i is 0. This corresponds to (payload) data rates from 768 kbit/s to 5696 kbit/s in increments of 8 kbit/s for 32-TCPAM.

In this example:

- The HTU-C and HTU-R are set up for a Payload Data Rate of 5696 kbps using 32 TCPAM coding.
- The ZipWirePlus HTU-C and HTU-R are master of the clock and they provide the clock to the application.

Table 14-19 Enhanced G.shdsl API Configuration Sequence

Description	Opcode (HEX)	Parameter(s) [HEX]
DSL Reset	00	00 (SW Reset)
System Enable	01	01 (HTU-C) 09 (HTU-R)
System Configuration	06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)	1B	59 00 59 59 01 00 00 00
Training Mode	03	02 00 (G.shdsl Training / Coded 32-PAM/ Symmetric PSD)
PCM Clock Source (Default)	12	0A (TPCLK = DPLL; RPCLK = DPLL)
PCM Configuration	10	03 (HTU-C; DPLL Open; Asynchronous) 01 (HTU-R; DPLL Closed; Asynchronous)
Multi-frame Length (Default)	05	2F 2F (6mS)
Pre-Activation Configuration (Default)	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 04 (MS Sender = No Remote Configure) 00 (TPS_TC = Do not modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00(Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)
Activate (go)	0B	02 (fixed startup time-out, enable Activation Request)

14.3.17 Configuring for Proprietary 4.6Mbps Mode

The ZipWirePlus supports MindSpeed's proprietary 32-PAM line code which can run at rates up to 4.6 Mbps. Below is an example configuration for PCM = 4632kbps and DSL=4640kbps.

Table 14-20 4.6 Mbps Configuration Sequence Example

Description	Opcode (HEX)	Parameter(s) [HEX]
DSL Reset	00	00 (SW Reset)
System Enable	01	01 (HTU-C) 09 (HTU-R)
System Configuration	06	63 (G.shdsl Multirate mode)
Multirate Setting (set data rates)	1B	48 00 48 48 01 03 00 01
Training Mode	03	02 00 (G.shdsl Training / Coded 32-PAM/ Symmetric PSD)
PCM Clock Source (Default)	12	0A (TPCLK = DPLL; RPCLK = DPLL)
PCM Configuration	10	03 (HTU-C; DPLL Open; Asynchronous) 01 (HTU-R; DPLL Closed; Asynchronous)
Multi-frame Length (Default)	05	2F 2F (6mS)
Pre-Activation Configuration (Default)	0F	04 (Mode = G.hs) 00 (Line Probe = disabled) 00 (PBO Mode = Auto) 00 (PBO Value: don't care) 04 (MS Sender = No Remote Configure) 00 (TPS_TC = Do not modify PCM/ATM Interface) 00 (Data Rate Source =List) 01 (Annex A) 00(Range i-Bit: don't care) 00 (Range Min-N: don't care) 00 (Range Max-N: don't care) 00 (reserved)
Activate (go)	0B	02 (fixed startup time-out, enable Activation Request)

15.0 ZipWirePlus API Commands

15.1 API Commands: Quick Reference

Table 15-1 lists a summary of the API commands.

Table 15-1 API Command Summary

HEX	Decimal	Command (C Constant)	In Length	Out Length	Page Ref
		Control Commands			
0	0	_DSL_RESET_SYSTEM	1	N/A	page 15-9
1	1	_DSL_SYSTEM_ENABLE	1	N/A	page 15-10
2	2	_DSL_AFE_CONFIG	1	N/A	page 15-80
3	3	_DSL_TRAINING_MODE	2	N/A	page 15-13
4	4	_DSL_CLOCK_CONFIG	3	N/A	page 15-11
5	5	_DSL_PCM_MF_LEN	2	N/A	page 15-41
6	6	_DSL_SYSTEM_CONFIG	1	N/A	page 15-12
9	9	_DSL_LOOPBACK	1	N/A	page 15-107
B	11	_DSL_ACTIVATION	1	N/A	page 15-16
C	12	_DSL_FORCE_DEACTIVATE	1	N/A	page 15-17
D	13	_DSL_TEST_MODE	1	N/A	page 15-109
E	14	_DSL_DATA_RATE	2	N/A	page 15-73
F	15	_DSL_PREACTIVATION_CFG	12	N/A	page 15-28
10	16	_DSL_FR_PCM_CONFIG	1	N/A	page 15-40
11	17	_DSL_FR_HDSL_CONFIG	1	N/A	page 15-70
12	18	_DSL_PCM_CLK_CONF	1	N/A	page 15-38
13	19	_AFE_TX_GAIN	1	N/A	page 15-81
14	20	_DSL_PREACT_USER_INFO	2	N/A	page 15-30
15	21	_DSL_PREACT_RATE_LIST	2-74	N/A	page 15-30

HEX	Decimal	Command (C Constant)	In Length	Out Length	Page Ref
16	22	_DSL_TX_ISO_PULSE	1	N/A	page 15-110
18	24	_BP_ERLE_TEST_MODE	1	N/A	page 15-111
19	25	_DSL_MULTI_PAIR_CONFIG	1	N/A	page 15-43
1A	26	_DSL_NB_MULTI_RATE_CONFIG	6	N/A	page 15-51
1B	27	_DSL_MULTI_RATE_CONFIG	8	N/A	page 15-14
1C	28	_ATM_PHY_MODE	1	N/A	page 15-37
1D	29	_ATM_PHY_UTOPIA_CONFIG	1	N/A	page 15-36
1E	30	_ATM_PHY_IF_MODE	2	N/A	page 15-33
1F	31	_ATM_PHY_INJECT_HEC_ERROR	1	N/A	page 15-116
20	33	_ATM_PHY_CONFIGURE	2	N/A	page 15-35
21	34	_DSL_TNB_BER_STATE	1	N/A	page 15-126
22	35	_DSL_RNB_BER_STATE	1	N/A	page 15-127
23	35	_DSL_TP_BER_STATE	1	N/A	page 15-123
24	36	_DSL_RP_BER_STATE	1	N/A	page 15-124
25	37	_DSL_PRBS_CONFIGURE	1	N/A	page 15-121
26	38	_DSL_CONST_FILL	1	N/A	page 15-122
27	39	_DSL_DBANK	3	N/A	page 15-117
28	40	_DSL_NB_CONFIG	4	N/A	page 15-53
29	41	_DSL_TNB_FRM_OFST	2	N/A	page 15-55
2A	42	_DSL_RNB_FRM_OFST	2	N/A	page 15-55
2B	43	_DSL_CONFIG_PID	1	N/A	page 15-74
2C	44	_DSL_PCM_WATER_LEVEL	5	NA	page 15-45
2D	45	_DSL_NB_WATER_LEVEL	5	NA	page 15-57
2E	46	_DSL_TP_FRM_OFST	2	N/A	page 15-42
2F	47	_DSL_RP_FRM_OFST	2	N/A	page 15-42
30	48	_DSL_TP_MAPPER_VALUE	65	N/A	page 15-47
31	49	_DSL_TP_MAPPER_WRITE	1	N/A	page 15-48

HEX	Decimal	Command (C Constant)	In Length	Out Length	Page Ref
32	50	_DSL_RP_MAPPER_VALUE	65	N/A	page 15-49
33	51	_DSL_RP_MAPPER_WRITE	1	N/A	page 15-50
34	52	_DSL_TH_MAPPER_VALUE	65	N/A	page 15-75
35	53	_DSL_TH_MAPPER_WRITE	1	N/A	page 15-76
36	54	_DSL_RH_MAPPER_VALUE	65	N/A	page 15-78
37	55	_DSL_RH_MAPPER_WRITE	1	N/A	page 15-78
38	56	_DSL_TNB_MAPPER_VALUE	65	N/A	page 15-58
39	57	_DSL_TNB_MAPPER_WRITE	1	N/A	page 15-59
3A	58	_DSL_RNB_MAPPER_VALUE	65	N/A	page 15-60
3B	59	_DSL_RNB_MAPPER_WRITE	1	N/A	page 15-61
40	64	_DSL_CLEAR_ERROR_CTRS	1	N/A	page 15-113
41	65	_DSL_INJECT_CRC_ERROR	1	N/A	page 15-116
43	67	_DSL_THRESHOLDS	6	N/A	page 15-21
44	68	_E1_PRA_CONFIG	5	N/A	page 15-92
45	69	_E1_PRA_TX_GEN_VALUES	6	N/A	page 15-95
46	70	_E1_PRA_RX_GEN_VALUES	6	N/A	page 15-96
47	71	_E1_PRA_INJECT_CRC_ERROR	2	N/A	page 15-97
48	72	_E1_PRA_TX_CODE	1	N/A	page 15-97
4A	74	_DSL_FR_SET_STATE_MACHINE	1	N/A	page 15-62
4B	75	_DSL_FR_TNB_RESET	1	N/A	page 15-55
4C	76	_DSL_FR_RNB_RESET	1	N/A	page 15-56
4D	77	_EOC_RESET	1	N/A	page 15-92
4E	78	_DSL_FR_TX_RESET	1	N/A	page 15-43
4F	79	_DSL_FR_RX_RESET	1	N/A	page 15-44
50	80	_DSL_INTR_HOST_MASK	1	N/A	page 15-18
51	81	_DSL_INTR_API_SUBMASK	3-41	N/A	page 15-19
53	83	_DSL_DOWNLOAD_START	4	N/A	page 15-8
54	84	_DSL_DOWNLOAD_DATA	1–75	N/A	page 15-8

HEX	Decimal	Command (C Constant)	In Length	Out Length	Page Ref
55	85	_DSL_DOWNLOAD_END	1	N/A	page 15-9
58	88	_DSL_DPLL_CLOCK_GEN	4	N/A	page 15-45
59	89	_DSL_NB_DPLL_CLOCK_GEN	4	N/A	page 15-57
5A	90	_DSL_FR_TXCLK_CONTROL	2	NA	page 15-39
5B	91	_DSL_GHS_STOP_REGEN_SILENCE	1	N/A	page 15-105
5C	92	_DSL_GHS_START_REGEN_SILENCE	1	N/A	page 15-105
5D	93	_DSL_GHS_REGEN_RATE_OVERRIDE	5	N/A	page 15-106
5E	94	_DSL_GHS_REGEN_DIAGNOSTIC	1	N/A	page 15-107
5F	95	_DSL_DSP_CONFIG	1	N/A	page 15-120
60	96	_DSL_WRITE_IND_BITS	2	N/A	page 15-63
61	97	_DPLL_REF_SOURCE	3	N/A	page 15-45
62	98	_DSL_AUTO_IND	2	N/A	page 15-66
63	99	_DSL_WRITE_ZBITS	6 or 12	N/A	page 15-66
64	100	_DSL_WRITE_EOC	3,5,12,18 or 21	N/A	page 15-70
65	101	_DSL_FR_2b1Q_CONFIG	1	N/A	Page 15-74
75	117	_DSL_WRITE_REG	3-75	N/A	page 15-129
76	118	_DSL_WRITE_AFE	2-75	N/A	page 15-130
Status Commands					
80	128	_DSL_READ_CONTROL	1	Up to 75	page 15-28
81	129	_DSL_GHS_GET_FINAL_RATE	1	4	page 15-106
82	130	_DSL_FAR_END_ATTEN	1	1	page 15-81
83	131	_DSL_NOISE_MARGIN	1	1	page 15-82
85	133	_DSL_STATUS	1	8	page 15-21
87	135	_DSL_READ_MIN_MAX_WL	1	8	page 15-117
88	136	_DSL_PREACT_GET_FE_CAPS	3	Up to 75	page 15-32
89	137	_DSL_PREACT_GET_OPT_DATA_RATE	1	2	page 15-33
8A	138	_DSL_VERSIONS	1	11	page 15-26

HEX	Decimal	Command (C Constant)	In Length	Out Length	Page Ref
8C	140	_DSL_TP_BER_RESULTS	1	5	page 15-123
8D	141	_DSL_RP_BER_RESULTS	1	5	page 15-125
8F	143	_DSL_STAGE_NUMBER	1	10	page 15-112
90	144	_DSL_AFE_SETTING	1	1	Page 15-80
91	145	_DSL_TNB_BER_RESULTS	1	5	page 15-126
92	146	_DSL_RNB_BER_RESULTS	1	5	page 15-128
93	147	_BP_ERLE_RESULTS	1	16	page 15-112
94	148	_DSL_POWER_BACK_OFF_RESULT	1	2	page 15-82
9C	156	_DSL_OPER_ERR_CTRS	1	23	page 15-114
9D	157	_DSL_TIME	1	12	page 15-86
9E	158	_DSL_HDSL_PERF_ERR_CTRS	1	10	page 15-83
A0	160	_DSL_READ_REG	3	1-64	page 15-130
A1	161	_DSL_READ_AFE	2	1-64	page 15-131
A2	162	_DSL_SYSTEM_PERF_ERR_CTRS	1	4	page 15-83
A3	163	_DSL_TP_MAPPER_READ	2	Up to 75 (L)	page 15-48
A4	164	_DSL_RP_MAPPER_READ	2	Up to 75 (L)	page 15-50
A5	165	_DSL_TH_MAPPER_READ	2	Up to 75 (L)	page 15-76
A6	166	_DSL_RH_MAPPER_READ	2	Up to 75 (L)	page 15-79
A7	167	_DSL_TNB_MAPPER_READ	2	Up to 75 (L)	page 15-59
A8	168	_DSL_RNB_MAPPER_READ	2	Up to 75 (L)	page 15-61
AA	170	_API_DIAG_GET_CONFIG	1	Up to 75 (L)	page 15-118
AB	171	_API_DIAG_GET_STATUS	1	Up to 75 (L)	page 15-119
AC	172	_API_DIAG_GET_PERF	1	Up to 75 (L)	page 15-119
AE	174	_EOC_RX_GET_STATS	1	5	page 15-90
B0	176	_EOC_TX_SEND_COMMAND	Up to 75	3	page 15-87

HEX	Decimal	Command (C Constant)	In Length	Out Length	Page Ref
			(L)		
B1	177	_EOC_RX_GET_MSG	1	Up to 75 (L)	page 15-88
B2	178	_EOC_TX_GET_MSG_STATUS	1	2	page 15-88
B4	180	_HDSL1_STATUS	1	1	page 15-91
B8	184	_ATM_PHY_OPER_ERR_CTRS	1	5	page 15-115
B9	185	_ATM_PHY_PERF_ERR_CTRS	1	6	page 15-84
BA	186	_ATM_PHY_CELL_CTRS	1	17	page 15-85
C0	192	_E1_PRA_TX_MON_CHANGE	1	1	page 15-98
C1	193	_E1_PRA_RX_MON_CHANGE	1	1	page 15-99
C2	194	_E1_PRA_TX_MON_VALUES	1	6	page 15-100
C3	195	_E1_PRA_RX_MON_VALUES	1	6	page 15-101
C4	196	_E1_PRA_ERROR_CTRS	1	8	page 15-102
C5	197	_E1_PRA_MF_STAT	1	1	page 15-103
C6	198	_E1_PRA_ALARM_STATUS	1	2	page 15-103
D0	208	_DSL_READ_IND_BITS	1	2	page 15-67
D1	208	_DSL_READ_ZBITS	1	6	page 15-70

15.2 API Command Set Documentation Convention

Table 15-2 summaries the conventions used to describe API commands.

Table 15-2 ZipWirePlus API Command Set Documentation Convention

API Command Name			C	R
Description of the API Command.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
C constant found in Dsl_api.h	Opcode value as defined in DSL_API.H	Opcode Type: Control/ Status	Number of Incoming Bytes	Number of Outgoing Bytes
Incoming Parameters				
Byte	**Content**	**Description**		
1	Byte 1 Name	Incoming Byte 1 description.		
2	Byte 2 Name	Incoming Byte 2 description.		
N	Byte N Name	Incoming Byte N description.		
Outgoing Parameters				
Byte	**Content**	**Description**		
1	Byte 1 Name	Outgoing Byte 1 description		
2	Byte 2 Name	Outgoing Byte 2 description		
N	Byte N Name	Outgoing Byte N description		
NOTE: The outgoing data parameter description is provided only for status commands. If required, additional descriptions for the incoming and outgoing data parameters are provided at the end of the command.				

15.3 Download API Commands

15.3.1 Download Start (Length)

Download Start					
This command begins a new download. The size of program code (length) validates the download procedure. This command must be issued before the Download Data command is issued. Issuing the Download Start command in the middle of a download resets the download process.					
NOTE: This command is only supported when the ZipWirePlus device is in boot code.					
C Constant		**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_DOWNLOAD_START		0x53	Control	4	None
Incoming Parameters					
Byte	**Content**	**Description**			
1-4	Length	Two-byte value specifying the PRAM length. Low byte is programmed first.			
NOTE: The ZipWirePlus family requires 2 bytes to specify the PRAM length. However, to maintain compatibility with the CX28985, the ZipWirePlus device can accept both a length of 2 or 4 bytes. When specifying the length to 4 bytes, the upper 2 bytes must be 0x00.					

15.3.2 Download Data

Download Data					
This command transfers the next block of the program data. All download packets should have a data parameter length of 75 bytes or less until the last packet, which contains only the length of byte necessary to complete the download.					
NOTE: This command is only supported when the ZipWirePlus device is in boot code.					
C Constant		**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_DOWNLOAD_DATA		0x54	Control	Up to 75 (L)	None
Incoming Parameters					
Byte	**Content**	**Description**			
1–L	Download Data	Download data. Data is copied sequentially into the next PRAM location.			

15.3.3 Download End (Checksum)

Download End					
This command indicates the end of the download. The download data checksum is passed in so the ZipWirePlus device can validate the download contents.					
NOTE: This command is only supported when the ZipWirePlus device is in boot code.					
C Constant		**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_DOWNLOAD_END		0x55	Control	1	None
Incoming Parameters					
Byte	**Content**	**Description**			
1	Checksum	Checksum of downloaded data content. This does not include the length or checksum bytes.			

15.4 System Configuration and Status API Commands

15.4.1 DSL Reset

DSL Reset					
This command issues a reset to the DSL system. The software reset sets the DSL Reset flag. The program reconfigures the device to the default values. The hardware reset forces the code to jump back to the internal boot ROM code. The host processor needs to re-perform the download procedure. The actual software or hardware reset takes place 50 ms after the API command is acknowledged. This delay gives the host processor time to process the API response. No other API command should be issued until the subsequent _ACK_BOOT_WAKE_UP or _ACK_OPER_WAKE_UP is received (see Section 4.3).					
C Constant		**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_RESET_SYSTEM		0x00	Control	1	None
Incoming Parameters					
Byte	**Content**	**Description**			
1	Reset	0x00 = Software reset (_DSL_SW_RESET) 0x01 = Hardware reset (_DSL_HW_RESET)			

15.4.2 DSL System Enable

DSL System Enable							
This command enables or disables the ZipWirePlus System and sets the terminal type. This command causes the ZipWirePlus device to deactivate any training or test modes and transition to the Configure ZipWirePlus state (see Figure 4-2). The ASM is disabled.							
C Constant			**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**	
_DSL_SYSTEM_ENABLE			0x01	Control	1	None	
Incoming Parameters							
Byte	**Content**		**Description**				
1	System State		See the bit-field description below. Default value = 0x00.				
	Bit	**Content**	**Description**				
	7:5	Reserved	Reserved				
	4:3	Terminal Type	Sets the DSL terminal type. The terminal type determines the activation procedure, timing recovery, scrambler and de-scrambler taps, and more. In a multi-pair system, each bit pump on a board may be individually set as an HTU-C or HTU-R. However, most applications configure all bit pumps on a given board to the same terminal type.				
			Value	**Option**	**Description**	**C Constant**	
			0x00	HTU-C	Central Office Terminal	_DSL_HTUC	
			0x01	HTU-R	Remote Terminal	_DSL_HTUR	
			0x02	REG-C	Regenerator HTU-C	_DSL_REGC	
			0x03	REG-R	Regenerator HTU-R	_DSL_REGR	
	2:1	Reserved	Reserved				
	0	DSL System State	This command enables or disables the ZipWirePlus Transceiver. Setting the ZipWirePlus State to Off (Out-of-Service) puts the chip in a power-down mode. Setting the ZipWirePlus State to On (In-Service) puts the system in a default configuration. Other API commands must then be issued to properly configure the device. 0 = Out-of-Service (default) 1 = In-Service				

15.4.3 DSL Clock Configuration

DSL Clock Configuration				
This command configures the clocking mode and network timing reference options. The network timing reference clock frequency (NTR Clock Frequency) is sourced from either the EXT_CLK_REF pin or i_TPCLK. The clock frequency range is from 8 kHz to 18.432 MHz in 8 kHz resolution. Recommended settings: To disable NTR, set EXT_CLK_REF, NTR Select, and NTR Clock Frequency parameters to 0. The Clocking Mode parameter can still be either the plesiosynchronous (0) or synchronous (2) option. To enable NTR for HTU-C, set: Clocking Mode to desired option EXT_CLK_REF to input (0) NTR Select to desired option (non-zero) NTR Clock Frequency to match clock input. To enable NTR for HTU-R, set: Clocking Mode to desired option (this may get overwritten by the G.hs remote configuration) EXT_CLK_REF to output (1) NTR Select to disabled (0) NTR Clock Frequency to 0 (don't care)				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_CLOCK_CONFIG	0x04	Control	3	None

Incoming Parameters			
Byte	**Content**		**Description**
1	Clock Config		See the bit-field description below.
	Bit	**Content**	**Description**
	7:6	Reserved	Reserved
	5:4	Clocking Mode	Select the clock mode, See Section 5.1 for details. 0 = Plesiosynchronous (default) 1 = Plesiosynchronous with timing reference 2 = Synchronous 3 = Hybrid
	3:1	NTR Select	Select the NTR source 0 = Disabled (default). The software will select either free running (HTU-C) or DSL/G.hs timing recovery (HTU-R) based on the terminal type (_DSL_SYSTEM_ENABLE, opcode 0x01) 1 = Source from 'i_TPCLK', which can either be sourced from the TPCLK pin, PEXTCLK pin, PCM DPLL, or NB DPLL (see _DSL_PCM_CLK_CONF, opcode 0x12) 2 = Source from the EXT_CLK_REF pin 3 - 7 = Reserved.
	0	EXT_CLK_REF Direction	Set the EXT_CLK_REF pin to an input or output. 0 = Input (default). 1 = Output

2-3	NTR Clock Frequency	16-bit unsigned value specifying the network timing reference frequency. The low byte is sent first. This is a 1-based parameter where a 1 means 1 (8kHz). Default is 1 (8 kHz). The formula for programming the API command is: $value = \frac{NTRClockFreq}{8k}$
		NOTE: 16-bit value = (high byte << 8) + (low byte).

15.4.4 DSL System Configuration

DSL System Configuration				
This command configures the ZipWirePlus basic mode of operation. This command pre-configures other API commands, such as the DSL and PCM data rate, DSL line code, training mode, etc. See Section 14.1 for a description of how the ZipWirePlus is configured for different applications. This command causes the ZipWirePlus device to deactivate any training and/or test modes and transition to the Configure ZipWirePlus state (see Figure 4-2). The device is configured based on the settings. The Activation State Manager (ASM) is disabled.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_SYSTEM_CONFIG	0x06	Control	1	None

Incoming Parameters

Byte	Content	Description
1	DSL Configuration	See bit-field description below.

Bit	Content	Description
7:4	DSL Configuration	The DSL Configuration must be set to 0x06
3:0	Frame Structure (Frame Format)	The frame structure determines the relative position of the payload and overhead bits (sync word, EOC, Indicator, etc.) within the DSL frame. See Chapter 16.0 for more details of the various frame structures.

Value	Mode	Description	C Constant
0x00	Framer Bypass	Similar to Framer Transparent Mode except the RPMFSYNC pin outputs the symbol clock of the DSP	_FRAMER_BYPASS_FORMAT
0x01	HDSL1 Frame	Uses the HDSL1 frame structure.	_HDSL1_FRAME_FORMAT
0x02	HDSL2 Frame	Uses the HDSL2 (OPTIS) frame structure.	_HDSL2_FRAME_FORMAT
0x03	G.shdsl Frame	Uses the G.shdsl frame structure (default).	_GSHDSL_FRAME_FORMAT
0x04	RADSL Frame	Uses the RADSL frame structure (ATM over CAP).	_RADSL_FRAME_FORMAT

			0x05	Framer Transparent	Adds no additional DSL Framer overhead.	_FRAMER_TRANSPARENT_FORMAT
			0x06	IDSL Frame	Uses the IDSL Frame Structure	_IDSL_FRAME_FORMAT
			0x07-0x0F	Reserved	___	___

15.4.5 Bit Pump Training Mode

Bit Pump Training Mode				
This command configures the Bit Pump training mode and PSD mask.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_TRAINING_MODE	0x03	Control	2	None

Incoming Parameters						
Byte	**Content**		**Description**			
1	Training Mode		See bit field description below. The default will vary depending on code image. G.shdsl = 0x12 OPTIS (HDSL2) = 0x11 2B1Q (HDSL1, IDSL & SDSL) = 0x60			
	Bit	**Content**	**Description**			
	7:4	DSL Line Code	Sets the desired final DSL line code. The automatic mode will select the appropriate line code based on the training mode.			
			Value	**Mode**	**Line Code**	**C Constant**
			0x00	4-Bit Trellis Coded	32 PAM	_32PAM_CODED_LINE
			0x01	3-Bit Trellis Coded	16 PAM (OPTIS/ G.shdsl)	_16PAM_CODED_LINE
			0x02	2-Bit Trellis Coded	8 PAM	_8PAM_CODED_LINE
			0x03	1-Bit Trellis Coded	4 PAM	_4PAM_CODED_LINE
			0x04	4-Bit Uncoded	16 PAM	_16PAM_UNCODED_LINE
			0x05	3-Bit Uncoded	8 PAM	_8PAM_UNCODED_LINE
			0x06	2-Bit Uncoded	4 PAM (2B1Q)	_4PAM_UNCODED_LINE

			0x07	1-Bit Uncoded	2 PAM (startup only)	_2PAM_UNCODED_LINE
			0x08–0x0F	Reserved	Reserved	Reserved
	3:0	Training Mode	Sets the desired bit pump training mode.			
			Value	**Training Mode**		**C Constant**
			0x00	2B1Q		_2B1Q_TRAINING
			0x01	HDSL2 OPTIS		_OPTIS_TRAINING
			0x02	G.shdsl		_GSHDSL_TRAINING
			0x03–0x0F	Reserved		—
2	PSD Mask Definition		Sets the desired bit pump PSD mask in accordance to the G.shdsl Annexes. This option only applies to the G.shdsl mode. 0x00 = Symmetric (default) 0x01 = Asymmetric			

15.4.6 Multirate Configuration

Multirate Configuration				
This command configures the multirate Configuration parameters. See Section 14.1.2 for more details about the multirate Configuration commands. When the 'Maintain DSL Link' bit (byte #7) is cleared (0), this command causes the ZipWirePlus device to deactivate any training or test modes and transition to the Configure ZipWirePlus state (see Figure 4-2). The device is configured based on the settings. The ASM is disabled. When the 'Maintain DSL Link' bit (byte #7) is set (1), this command will reconfigure only the PCM interface. The DSL domain is not reconfigured. This is useful when the application needs to adjust the PCM interface after the DSL link is established. The pre-activation (G.hs/Auto Baud) phase is used to establish the DSL data rate while the EOC channel is used to establish the application interface (PCM) data rates.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_MULTI_RATE_CONFIG	0x1B	Control	8	None

Incoming Parameters		
Byte	**Content**	**Description**
1-2	Number of PCM Time Slots	Total number of PCM time slots available. A value of 1 implies 1 time slot. The low byte is programmed first. X = 1…128
3	Number of DSL Time Slots	Total number of DSL time slots available. A value of 1 implies 1 time slot. X = 1…89
4	Number of Occupied PCM Time Slots	Total number of occupied time slots available. A value of 1 implies 1 time slot. X = 1…MIN (Total PCM, Total DSL)

5	Start PCM Time Slot Location		Indicates the location of the first time slot to extract from the PCM bus. The first PCM time slot is numbered 1.
6	Number of ‘i-bits’		Total number of ‘i-bits’ available. X = 0…7
7	Mapping Format		See bit-field description below.
	Bit	**Content**	**Description**
	7	Maintain DSL Link	0 = No, configures complete device. 1 = Yes, don’t drop DSL link. Only reconfigure PCM interface. This option should only be used when the DSL link is up.
	6:2	Reserved	Reserved
	1:0	Mapping Mode	Determines either Block mapping or Interleave mapping. 0 =Block Mapping (default) 1 = Interleave Mapping. 2–3= Reserved
8	Interleave Ratio		Determines the interleave ratio on the PCM bus. An Interleave Ratio of 0x01 implies that all PCM timeslots will be used. A value of 0x02 implies using every other timeslot. A value of 0x00 is invalid. This value is only applicable when the Mapping Mode is set to the Interleave Mapping option.

15.4.7 DSL Activation

DSL Activation

This command configures DSL Activation.

> ***NOTE:*** 1: The activation request is disabled for certain diagnostic test modes and loop backs. When disabled, the device performs no automatic functions.
>
> ***NOTE:*** 2: When DSL activation is enabled, the device exits any test modes and loopbacks except PCM/NB interface loopbacks (_FR_HDSL_ON_PCM_LB, _FR_PCM_ON_PCM_LB, _FR_NB_ON_NB_LB, _FR_HDSL_ON_NB_LB) and ATM_SOURCE_LB.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_ACTIVATION	0x0B	Control	1	None

Incoming Parameters

Byte	Content	Description
1	Activation Configuration	See bit-field description below.

Bit	Content	Description
7	Reserved	Reserved.
6	Auto TFIFO Reset	0 = Default. The TFIFO reset occurs normally. 1 = TFIFO reset occurs after the DPLL has stabilized. This should be enabled only when the RPMFSYNC is looped back to the TPMFSYNC.
5:4	Activation Time-Out Setting	This command sets the activation interval time-out setting.

Value	Option	Description	C Constant
0x00	Standards Based	This option is used when running at the standard G.shdsl, HDSL2 and HDSL1 data rates (default value).	_ACT_TIME_STANDARD
0x01	Reserved	Reserved	Reserved
0x02	Variable Number of Symbols	The activation interval is based on data rate rather than fixed time. This option should be used in 2B1Q (SDSL) applications.	_ACT_TIME_VARIABLE

2B1Q (SDSL) Variable Rate Startup Times

Data Rate (kbps)	Typical Startup Time(s)	Activation Time-Out(s)
144	64.4	128.8
288	35.3	70.5

				400	26.3	52.6
				784	16.8	33.7
				1,168	13.3	26.7
				1,552	11.5	23.1
				2,320	9.8	19.5
			0x03	Reserved	Reserved	Reserved
	3:2	LOSWT Additional Time	When LOSW occurs, ZipWirePlus device waits for LOSWT timer to expire. This parameter adds additional time to the default value of the LOSWT timer. 00 = Disabled (Default) which means 2 seconds 01 = Increase LOSWT by 1 sec. which means 3 seconds 10 = Increase LOSWT by 2 sec. which means 4 seconds 11 = Increase LOSWT by 3 sec. which means 5 seconds			
	1	Activation Request State	This command sets the Activation Request flag (ACTREQ). The activation request controls the state of the Activation State Manager. Setting this bit enables the activation request and sets the ASM state to the active state. Clearing this bit disables the activation request. 0 = Disabled (default) 1 = Enabled			
	0	Reserved	Reserved.			

15.4.8 DSL Force Deactivate

DSL Force Deactivate				
This command forces the ZipWirePlus system to deactivate. If the Activation State Manager is enabled, the system then re-performs the startup. If in normal operation, the Activation State Manager proceeds through the pending deactivated state. The force-deactivate flag is then cleared when the Activation State Manager reaches the deactivated state. If the system is in the process of a startup, the ZipWirePlus immediately fails that startup attempt. This command allows the host processor to easily control the startup state machine if the host processor detects a higher-level error.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_FORCE_DEACTIVATE	0x0C	Control	1	None

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x01 for future compatibility.

15.4.9 Unsolicited Interrupts Mask

Unsolicited Interrupt Mask

This command will turn on/off unsolicited interrupt for ASM transitions or Unsolicited API commands. Disabling unsolicited API commands will globally disable all unsolicited interrupts for API commands. Individual API commands need to be selectively enabled via the _DSL_INTR_API_SUBMASK (0x51) API command.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_INTR_HOST_MASK	0x50	Control	1	None

Incoming Parameters

Byte	Content		Description
1	Interrupt Configuration		See the bit-field description below.
	Bit	**Content**	**Description**
	7:4	Reserved	Reserved
	3	Dying Gasp	The dying gasp bit is used to enable or disable unsolicited interrupts due to events triggered by Dying Gasp report. 0–Disable unsolicited interrupts from the Dying Gasp. 1–Enable unsolicited interrupts from the Dying Gasp.
	2	Reserved	Reserved
	1	ASM Transition	The ASM Transition bit is used to enable or disable unsolicited interrupts due to events triggered by the ASM. 0–Disable unsolicited interrupts from the ASM. 1–Enable unsolicited interrupts from the ASM.
	0	Unsolicited API Commands	The Unsolicited API Commands bit is used to enable or disable unsolicited interrupts due to API commands. 0–Disable unsolicited interrupts for API commands. 1–Enable unsolicited interrupts for API commands.

15.4.10 Unsolicited Interrupts API Command Sub Mask

Unsolicited Interrupts API Command Sub Mask				
This command will turn on/off unsolicited API commands which need to generate interrupts to the host. A maximum of 20 API commands can be specified at one time with the _DSL_INTR_API_SUBMASK (0x50) API command.				
NOTE: Currently, the ZipWirePlus device supports the _EOC_RX_GET_MSG (0xB1), _DSL_READ_ZBITS (0xD1), _*E1*_PRA_TX_MON_CHANGE (0xC0), *E1*_PRA_RX_MON_CHANGE (0xC1). Any other API opcode will be ignored				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_INTR_API_SUBMASK	0x51	Control	3-41 (L)	None

Incoming Parameters

Byte	**Content**		**Description**
1	Count		The count of how many API commands in the list. This counter is 1 based; a 1 implies 1 entry.
2	Opcode		Opcode of the API command for which unsolicited interrupts is to be enabled or disabled. The enable or disable option is specified in the next byte.
3	Configuration		
	Bit	**Content**	**Description**
	HDSL1 Mode Only		
	7:3	Reserved	Reserved
	2	Indicator-Bits	If the opcode is _EOC_RX_GET_MSG, this bit specifies the type of information delivered to the host through this API command. (Valid for HDSL1 Mode Only) 0 = Z-bits should not be sent to the host 1 = EOC bits, Z-bits and indicator bits should be sent to the host.
	1	Z-Bits	If the opcode is _EOC_RX_GET_MSG, this bit specifies the type of information delivered to the host through this API command. (Valid for HDSL1 Mode Only) 0 = Only EOC bits should be sent to the host 1 = EOC bits and Z-bits should be sent to the host.
	0	Enable/Disable	0 = Disable interrupt for the API specified by opcode in previous byte.. 1 = Enable interrupt for the API specified by opcode in previous byte.
	IDSL NT Mode Only		

	7:1	Indicator-Bits	For IDSL, If the opcode is _DSL_READ_IND_BITS, the host can specify the subset of indicator bits using Bits 7-1 as defined below. By setting the corresponding bit, the ZipWirePlus will constanly monitor the specified indicator bit coming from the DSL line and notify the host if a change in the value is detected. The bit mask for the NT: Bit 7: sai Bit 6: cso Bit 5:ntm Bit 4: ps2 Bit 3: febe Bit 2: ps1 Bit 1: act
	0	Enable/Disable	0 = Disable interrupt for the API specified by opcode in previous byte.. 1 = Enable interrupt for the API specified by opcode in previous byte.
…	…		…
L-1	Opcode		Opcode of the API command for which unsolicited interrupts is to be enabled or disabled. The enable or disable option is specified in the next byte.
L	Configuration		
	Bit	**Content**	**Description**
	HDSL1 Mode Only		
	7:3	Reserved	Reserved
	2	Indicator-Bits	If the opcode is _EOC_RX_GET_MSG, this bit specifies the type of information delivered to the host through this API command. (Valid for HDSL1 Mode Only 0 = Z-bits should not be sent to the host 1 = EOC bits, Z-bits and indicator bits should be sent to the host.
	1	Z-Bits	If the opcode is _EOC_RX_GET_MSG, this bit specifies the type of information delivered to the host through this API command. (Valid for HDSL1 Mode Only) 0 = Only EOC bits should be sent to the host 1 = EOC bits and Z-bits should be sent to the host.
	0	Enable/Disable	0 = Disable interrupt for the API specified by opcode in previous byte.. 1 = Enable interrupt for the API specified by opcode in previous byte.
	IDSL NT Mode Only		

	7:1	Indicator-Bits	For IDSL, If the opcode is _DSL_READ_IND_BITS, the host can specify the subset of indicator bits using Bits 7-1 as defined below. By setting the corresponding bit, the ZipWirePlus will constanly monitor the specified indicator bit coming from the DSL line and notify the host if a change in the value is detected. The bit mask for the NT: Bit 7: sai Bit 6: cso Bit 5:ntm Bit 4: ps2 Bit 3: febe Bit 2: ps1 Bit 1: act
	0	Enable/Disable	0 = Disable interrupt for the API specified by opcode in previous byte.. 1 = Enable interrupt for the API specified by opcode in previous byte.

15.4.11 DSL Thresholds

DSL Thresholds				
This command sets the thresholds for various alarms and events.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_THRESHOLDS	0x43	Control	6	None

Incoming Parameters		
Byte	**Content**	**Description**
1	Dying Gasp	Unsigned 1-byte value specifying the number of DSL Frames before reporting the Dying Gasp (power status) alarm. Default is 0x03.
2	NMR (SNR)	Signed 1-byte value specifying the NMR threshold. The value is specified in 0.5 dB increments. When the modem detects the actual NMR below this threshold, the modem will drop the link and try to retrain. Default is –10 (–5dB).
3	Attenuation	Unsigned 1-byte value specifying the Attenuation threshold. The value is specified in 0.5dB increments. Default value is 0xFF (127.5dB). (NOT SUPPORTED)
4	Target Line Probe NMR (SNR)	Signed 1-byte value specifying the target line probe NMR threshold. The value is specified in 0.5 dB increments. Default is +10 (+5dB).
5	Reserved	Set to 0x00 for future compatibility.
6	Reserved	Set to 0x00 for future compatibility.

15.4.12 DSL Status

DSL Status				
This command queries the DSL Status registers. These status bytes provide dynamic information about the ZipWirePlus system. **Note1**: DSL_STATUS (0x85), _ATM_PHY_PERF_ERR_CTRS (0xB9) and _ATM_PHY_CELL_CTRS (0xBA) API commands convey the ATM OAM parameters to the host upon the request. The fields (nlcd defect and nlcd failure bits) are part of STATUS_2 byte as defined below.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_STATUS	0x85	Status	1	8

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters		
1	STATUS_1	The STATUS_1 contains basic status of the system. The Host Processor should only be required to poll this one status register to see if the system is functioning, i.e., simple Go / No Go. See the following STATUS_1: DSL Status Bit Definitions for complete definition.
2	STATUS_2	This byte contains ATM Operation and Maintenance (OAM) Parameters.

STATUS_1: DSL Status Bit Definitions		
Status_1 Bit	**Description**	**Bit Definition**
7–6	Activation Status	00 = Idle (_ASM_STAT_IDLE) 01 = Normal Operation (_ASM_STAT_SUCCESS) 10 = Deactivated (_ASM_STAT_DEACTIVATED) 11 = In-Progress (_ASM_STAT_IN_PROGRESS)
5	Fatal Error	0 = No 1 = Yes
4	NMR OK—Line Quality	0 = No (poor line quality) 1 = Yes (good line quality)
3	NTR Lock	0 = No 1 = Yes
2	LOSW—Loss of Sync Word	0 = No 1 = Yes
1	Dying Gasp	0 = No 1 = Yes
0	LOS—Loss of Signal	0 = No 1 = Yes

		STATUS_2: OAM Bit Definition		
		Status_2 Bit	**Description**	**Bit Definition**
		7-3	Reserved	Reserved.
		2	*nlcd* : Near-End Loss of Cell Delineation failure.	0 = No 1 = Yes
		1	*ncld :* Near-end Loss of Cell Delineation defect.	0 = No 1 = Yes
		0	*nhec :* Near-end Header Error Control anomaly.	**0 = No** **1 = Yes**
3	STATUS_3	*Startup Failure Status Bit Definition* for a list of the Startup Failure Status bit definitions.		
		STATUS_3: Startup Failure Status Bit Definition		
		Status_2 Bit	**Description**	**Bit Definition**
		7	Activation Time-Out.	0 = No 1 = Yes
		6	Failed 3 (or n) consecutive startup attempts. N is an API command.	0 = No 1 = Yes
		5-4	Reserved.	Reserved.
		3-0	Activation FailureResult will be latched until next successful or failed startup attempt.	0 = None 1 = Bad NMR 2 = Unable to Frequency Lock 3 = Failed Pre-Activation Startup 4 = Unable to detect Sync Word 5 = Failed Pair ID
4	STATUS_4	See the following STATUS_4: DSL Framer Status Bit Definitions for a list of DSL Framer Status bit definitions.		
		STATUS_4: DSL Framer Status Bit Definitions		
		Status_4 Bit	**Description**	**Bit Definition**
		7–6	DSL Sync State.	00 = Out-of-Sync 01 = Acquiring Sync 10 = In-Sync 11 = Losing Sync

		5	Tip/Ring Reversal	0 = No 1 = Yes
		4	Receive FIFO Error.	0 = No 1 = Yes
		3	Transmit FIFO Error.	0 = No 1 = Yes
		2	Transmit Stuff Error.	0 = No 1 = Yes
		1	NB DPLL Locked—only valid in NB DPLL Closed Loop mode.	0 = Not locked 1 = Locked, DPLL Stable
		0	DPLL Locked—only valid in DPLL Closed Loop mode.	0 = Not locked 1 = Locked, DPLL Stable
5	STATUS_5	See STATUS_5: ATM PHY Status Bit Definitions for a list of ATM PHY status bit definitions.		
		STATUS_5: ATM PHY Status Bit Definitions		
		Status_5 Bit	**Description**	**Bit Definition**
		7	Parity error	0 = No 1 = Yes
		6	Reserved	Reserved
		5	Transmit FIFO overflow error	0 = No 1 = Yes
		4	Receive FIFO overflow error	0 = No 1 = Yes
		3	Cell Sent	0 = No 1 = Yes
		2	Bus Conflict error	0 = No 1 = Yes
		1	Reserved	Reserved
		0	LOCD–Loss of cell delineation.	0 = No 1 = Yes
6	STATUS_6	STATUS_6: DSL Framer status bit definitions		
		Status_6 Bit	**Description**	**Bit Definition**
		7–2	Reserved.	Reserved
		1	Invalid TNBCLK Detected	0 = No 1 = Yes

		0	Invalid TPCLK Detected	0 = No 1 = Yes
7	STATUS_7	Reserved.		
8	STATUS_8	Unsolicited Interrupt Bit Definitions. See also _DSL_INTR_HOST_MASK (opcode 0x50)		
		STATUS_8: Unsolicited Interrupt Bit Definitions		
		Status_8 Bit	**Description**	**Bit Definition**
		7–4	Reserved.	Reserved
		3	Dying Gasp -- If HTU-R has a power failure, HTU-R sends power status indicator bits (active low) to HTU-C. HTU-C will report dying gasp unsolicited interrupt to the host	0 = No 1 = Yes
		2	Reserved	Reserved
		1	Activation State Manager (ASM) Transition—this bit is set when the ASM transitions into the Activate State (normal operation) or when the ASM transitions from the Pending State to the Deactivated State. The host reads the STATUS_1, Activation Status bits to determine the link-up or link-down status. In summary, this bit only provides link-down to link-up transition and link-up to link-down transition.	0 = No 1 = Yes
		0	Unsolicited API Commands–this bit is set when an API command caused the Unsolicited Interrupt. The host reads the API Outgoing mailbox to determine the API contents.	0 = No 1 = Yes

15.4.13 Versions

Versions				
Requests the ZipWirePlus hardware, software, and silicon version numbers.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_VERSIONS	0x8A	Status	1	11

Incoming Parameters

Byte	Content	Description
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters

Byte	Content	Description	
1	Major Software Version	1-byte unsigned integer field returning the major software version.	
2	Minor Software Version	1-byte unsigned integer field returning the minor software version.	
3	Internal Software Version	1-byte unsigned integer field returning internal software version. Used during S/W development. This field will return a 0x00 for all official releases.	
4	Compiler Build Option Low	Low byte of 2-byte field containing the compiler build option.	
		Bit #	**Description**
		7	Reserved
		6	G.shdsl
		5	Reserved
		4	HDSL1 (SDSL)
		3	HDSL2 (OPTIS)
		2	Regenerator
		1	HTU-R
		0	HTU-C
5	Compiler Build Option High	Upper byte of the compiler build option field.	
		Bit #	**Description**
		7	Multi-Pair
		6	NTR
		5	E1 Framer
		4	RADSL
		3	Narrowband

		2	ATM
		1	TDEBUG
		0	Line Probe
6	DSP Silicon Type	1-byte unsigned integer field returning the DSP silicon type.	
		Byte Value	**Type**
		0x00	Bt8952
		0x01	Bt8960
		0x02	Bt8970
		0x03	RS8973
		0x75	ZipWirePlus family
		0x85	M28985
		NOTE: The legacy 2B1Q devices are listed because this information can be retrieved across the EOC channel.	
7	DSP Silicon Revision	1-byte unsigned integer field returning the DSP silicon revision.	
		Byte Value	**ZipWirePlus Revision**
		0x03	X.2
		0x04	X.3
8	AFE Silicon Type	1-byte unsigned integer field returning the AFE silicon version type.	
		Byte Value	**Type**
		0x00	M28927
9	AFE Silicon Revision	1-byte unsigned integer field returning the AFE silicon revision.	
		Byte Value	**AFE Revision**
		0x00	X.0
		0x01	X.3
		0x08	X.5
10	Reserved	Reserved	
11	Reserved	Reserved	

15.4.14 Read DSL Control Commands

Read DSL Control Commands					
The Read DSL Control Commands API command is a read-back of the current setting for each of API control commands.					
C Constant		Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_READ_CONTROL		0x80	Status	1	Up to 75 (L)
Incoming Parameters					
Byte	Content	Description			
1	Control Command Opcode	Return the configuration for the specified control command. Input range is 0–127 (0x7F).			
Outgoing Parameters					
Byte	Content	Description			
1-L	—	Refer to the specific control command.			

15.5 Pre-Activation Configuration and Status API Commands

15.5.1 DSL Pre-Activation Mode

DSL Pre-Activation Mode					
This command configures the pre-activation mode. This command takes effect on subsequent startups.					
C Constant		Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_PREACTIVATION_CFG		0x0F	Control	12	None
Incoming Parameters					
Byte	Content	Description			Applicable Standard
1	Pre-Act Mode	Set the desired pre-activation mode: 0x00 = None 0x01 = Auto Baud 0x02 = HDSL2 Pre-Act 0x03 = Reserved 0x04 = G.hs (default)			G.Shdsl 2B1Q (SDSL) HDSL2 HDSL1
2	Line Probe Enable	Enables or disables the line probe protocol. 0x00 = Disabled (default) 0x01 = Enabled			G.Shdsl 2B1Q (SDSL)

3-4	Reserved	Set to 0x00 for future compatibility.	G.Shdsl 2B1Q (SDSL) HDSL2 HDSL1
5	Mode Select Sender	Determines which modems sends the Mode Select API command, see Section 7.1.2.2. This byte is set to 0x00 when not applicable to the selected standard. 0x00 = HTU-C sends Mode Select message (default) 0x01 = HTU-R sends Mode Select message 0x04 = No remote configuration, use local configuration	G.Shdsl
6	TPS-TC Config	Specify the local PCM TPS-TC configuration after G.hs is completed. 0x00 = Do not modify PCM/ATM interface 0x01 = Clear Channel 0x02 = Clear Channel Byte Oriented 0x04 = Unaligned DS1 (E1/T1) 0x08 = Aligned DS1/Fractional DS1 (E1/T1) 0x10 = ATM (default) 0x20 = Synchronous ISDN-BRA	G.Shdsl 2B1Q (SDSL)
7	Line Probe Configuration	**Bit #** — **Description** 7:5 — Reserved 4 — PSD Mask 0 = symmetric 1 = asymmetric 3:2 — Reserved 0:1 — Data Rate Source Specifies how the modem determines the final data rate, see Section 7.1.2.3. 0x00 = List (default) 0x01 = Reserved 0x02 = All (only applicable to Auto Baud (2B1Q), recommended for HTU-R)	G.Shdsl **2B1Q (SDSL)**
8	Annex Type	This byte is set to 0x00 when not applicable to the selected standard. 0x00 = Reserved 0x01 = Annex A (default) 0x02 = Annex B 0x03 = Annex F (g.shdsl.bis)	G.Shdsl
9	i-bits Mask	i-bit mask. See Data Rate Entry, Specifying the i-bit Mask on page 7-6 for details. This byte is set to 0x00 when not applicable to the selected standard.	G.Shdsl 2B1Q (SDSL)

10-12	Reserved	Set to 0x00 for future compatibility.	G.Shdsl 2B1Q (SDSL) HDSL2 HDSL1

15.5.2 DSL Pre-Activation (G.hs) User Information

DSL Pre-Activation (G.hs) User Information				
This command configures the G.hs user information. This information is transported within the G.hs transactions. The ZipWirePlus does not make any decisions based on this information.				
NOTE: The incoming data parameters will differ when using G.hs versus Auto Baud (2B1Q).				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_PREACT_USER_INFO	0x14	Control	2	None

Incoming Parameters (G.hs application)		
Byte	**Content**	**Description**
1-2	Country Code	G.hs country code. (default = 0x0000)
Incoming Parameters (Auto Baud application)		
Byte	**Content**	**Description**
1	Layer 1/2 Info Byte #1	Layer 1/2 information Byte #1. See Section 7.2.2.6. (default = 0x00)
2	Layer 1/2 Info Byte #1	Layer 1/2 information Byte #2. See Section 7.2.2.6. (default = 0x00)

15.5.3 Pre-Activation Data Rate List

Pre-Activation Data Rate List				
This command specifies the list of supported payload data rates. This command is only applicable when the Data Rate Source (byte #6) in the _DSL_PREACTIVATION_CFG (0x0F) API command is set to the List option (0x00). The i-bits mask (byte #1) of the _DSL_PREACTIVATION_CFG (0x0F) API command applies to all N x 64k entries in the list table. The modem can support up to 72 (N x 64k) entries. The N x 64k values must be in increasing order (lowest rate first and highest rate last).				
NOTE: The incoming data parameters will differ when using G.hs versus Auto Baud (2B1Q).				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_PREACT_RATE_LIST	0x15	Control	Up to 75(2 + L where L is the number of 'N x 64k' entries)	None

Incoming Parameters (G.hs applications)		
Byte	**Content**	**Description**
1	LineProbe or Fixed Rate	1 byte specifying if the rate list is to be used for the line probe mode or for the fixed rate mode. 0x00: Line Probe Mode (default) 0x01: Fixed Rate Mode – only supports one entry.
2	Num N Entries (L)	1-byte specifying the number of N x64k entries. The value is one based, i.e., a value of 1 implies 1 entry.
3	1st N x 64k	First entry.
4	2nd N x 64k	Second entry.
2+L	Lth N x 64k	Last entry.
Incoming Parameters (Auto Baud applications)		
Byte	**Content**	**Description**
1	Num Entries (L)	1-byte specifying the number of N x 64k + 8 x 8k entries. The value is zero based, i.e., a 0 value implies 1 entry.
2	1st Entry: N	First entry N value.
3	1st Entry: I	First entry i value.
4	2nd Entry: N	Second entry N value.
5	2nd Entry: i	Second entry i value.
2xL	Lth Entry: N	Last entry N value.
2xL+1	Lth Entry: I	Last entry i value.

15.5.4 Pre-Activation Get Far-End Capabilities

Pre-Activation Get Far-End Capabilities					
This command returns the far-end capabilities information exchanged during the pre-activation. For G.hs applications, the ZipWirePlus firmware parses the G.hs messages and stores the information into a manageable array. The far-end capabilities list message can theoretically be up to any length. The host might need to issue this command multiple times to read out the entire data contents. This command is provided as a debug tool in the event the two modems don't reach a common configuration. The host processor can use this information to re-configure the modem to increase the likelihood of a successful training. ***NOTE:*** This API is not supported for G.hs applications. For Auto Baud applications, this command returns the layer 1 and layer 2 type information; which is always 2 bytes. The first byte contains the Byte #1 information while the second byte contains the Byte #2 information. The starting address must be 0x0000 while the length must be 0x01 (2 bytes). The host processor would use this information to configure the layer 1/2 parameters.					
NOTE: The outgoing data parameters will differ when using G.hs versus Auto Baud					
C Constant		**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_PREACT_GET_FE_CAPS		0x88	Status	3	Up to 75
Incoming Parameters					
Byte	**Content**	**Description**			
1-2	Starting Address	2-byte field specifying the starting address (index) of the capabilities list message. Low byte first.			
3	Length (L)	1-byte field specifying the number of bytes to read. A 0x00 implies 1 byte.			
Outgoing Parameters (G.hs)					
Byte	**Content**	**Description**			
1-L	Data	Block of data. The first byte corresponds to the specified address, the second byte to the address + 1, etc.			

15.5.5 Pre-Activation Get Optimal Data Rate

Pre-Activation Get Optimal Data Rate				
This command returns optimal data rate determined during line probe. This value may be different then the final data rate chosen because the rate list may not contain the optimal data rate For example, the user may enter in 1152 kbps and 1536 kbps but the line probe may determine the optimal rate to be 1280 kbps. The final data rate would be 1152 kbps.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_PREACT_GET_OPT_DATA_RATE	0x89	Status	1	2

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.
Outgoing Parameters		
Byte	**Content**	**Description**
1-2	Optimal Data Rate	2-byte field specifying the optimal data rate. Low byte sent first. Data Rate = value * 8k

15.6 ATM Interface Configuration API Commands

15.6.1 ATM PHY Interface Mode

ATM PHY Interface Mode				
This command configures the ATM PHY Interface operating mode (see Section 6.1).				
Opcode C Constant	**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_ATM_PHY_IF_MODE	0x1E	Control	2	None

Incoming Parameters			
Byte	**Content**		**Description**
1	ATM PHY Interface Mode		See bit-field description below. ***NOTE:*** Control bits 1–7 are only valid if the ATM Enable (bit 0) is set.
	Bit	**Content**	**Description**
	7	Reserved	Set the bit to ‘0’ for future compatibility.

	6	One Second Master/Slave	Set the ATM one-second master/slave mode. 0 = Master (default), OneSecOut is connected to the DSLSYNCO pin and connected back into the ATM OneSecIn input. 1 = Slave. OneSecIn input is connected from DSLSYNCI Input pin.		
	5	ATM Serial Interface Mode	Enable or disable the ATM serial interface mode. 0 = Disable (default), use the UTOPIA interface 1 = Enable, use the ATM serial interface		
	4:3	ATM Receive Interface Mode	Determines the receive path connections.		
			Value	**Mode**	**Description**
			0x00	Framer Bypass	ATM receive block is connected to the DSP interface.
			0x01	DSL Framer Aux	ATM receive block is connected to the DSL Framer auxiliary interface. The ATM PHY TC is configured to General Purpose mode.
			0x02	T1/E1 Mode	ATM receive block is connected to the DSL Framer PCM interface. The ATM PHY TC is configured to E1 or T1 mode.
			0x03	General Purpose	ATM receive block is connected to the DSL Framer PCM interface. The ATM PHY TC is configured to General Purpose mode. (default)
	2:1	ATM Transmit Interface Mode	Determines the transmit path connections.		
			Value	**Mode**	**Description**
			0x00	Framer Bypass	ATM transmit block is connected to the DSP interface.
			0x01	DSL Framer Aux	ATM transmit block is connected to the DSL Framer auxiliary interface. The ATM PHY TC is configured to General Purpose mode.
			0x02	T1/E1 Mode	ATM transmit block is connected to the DSL Framer PCM interface. The ATM PHY TC is configured to E1 or T1 mode.
			0x03	General Purpose	ATM transmit block is connected to the DSL Framer PCM interface. The ATM PHY TC is configured to General Purpose mode. (default)
	0	ATM Enable	Enable or disable the ATM PHY TC section of the device. 0 = ATM block is disabled and bypassed (Default). 1 = ATM block is connected and functional.		
2	ATM Byte Alignment		Enable or disable byte level alignment for the ATM Block. 0 = Unaligned 1 = Aligned (default)		

15.6.2 ATM PHY Configuration

ATM PHY Configuration				
This command configures miscellaneous modes of the ATM PHY block.				
Opcode C Constant	**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_ATM_PHY_CONFIGURE	0x20	Control	2	None

Incoming Parameters						
Byte	**Content**		**Description**			
1	UTOPIA Address		Set the UTOPIA address, range 0x00 to 0x1E. 0x1F is reserved for the NULL address. Default is 0x00.			
2	ATM Config		See bit-field description below.			
	Bit	**Content**	**Description**			
	7:4	Reserved	Set the bits to '0' for future compatibility.			
	3	Auto Correct HEC	Enable or disable the auto correct HEC feature. 0 = Disable 1 = Enable (default)			
	2	HEC Coset	Enable or disable the HEC Coset. 0 = Disable 1 = Enable (default)			
	1:0	Scrambler Mode	Set the desired ATM Scrambler mode: Self Synchronizing Scrambler (SSS) or Distributed Sample Scrambler (DSS).			
			Value	**Mode**	**Description**	
			0x00	Off	Disable scrambler/de-scrambler	
			0x01	SSS	Enable SSS mode (default)	
			0x02	DSS	Enable DSS mode	
			0x03	N/A	Reserved	

15.6.3 ATM PHY UTOPIA Configuration

ATM PHY UTOPIA Configuration				
This command configures the ATM PHY UTOPIA interface mode.				
Opcode C Constant	**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_ATM_PHY_UTOPIA_CONFIG	0x1D	Control	1	None

Incoming Parameters

Byte	Content	Description
1	UTOPIA Config	See bit-field description below.

Bit	Content	Description
7:6	Reserved	Set the bits to ‘0’ for future compatibility.
5:4	Bus Width	Set the ATM PHY UTOPIA bus width.
3:2	Reserved	Set the bits to ‘0’ for future compatibility.
1:0	UTOPIA Mode	Set the ATM PHY UTOPIA mode.

Bits 5:4 Bus Width:

Value	Mode	C Constant
00	8-Bit	_ATM_UTOPIA_8BIT_MODE
01	16-Bit	_ATM_UTOPIA_16BIT_MODE (default)
10–11	Reserved	Reserved

Bits 1:0 UTOPIA Mode:

Value	Mode	C Constant
00	UTOPIA Level 1	_ATM_UTOPIA_LEVEL1_MODE
01	UTOPIA Level 2	_ATM_UTOPIA_LEVEL2_MODE (default)
10–11	Reserved	Reserved

15.6.4 ATM PHY Mode

ATM PHY Mode				
This command sets the ATM PHY mode. This command is properly configured by the DSL System Configuration command (opcode 0x06) and ATM PHY Interface Mode command (opcode 0x1E) API commands. The ATM PHY mode is set to T1 or E1 mode whenever a system configuration of T1 or E1 is selected. For Multirate and non-standard applications, the ATM PHY mode is set to general purpose (generic) mode.				
Opcode C Constant	**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_ATM_PHY_MODE	0x1C	Control	1	None

Incoming Parameters

Byte	Content	Description
1	ATM PHY Mode	See bit-field description below.

Bit	Content	Description
7:4	Reserved	Set the bits to '0' for future compatibility.
3:0	ATM PHY Mode	Set the desired ATM PHY Mode. See the *ATM PHY Mode Bit Definitions* table below.

ATM PHY Mode Bit Definition

Value	Mode	C Constant
0x00	T1	_ATM_PHY_T1_MODE
0x01	E1	_ATM_PHY_E1_MODE
0x02	DS3 (1)	_ATM_PHY_DS3_MODE
0x03	E3 (1)	_ATM_PHY_E3_MODE
0x04	J2 (1)	_ATM_PHY_J2_MODE
0x05 (default)	General Purpose	_ATM_PHY_GENERAL_MODE
0x06	Disable	_ATM_PHY_POWER_DOWN
0x07- 0x0F	Reserved	—

NOTE: The DS3, E3, and J2 options are not supported by the ZipWirePlus device. These options are listed to provide a consistency between other Mindspeed product families.

15.7 PCM Interface Configuration API Commands

15.7.1 PCM Clock Configuration

PCM Clock Configuration				
Configures the PCM Transmit and Receive Clock sources.				
Opcode C Constant	**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_PCM_CLK_CONF	0x12	Control	1	None

Incoming Parameters

Byte	Content		Description
1	PCM Clock Configuration		See bit-field description below.
	Bit	**Content**	**Description**
	7:6	RP_CLK_TRAIN	Receive PCM clock source during training. The RPCLK is set to this option when the startup is initiated and the RPCLK will be put back to the RP_CLK_SOURCE after the startup is complete (for both successful and failed startups). 00 = Normal, no change. Use RP_CLK_SOURCE during training (default) 01 = TPCLK input pin 10 = PEXTCLK input 11 = Reserved
	5	Reserved	Set the bit to ‘0’ for future compatibility.
	4	TP_CLK_POL	Transmit PCM clock polarity. 0 = Normal clock—rising edge outputs, falling edge inputs (default) 1 = Inverted clock—falling edge outputs, rising edge inputs
	3:2	RP_CLK_SOURCE	Receive PCM clock source. 00 = TPCLK input pin 01 = PEXTCLK input 10 = PCM DPLL (default) 11 = Narrowband DPLL
	1:0	TP_CLK_SOURCE	Transmit PCM clock source. 00 = TPCLK input pin 01 = PEXTCLK input 10 = PCM DPLL (default) 11 = Narrowband DPLL

15.7.2 DSL Framer Transmit Clock Control

DSL Framer—Transmit Clock Control					
This command controls the TPCLK & TNBCLK behavior when an invalid TPCLK or TNBCLK is detected. An invalid clock is caused by a missing or mismatched (incorrect frequency) TPCLK or TNBCLK.. ***NOTE:*** The STATUS_6: Invalid TPCLK Detect and Invalid TNBCLK Detect bits are cleared when this command is issued.					
Opcode C Constant		**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_FR_TXCLK_CONTROL		0x5A	Control	2	None

Incoming Parameters

Byte	Content		Description
1	PCM Configuration		See bit-field description below.
	Bit	**Content**	**Description**
	7	Auto TPCLK Switch	When enabled, this option will automatically switch the TPCLK source whenever an invalid TPCLK is detected. 0 = Disable, don't change TPCLK source or DPLL Mode (default) 1 = Enable
	6:3	Reserved	Set the bits to '0' for future compatibility.
	2	DPLL_MODE_ERROR	DPLL Mode when an invalid TPCLK is detected and Auto TPCLK Switch is Enabled. 0 = Don't change PCM DPLL Mode (default) 1 = Set PCM DPLL Mode to Open Loop
	0:1	TPCLK_SOURCE_ERROR	Transmit PCM clock source when an invalid TPCLK is detected and Auto TPCLK Switch is Enabled. 00 = TPCLK input pin 01 = PEXTCLK input 10 = PCM DPLL (default) 11 = Narrowband DPLL
2	NB Configuration		See bit-field description below.
	Bit	**Content**	**Description**
	7	Auto TNBCLK Switch	When enabled, this option will automatically switch the TNBCLK source whenever an invalid TNBCLK is detected. 0 = Disable, don't change TNBCLK source or DPLL Mode (default) 1 = Enable
	6:3	Reserved	Set the bits to '0' for future compatibility.
	2	DPLL_MODE_ERROR	DPLL Mode when an invalid TNBCLK is detected and Auto TNBCLK Switch is Enabled. 0 = Don't change NB DPLL Mode (default) 1 = Set NBDPLL Mode to Open Loop

	0:1	TNBCLK_SOURCE_ ERROR	Transmit PCM clock source when an invalid TNBCLK is detected and Auto TNBCLK Switch is Enabled. 00 = TNBCLK input pin 01 = PEXTCLK input 10 = Narrowband DPLL (default) 11 = PCM DPLL

15.7.3 DSL Framer — PCM Configuration

DSL Framer—PCM Configuration				
This command reconfigures the ZipWirePlus DSL Framer PCM block.				
Opcode C Constant	**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_FR_PCM_CONFIG	0x10	Control	1	None

Incoming Parameters			
Byte	**Content**		**Description**
1	PCM Configuration		See bit-field description below.
	Bit	**Content**	**Description**
	7:6	PCM/NB Mode	Set the PCM and Narrowband active ports. 00 = PCM Only (default) 01 = Narrowband Only 10 = PCM + Narrowband 11 = Reserved
	5	TPMFSYNC Direction	Configures the input/output direction of the transmit PCM multi-frame pin (TPMFSYNC). 0 = Input (default) 1 = Output (execute last before API 0x0B)
	4	RPMFSYNC Mode	Configures the receive PCM multi-frame mode. 0 = Normal, source from DSL sync detector (default) 1 = System Bus (common sync), source from TPMFSYNC
	3:2	PCM DPLL Ref Select	Determines the source of the PCM DPLL reference. 00 = Internal DSL sync detector (default) 01 = TH_REF. Used by X.2 in synchronous or transparent mode. 10 = DSLSYNCI input pin. Used in multi-pair applications. 11 = DSL data rate clock. Used by X.3 in synchronous or transparent mode.

	1	DPLL Mode	Configure the PCM DPLL mode. The DPLL clock can operate in either closed-loop or open-loop mode. The PCM Rx clock is typically sourced from the DPLL clock. Reconfiguring the DPLL mode will cause the DPLL state machine to re-run. 0 = DPLL operates in a closed loop to recover the PCM receive clock from the master DSL receive channel. During startup, the DPLL is switched to open-loop mode to provide a stable PCM RCLK. Once startup is completed, the DPLL is switched back to closed-loop mode. (HTU-R default) 1 = DPLL always operates in open-loop mode. The DPLL clock provides a fixed frequency. (HTU-C default)
	0	PCM Float	Configure the PCM Frame Format. When running in framed format, the sync bit indicates bit 0 of time slot 0 of frame 0. Unframed format allows unframed or asynchronous payload mapping of PCM frames into DSL frames. 0 = Sync aligned with bit 0 of time slot 0 1 = Asynchronous sync versus data alignment (default)

15.7.4 PCM Multiframe Length

PCM Multiframe Length

Program the PCM multiframe length. The PCM multiframe is used to indicate bit 0 of time slot 0.

The internal software maintains two counters to produce the PCM multiframe length—MF_LEN and REF_LEN.

MF_LEN—multiframe length, this counter specifies the number of frames per multiframe.

REF_LEN—DSL frame reference length, this counter determines the number of MF_LEN frames per the 6 ms DSL frame. The formula to determine REF_LEN is:

REF_LEN = 48 / MF_LEN; where the 48 is derived from the 48 payload blocks per 6 ms DSL frame.

Because both of these counters must be of integer multiples, the PCM multiframe length can only be programmed to the following values (the value in the parenthesis is the appropriate API setting): 125 µs (0), 250 µs (1), 375 µs (2), 500 µs (3), 750 µs (5), 1.0 ms (7), 1.5 ms (11), 2 ms (15), 3 ms (23), and 6 ms (47)

NOTE: The PCM multiframe length API command is only applicable in frame structures that operate on 6 ms boundaries, i.e., G.shdsl, HDSL2 OPTIS, and HDSL1.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_PCM_MF_LEN	0x05	Control	2	None

Incoming Parameters

Byte	Content	Description
1	Transmit Multiframe Length	A value of 0 implies 1 frame. One frame is 125 µs. X = 47 (1 to 48 frames or 125 µs–6 ms). Default is 6 ms.
2	Receive Multiframe Length	A value of 0 implies 1 frame. One frame is 125 µs. X = 47 (1 to 48 frames or 125 µs–6 ms). Default is 6 ms.

15.7.5 Transmit PCM Frame Offset

Transmit PCM Frame Offset				
This command sets the transmit PCM frame offset relative to the TPMFSYNC signal. This allows the system to define which bit the PCM MFSYNC signal marks within the PCM multi-frame.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_TP_FRM_OFST	0x2E	Control	2	None

Incoming Parameters

Byte	Content	Description
1-2	Offset	16-bit unsigned value specifying the frame offset. Low byte programmed first.

NOTE: 16-bit value = (high byte << 8) + (low byte).

15.7.6 Receive PCM Frame Offset

Receive PCM Frame Offset				
This command sets the receive PCM frame offset relative to the RPMFSYNC signal. This allows the system to define which bit the PCM MFSYNC signal marks within the PCM multi-frame.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_RP_FRM_OFST	0x2F	Control	2	None

Incoming Parameters

Byte	Content	Description
1-2	Offset	16-bit unsigned value specifying the frame offset. Low byte programmed first.

NOTE: 16-bit value = (high byte << 8) + (low byte).

15.7.7 Multipair Configuration

Multipair Configuration				
This command configures the ZipWirePlus for multipair applications.				
Opcode C Constant	**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_MULTI_PAIR_CONFIG	0x19	Control	1	None

Incoming Parameters			
Byte	**Content**		**Description**
1	Multipair Config		See bit-field description below.
	Bit	**Content**	**Description**
	7:5	Reserved	Set the bits to '0' for future compatibility.
	4	Master/Slave Loop Select	Set the ZipWirePlus device to be the master loop or slave loop. 0 = Master Loop (default) 1 = Slave Loop, multiple devices within a group can be configured as a slave.
	3:2	Reserved	Set the bits to '0' for future compatibility.
	1:0	Multipair Mode	Set the multipair configuration mode 00 = Normal, single pair configuration (default) 01 = PCM Bused Mode, see ZipWirePlus data sheet. 10 = PCM Cascade Mode, see ZipWirePlus data sheet. 11 = ATM Cascade Mode, see ZipWirePlus data sheet.

15.7.8 DSL Framer Transmit PCM Path Reset

DSL Framer Transmit PCM Path Reset				
Writing a 1 to a single bit in this command resets the operation of the corresponding module in the transmit PCM path. After the reset operation is completed, the device clears the bit, the read-back reads 0.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_FR_TX_RESET	0x4E	Control	1	None

Incoming Parameters				
Byte	**Content**	**Description**		
1	Transmit Mask	See the Transmit Mask Bit Definitions table below.		
		Transmit Mask Bit Definition		
		Bit	**Module**	**Description**
		7:6	Reserved	Reserved

		Bit	Module	Description
		5	TP_SD_RST	Writing 1 to this bit initializes the transmit BSP Sync Detector process.
		4	TX_FIFO_RST	Writing 1 to this bit initializes the transmit FIFO pointers.
		3	TX_WL_RST	Writing 1 to this bit initializes the transmit water level depth.
		2	TX_WL_START	Writing 1 to this bit starts the process of transmit water level measurement.
		1	TP_BER RST	Writing 1 to this bit resets the transmit PCM BER Meter process (from PCM).
		0	TP_PRBS_RST	Writing 1 to this bit resets the transmit PCM PRBS process (towards DSL).

15.7.9 DSL Framer Receive PCM Path Reset

DSL Framer Receive Path PCM Reset				
Writing a 1 to a single bit in this command resets the operation of the corresponding module in the receive PCM path. After the reset operation is completed, the device clears the bit, the read-back reads 0.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_FR_RX_RESET	0x4F	Control	1	None

Incoming Parameters				
Byte	**Content**	**Description**		
1	Receive Mask	See the Receive *Mask Bit Definitions* table below.		
		Receive Mask Bit Definition		
		Bit	**Module**	**Description**
		7	HSYNC_RST	Writing 1 to this bit resets the DSL SYNC Detector state machine.
		6	DPLL_RST	Writing 1 to this bit resets the DPLL state machine.
		5	RP_SD_RST	Writing 1 to this bit initializes the receive BSP Sync Detector process.
		4	RX_FIFO_RST	Writing 1 to this bit initializes the receive FIFO pointers.
		3	RX_WL_RST	Writing 1 to this bit initializes the receive water level depth.
		2	RX_WL_START	Writing 1 to this bit starts the process of receive water level measurement.

		1	RP_BER RST	Writing 1 to this bit resets the receive PCM BER Meter process (from DSL).
		0	RP_PRBS_RST	Writing 1 to this bit resets the receive PCM PRBS process (towards PCM).

15.7.10 DSL Framer PCM DPLL Clock Generator

DSL Framer PCM DPLL Clock Generator				
This command configures the PCM DPLL for the desired frequency. The DPLL clock can be configured to a 1Hz resolution. The clock is derived from the XTAL but it is not guaranteed to have the same frequency offset.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_DPLL_CLOCK_GEN	0x58	Control	4	None

Incoming Parameters		
Byte	**Content**	**Description**
1-4	Frequency	Four-byte value specifying the desired DPLL output frequency. The low byte is programmed first.
		NOTE: 32-bit value = (byte 4 << 24) + (byte 3 << 16) + (byte 2 << 8) + (byte 1)

15.7.11 DPLL Reference Configuration

DPLL Reference Configure				
This command configures the source of the DPLL 6ms input when not from the DSL Sync detector. The reference can either be the HBCLK or the PEXTCLK pin. HBCLK is the DSP(bitpump) clock. The clock frequency range is from 8 kHz to 18.432 MHz in 8 kHz resolution. Recommended settings: When running ISDN clocking mode, input the network timing reference into the PEXTCLK. The DPLL output clock will be referenced to the PEXTCLK offset. The _DSL_FR_PCM_CONFIG API has to be set to use _DPLL_REF_DSL_DATA_RATE . This feature will be enabled for both PCM and NB. This defaults to use HBCLK and the DPLL reference comes for the DSL sync detector (Non-ISDN clocking mode).				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DPLL_REF_SOURCE	0x61	Control	3	None

Incoming Parameters			
Byte	**Content**		**Description**
1	Clock Config		See the bit-field description below.
	Bit	**Content**	**Description**

	7:2	Reserved	Reserved
	1:0	Ref Clock Source	Select the clock mode. 00 = Reserved 01 = PEXTCLK_pin 10 = HBCLK (default) 11 = SCLK
2-3	Reference Clock Divider		16-bit unsigned value specifying the network timing reference frequency. The low byte is sent first. This is a 1-based parameter where a 1 means 1 (8kHz). Default is 1 (8 kHz). The formula for programming the API command is: Value= $\frac{REF\_CLK\_SRC\_FREQUENCY}{8k}$
NOTE: 16-bit value = (high byte << 8) + (low byte).			

15.7.12 DSL Framer PCM Water Level

DSL Framer PCM Water Level				
This command configures the PCM water level mode and values. The new water level values will be used when the appropriate WL_RST is performed; either from a new startup or _DSL_FR_SET_STATE_MACHINE (0x4A).				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_PCM_WATER_LEVEL	0x2C	Control	5	None

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.
2-3	Tx. PCM Water Level	9-bit (2-byte) value specifying the initial transmit PCM water level depth (in TPCLK writes). The low byte is programmed first. A value of 0 implies a 1 clock cycle delay. Default value is 0xF0.
4-5	Rx. PCM Water Level	9-bit (2-byte) value specifying the initial receive PCM water level depth (in DSL data rate clock writes). The low byte is programmed first. A value of 0 implies a 1 clock cycle delay. Default value is 0xF0.

15.8 PCM Mapper Configuration and Status API Commands

15.8.1 Transmit PCM Mapper Value

Transmit PCM Mapper Value
The Transmit PCM Mapper Value command allows the user to control the data going into the Transmit PCM FIFO. The Transmit PCM FIFO Mapper Table is implemented with an 8 x 128 RAM that can accommodate up to 128 different table entries. Each table entry can then process from 1 to 8 time slots, thus providing up to 1024 (8 x 128) time slots per Transmit PCM frame. The 128 table entries are indexed from 0 to 127 and are stored in data RAM. The Transmit PCM Mapper Write command is used to write the table from the data RAM to the device.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_TP_MAPPER_VALUE	0x30	Control	65	None

Incoming Parameters

Byte	Content		Description
1	Starting Address		1-byte value indicating the table entry starting address to write (zero based). The Starting Address can be between 0–127 corresponding to a total of 128 bytes for the TP_MAPPER table entry.
2–65	Table Entry		See bit-field description below. The default is based on the system configuration.
	Bit	**Content**	**Description**
	7:5	Number of Time Slots (NTS)	Indicates the number of time slots to be controlled in each table line (000 stands for 1 and 111 stands for 8).
	4	Reserved	Set the bit to ‘0’ for future compatibility.
	3	BER Enable	Enables the BER meter (from PCM) for the specified time slots: 0 = Discard 1 = Enable BER meter
	2:0	Data Source	Specifies the Transmit PCM data source: 000 = Disregards data 001 = DATA from serial input TPDAT 010 = PRBS generator (towards DSL) 011 = Assert TPINSEN pin and inserts data from TPINSDAT pin 100 = Previous time slot 101–111 = Not used

15.8.2 Transmit PCM Mapper Write

Transmit PCM Mapper Write				
Writes the Transmit PCM Mapper table to the device. This should only be called after the Transmit PCM Mapper table has been completely filled in. The appropriate resets are issued.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_TP_MAPPER_WRITE	0x31	Control	1	None
Incoming Parameters				
Byte	**Content**	**Description**		
1	Reserved	Set to 0x00 for future compatibility.		

15.8.3 Transmit PCM Mapper Read

Transmit PCM Mapper Read				
This command reads back the Transmit PCM Mapper setting (this reads back the local variable, not the hardware registers)				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_TP_MAPPER_READ	0xA3	Status	2	L
Incoming Parameters				
Byte	**Content**	**Description**		
1	Starting Address	1-byte value indicating the table entry starting address to read (zero based). The Starting Address can be between 0–127 corresponding to a total of 128 bytes for the TP_MAPPER table entry.		
2	Number of table entry to read (L)	Number of table entries to read (zero based). To read back (L) bytes the table entry parameter should be set to (L-1).		
Outgoing Parameters				
Byte	**Content**	**Description**		
1-L	Table Entry		See bit-field description below. The default is based on the system configuration.	
	Bit	**Content**	**Description**	
	7:5	Number of Time Slots (NTS)	Indicates the number of time slots to be controlled in each table line (000 stands for 1 and 111 stands for 8).	
	4	Reserved	Set the bit to ‘0’ for future compatibility.	

	Bit	Content	Description
	3	BER Enable	Enables the BER meter (from PCM) for the specified time slots: 0 = Discard 1 = Enable BER meter
	2:0	Data Source	Specifies the Transmit PCM data source: 000 = Disregards data 001 = DATA from serial input TPDAT 010 = PRBS generator (towards DSL) 011 = Assert TPINSEN pin and inserts data from TPINSDAT pin 100 = Previous time slot 101–111 = Not used

15.8.4 Receive PCM Mapper Value

Receive PCM Mapper Value

The Receive PCM Mapper Value command allows the user to control the data going out to RPDAT. The Receive PCM FIFO mapper table is implemented with an 8 x 128 RAM that can accommodate up to 128 different table entries. Each table entry can then process from 1 to 8 time slots, thus providing up to 1024 (8 x 128) time slots per Receive PCM frame. The 128 table entries are indexed from 0 to 127 and are stored in data RAM. The Receive PCM Mapper Write command is used to write the table from the data RAM to the device.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_RP_MAPPER_VALUE	0x32	Control	65	None

Incoming Parameters

Byte	Bit	Content	Description
1		Starting Address	1-byte value indicating the table entry starting address to write (zero based). The Starting Address can be between 0–127 corresponding to a total of 128 bytes for the RP_MAPPER table entry.
2–65		Table Entry	See bit-field description below. The default is based on the system configuration.
	Bit	**Content**	**Description**
	7:5	Number of Time Slots (NTS)	Indicates the Number of Time Slots (NTS) to be controlled in each table line (000 stands for 1 and 111 stands for 8).
	4	DROP	When enabled, asserts RPDROP output to mark specific time slots in RPDAT. When disabled and *RPDAT_MODE* = 1, RPDAT is three-stated. 0 = Disable 1 = Enable
	3	BER Enable	Enables the BER meter (from DSL) for the specified time slots. 0 = Discard 1 = Enable BER Meter

	2:0	Data Source	Specifies the Receive PCM data source for each time slot. 000 = RX PCM FIFO 001 = PRBS generator (towards PCM) 010 = DATA BANK Register 1 (DBANK_1) 011 = DATA BANK Register 2 (DBANK_2) 100 = DATA BANK Register 3 (DBANK_3) 101 = Data from RPEXTDAT input (used in multi-pair configuration) 110 = Not used 111 = Not used

15.8.5 Receive PCM Mapper Write

Receive PCM Mapper Write				
Writes the Receive PCM Mapper table to the device. This should only be called after the Receive PCM Mapper table has been completely filled in. The appropriate resets are issued.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_RP_MAPPER_WRITE	0x33	Control	1	None
Incoming Parameters				

Byte	Content	Description
1	Reserved	Set to 0x00 for future compatibility.

15.8.6 Receive PCM Mapper Read

Receive PCM Mapper Read				
This command reads back the receive PCM Mapper setting (this reads back the local variable, not the hardware registers)				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_RP_MAPPER_READ	0xA4	Status	2	L
Incoming Parameters				

Byte	Content	Description
1	Starting Address	1-byte value indicating the table entry starting address to read (zero based). The Starting Address can be between 0–127 corresponding to a total of 128 bytes for the RP_MAPPER table entry.
2	Number of table entry to read (L)	Number of table entries to read (zero based). To read back (L) bytes the table entry parameter should be set to (L-1).

Outgoing Parameters			
Byte	**Content**		**Description**
1-L	Table Entry		See bit-field description below. The default is based on the system configuration.
	Bit	**Content**	**Description**
	7:5	Number of Time Slots (NTS)	Indicates the Number of Time Slots (NTS) to be controlled in each table line (000 stands for 1 and 111 stands for 8).
	4	DROP	When enabled, asserts RPDROP output to mark specific time slots in RPDAT. When disabled and *RPDAT_MODE* = 1, RPDAT is three-stated. 0 = Disable 1 = Enable
	3	BER Enable	Enables the BER meter (from DSL) for the specified time slots. 0 = Discard 1 = Enable BER Meter
	2:0	Data Source	Specifies the Receive PCM data source for each time slot. 000 = RX PCM FIFO 001 = PRBS generator (towards PCM) 010 = DATA BANK Register 1 (DBANK_1) 011 = DATA BANK Register 2 (DBANK_2) 100 = DATA BANK Register 3 (DBANK_3) 101 = Data from RPEXTDAT input (used in multi-pair configuration) 110 = Not used 111 = Not used

15.9 Narrowband Interface Configuration API Commands

15.9.1 Narrowband Multirate Configuration

Narrowband Multirate Configuration				
This command configures the Narrowband multirate parameters. These parameters are used to determine the Narrowband bus speed and mapping into the DSL domain. When the 'Maintain DSL Link' bit (byte #7) is cleared (0), this command causes the ZipWirePlus device to deactivate any training or test modes and transition to the Configure ZipWirePlus state (see Figure 4-2). The device is configured based on the settings. The ASM is disabled. When the 'Maintain DSL Link' bit (byte #7) is set (1), this command will reconfigure only the NB interface. The DSL domain is not reconfigured. This is useful when the application needs to adjust the NB interface after the DSL link is established. The pre-activation (G.hs/Auto Baud) phase is used to establish the DSL data rate while the EOC channel is used to establish the application interface (PCM, NB, ATM) data rates.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_NB_MULTI_RATE_CONFIG	0x1A	Control	6	None

Incoming Parameters			
Byte	Content		Description
1-2	Number of NB Time Slots		Total number of Narrowband time slots available. A value of 1 implies 1 time slot. The low byte is programmed first. X = 1…72
3	Number of Occupied NB Time Slots		Total number of occupied time slots available. A value of 1 implies 1 time slot. This parameter should be set such that the number of Occupied PCM time slots plus the number of Occupied NB time slots is equal to the total number of DSL time slots.
4	Start NB Time Slot Location		Indicates the location of the first time slot to extract from the Narrowband bus. The first Narrowband time slot is numbered 1.
5	Multipair Config		See bit-field description below.
	Bit	Content	Description
	7:5	Reserved	Set the bits to '0' for future compatibility.
	4	Master/Slave Loop Select	Set the ZipWirePlus device to be the master loop or slave loop. 0 = Master Loop (default) 1 = Slave Loop, multiple devices within a group can be configured as a slave.
	3-2	Reserved	Set the bits to '0' for future compatibility.
	1:0	Multipair Mode	Set the multipair configuration mode 00 = Normal, single pair configuration (default) 01 = Narrowband PCM Bused Mode 10 = Narrowband PCM Cascade Mode 11 = Reserved
6	Mapping Format		See bit-field description below.
	Bit	Content	Description
	7	Maintain DSL Link	0 = No, configures complete device. 1 = Yes, don't drop DSL link. Only reconfigure NB interface. This option should only be used when the DSL link is up.
	6:3	Reserved	Reserved
	2	DSL Mapping Order	Determines the order of PCM and NB data within a DSL frame. 0 =PCM data appears before NB data. (default) 1= PCM data appears after NB data.
	1:0	Mapping Mode	Determines either Block mapping or Interleave mapping. 0x00 =Block Mapping (default) 0x01 = Interleave Mapping. 0x10–0x11= Reserved

15.9.2 Narrowband Configuration

Narrowband Configuration						
This command configures the Narrowband interface.						
Opcode C Constant			**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_NB_CONFIG			0x28	Control	4	None
Incoming Parameters						
Byte	**Content**		**Description**			
1	NB Config		Configure the Narrowband block. See bit-field description below.			
	Bit	**Content**	**Description**			
	7	TNBMFSYNC Select	Selects source of the TNBMFSYNC 0 = Use TNBMFSYNC pin, required when TNBDAT needs be aligned. 1= Use TPMFSYNC pin, internally routed (default)			
	6	NBEXTCLK Select	Selects source of Narrow Band internal PEXTCLK source. Required for Narrow Band multi-pair applications. 0 = Use PEXTCLK pin (default) 1= Use ATM_TX_CLK pin; required when need both PCM & NB multi-pair. Not applicable when using Utopia interface.			
	5	TNBMFSYNC Direction	Transmit Narrowband multi-frame sync (TNBMFSYNC) pin direction. 0 = Input (default) 1= Output			
	4	RNBMFSYNC Mode	Configures the receive Narrowband multiframe mode. 0 = Normal, source from DSL sync detector (default) 1 = System Bus (common sync), source from TNBMFSYNC			
	3	NB DPLL Ref Select	Determines the source of the Narrowband DPLL reference. 0 = Internal DSL sync detector (default) 1 = DSYSYNCI input pin. Used in multi-pair applications.			
	2	HTUR Skew Disable	0 = enable HTU-R Skew for NB Closed loop (default) 1 = disable HTU-R Skew for NB Open Loop. (Used if NB HTU-R is set to open loop).			
	1	NB DPLL Mode	Configure the Narrowband DPLL Mode. The DPLL clock can operate in either closed-loop or open-loop mode. The Narrowband Rx Clock is typically sourced from the DPLL clock. Reconfiguring the DPLL mode will cause the DPLL state machine to re-run. 0 = DPLL operates in closed loop to recover the Narrowband receive clock from the master HDSL receive channel. During startup, the DPLL is switched to open-loop mode to provide a stable Narrowband RCLK. Once startup is completed, the DPLL is switched back to closed-loop mode. (HTU-R default) 1 = DPLL always operates in open-loop mode. The DPLL clock provides a fixed frequency. (HTU-C default)			

	0	NB Float	Configure the Narrowband frame format alignment. When running in framed format, the sync bit is used to indicate bit 0 of time slot 0 of frame 0. Unframed format allows unframed or asynchronous payload mapping of Narrowband frames into HDSL frames. 0 = Sync aligned with bit 0 of time slot 0. (default) 1 = Asynchronous sync versus data alignment.
2	NB Clock Config		Configure the Narrowband transmit and receive clock sources. See bit-field description below.
	Bit	**Content**	**Description**
	7:6	RNB_CLK_TRAIN	Receive Narrowband clock source during training. The RNBCLK is set to this option when the startup is initiated and the RNBCLK will be put back to the RNB_CLK_SOURCE after the startup is complete (for both successful and failed startups). 00 = Normal, no change. Use RNB_CLK_SOURCE during training (default) 01 = TNBCLK input pin 10 = PEXTCLK input 11 = Reserved
	5	Reserved	Set the bit to '0' for future compatibility.
	4	TNB_CLK_POL	Transmit Narrowband clock polarity. 0 = Normal clock selected by TNB_CLK_SOURCE (rising edge outputs, falling edge inputs). (default) 1 = Inverted clock selected by TNB_CLK_SOURCE (falling edge outputs, rising edge inputs).
	3:2	RNB_CLK_SOURCE	Receive Narrowband clock source. 00 = TNBCLK input pin. 01 = PEXTCLK input. 10 = Narrowband DPLL recovery clock. (default) 11 = PCM DPLL recovery clock.
	1:0	TNB_CLK_SOURCE	Transmit Narrowband clock source. 00 = TNBCLK input pin. (default) 01 = PEXTCLK input. 10 = Narrowband DPLL recovery clock. 11 = PCM DPLL recovery clock.
3	NB Transmit Multi-Frame Length ***(1)***		1–byte specifying the transmit multi-frame length. Valid value is: 0 (125 us), 1 (250 us), 2 (375 us), 3 (500 us), 5 (750 us), 7 (1 ms), 11 (1.5 ms), 15 (2 ms), 23 (3 ms), and 47 (6 ms). Default value is 47 (6 ms).
4	NB Receive Multi-Frame Length ***(1)***		1–byte specifying the receive multi-frame length. Valid value are: 0 (125 us), 1 (250 us), 2 (375 us), 3 (500 us), 5 (750 us), 7 (1 ms), 11 (1.5 ms), 15 (2 ms), 23 (3 ms), and 47 (6 ms). Default value is 47 (6 ms).

NOTE: (1) See PCM MF Length API command for complete details. See Section 15.7.4.

15.9.3 Transmit Narrowband Frame Offset

Transmit Narrowband Frame Offset					
This command sets the transmit Narrowband frame offset relative to the TNBSYNC signal. This allows the system to define which bit the Narrowband MFSYNC signal marks within the Narrowband multi-frame.					
C Constant		**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_TNB_FRM_OFST		0x29	Control	2	None
Incoming Parameters					
Byte	**Content**	**Description**			
1-2	Offset	16-bit unsigned value specifying the frame offset. Low byte programmed first.			
NOTE: 16-bit value = (high byte << 8) + (low byte).					

15.9.4 Receive Narrowband Frame Offset

Receive Narrowband Frame Offset					
This command sets the receive Narrowband frame offset relative to the RNBSYNC signal. This allows the system to define which bit the Narrowband MFSYNC signal marks within the Narrowband multi-frame.					
C Constant		**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_RNB_FRM_OFST		0x2A	Control	2	None
Incoming Parameters					
Byte	**Content**	**Description**			
1-2	Offset	16-bit unsigned value specifying the frame offset. Low byte programmed first.			
NOTE: 16-bit value = (high byte << 8) + (low byte).					

15.9.5 DSL Framer Transmit Narrowband Path Reset

DSL Framer Transmit Narrowband Path Reset				
Writing a 1 to a single bit in this command resets the operation of the corresponding module in the transmit Narrowband path. After the reset operation is completed, the device clears the bit, the read-back reads 0.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_FR_TNB_RESET	0x4B	Control	1	None

Incoming Parameters				
Byte	**Content**	**Description**		
1	Transmit Mask	See the Transmit Mask Bit Definitions table below.		
		Transmit Mask Bit Definition		
		Bit	**Module**	**Description**
		7:6	Reserved	Reserved
		5	TNB_SD_RST	Writing 1 to this bit initializes the transmit BSP Sync Detector process.
		4	TNB_FIFO_RST	Writing 1 to this bit initializes the transmit FIFO pointers.
		3	TNB_WL_RST	Writing 1 to this bit initializes the transmit water level depth.
		2	TNB_WL_START	Writing 1 to this bit starts the process of transmit water level measurement.
		1	TNB_BER RST	Writing 1 to this bit resets the transmit NB BER Meter process (from NB).
		0	TNB_PRBS_RST	Writing 1 to this bit resets the transmit NB PRBS process (towards DSL).

15.9.6 DSL Framer Receive Narrowband Path Reset

DSL Framer Receive Narrowband Path Reset				
Writing a 1 to a single bit in this command resets the operation of the corresponding module in the receive Narrowband path. After the reset operation is completed, the device clears the bit, the read-back reads 0.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_FR_RNB_RESET	0x4C	Control	1	None

Incoming Parameters				
Byte	**Content**	**Description**		
1	Receive Mask	See the Receive *Mask Bit Definitions* table below.		
		Receive Mask Bit Definition		
		Bit	**Module**	**Description**
		7	Reserved	Reserved
		6	NBDPLL_RST	Writing 1 to this bit resets the NB DPLL state machine.
		5	RNB_SD_RST	Writing 1 to this bit initializes the receive BSP Sync Detector process.

		4	RNB_FIFO_RST	Writing 1 to this bit initializes the receive FIFO pointers.
		3	RNB_WL_RST	Writing 1 to this bit initializes the receive water level depth.
		2	RNB_WL_START	Writing 1 to this bit starts the process of receive water level measurement.
		1	RNB_BER RST	Writing 1 to this bit resets the receive NB BER Meter process (from DSL).
		0	RNB_PRBS_RST	Writing 1 to this bit resets the receive NB PRBS process (towards NB).

15.9.7 DSL Framer Narrowband DPLL Clock Generator

DSL Framer Narrowband DPLL Clock Generator				
This command configures the Narrowband DPLL for the desired frequency. The NB DPLL clock can be configured to a 1Hz resolution. The clock is derived from the XTAL but it is not guaranteed to have the same frequency offset.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_NB_DPLL_CLOCK_GEN	0x59	Control	4	None
Incoming Parameters				

Byte	Content	Description
1-4	Frequency	Four-byte value specifying the desired DPLL output frequency. The low byte is programmed first.
		NOTE: 32-bit value = (byte 4 << 24) + (byte 3 << 16) + (byte 2 << 8) + (byte 1)

15.9.8 DSL Framer Narrow Band Water Level

DSL Framer Narrow Band Water Level				
This command configures the Narrow Band water level mode and values. The new water level values will be used when the appropriate WL_RST is performed; either from a new startup or _DSL_FR_SET_STATE_MACHINE (0x4A).				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_NB_WATER_LEVEL	0x2D	Control	5	None
Incoming Parameters				

Byte	Content	Description
1	Reserved	Set to 0x00 for future compatibility.
2-3	Transmit NB Water Level	9-bit (2-byte) value specifying the initial transmit NB water level depth (in TNBCLK writes). The low byte is programmed first. A value of 0 implies a 1 clock cycle delay. Default value is 0x20.

4-5	Receive NB Water Level	9-bit (2-byte) value specifying the initial receive NB water level depth (in DSL data rate clock writes). The low byte is programmed first. A value of 0 implies a 1 clock cycle delay. Default value is 0x7F.

15.10 Narrowband Mapper Configuration and Status API Commands

15.10.1 Transmit Narrowband Mapper Value

Transmit Narrowband Mapper Value				
The Transmit Narrowband Mapper Value command allows the user to control the data going into the Transmit Narrowband FIFO. The Transmit Narrowband FIFO Mapper Table is implemented with an 8 x 128 RAM that can accommodate up to 128 different table entries. Each table entry can then process from 1 to 8 time slots, thus providing up to 1024 (8 x 128) time slots per Transmit Narrowband frame. The 128 table entries are indexed from 0 to 127 and are stored in data RAM. The Transmit Narrowband Mapper Write command is used to write the table from the data RAM to the device.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_TNB_MAPPER_VALUE	0x38	Control	65	None

Incoming Parameters			
Byte	**Content**		**Description**
1	Starting Address		1-byte value indicating the starting address of the table entry to write (zero based). The Starting Address can be between 0–127 corresponding to a total of 128 bytes for the TNB_MAPPER table entry.
2–65	Table Entry		See bit-field description below. The default is based on the system configuration.
	Bit	**Content**	**Description**
	7:5	Number of Time Slots (NTS)	Indicates the number of time slots to be controlled in each table line (000 stands for 1 and 111 stands for 8).
	4	Reserved	Set the bit to '0' for future compatibility.
	3	BER Enable	Enables the BER meter (from Narrowband) for the specified time slots: 0 = Discard 1 = Enable BER meter
	2:0	Data Source	Specifies the Transmit Narrowband data source: 000 = Disregards data 001 = DATA from serial input TNBDAT 010 = PRBS generator (towards DSL) 011 = Reserved 100 = Previous time slot 101–111 = Not used

15.10.2 Transmit Narrowband Mapper Write

Transmit Narrowband Mapper Write					
Writes the Transmit Narrowband Mapper table to the device. This should only be called after the Transmit Narrowband Mapper table has been completely filled in. The appropriate resets are issued.					
C Constant		Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_TNB_MAPPER_WRITE		0x39	Control	1	None
Incoming Parameters					
Byte	Content	Description			
1	Reserved	Set to 0x00 for future compatibility.			

15.10.3 Transmit Narrowband Mapper Read

Narrowband Transmit PCM Mapper Read						
This command reads back the Narrowband Transmit PCM Mapper setting (this reads back the local variable, not the hardware registers).						
C Constant			Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_TNB_MAPPER_READ			0xA7	Status	2	L
Incoming Parameters						
Byte	Content		Description			
1	Starting Address		1-byte value indicating the starting address of the table entry to read (zero based). The Starting Address can be between 0–127 corresponding to a total of 128 bytes for the TNB_MAPPER table entry.			
2	Number of table entry to read (L)		Number of table entries to read (zero based). To read back (L) bytes the table entry parameter should be set to (L-1).			
Outgoing Parameters						
Byte	Content		Description			
1-L	Table Entry		See bit-field description below. The default is based on the system configuration.			
	Bit	Content	Description			
	7:5	Number of Time Slots (NTS)	Indicates the number of time slots to be controlled in each table line (000 stands for 1 and 111 stands for 8).			
	4	Reserved	Set the bit to '0' for future compatibility.			
	3	BER Enable	Enables the BER meter (from Narrowband) for the specified time slots: 0 = Discard 1 = Enable BER meter			

	Bit	Content	Description
	2:0	Data Source	Specifies the Transmit Narrowband data source: 000 = Disregards data 001 = DATA from serial input TNBDAT 010 = PRBS generator (towards DSL) 011 = Reserved 100 = Previous time slot 101–111 = Not used

15.10.4 Receive Narrowband Mapper Value

Receive Narrowband Mapper Value				
The Receive Narrowband Mapper Value command allows the user to control the data going into the Receive Narrowband FIFO. The Receive Narrowband FIFO mapper table is implemented with an 8 x 128 RAM that can accommodate up to 128 different table entries. Each table entry can then process from 1 to 8 time slots, thus providing up to 1024 (8 x 128) time slots per Receive Narrowband frame. The 128 table entries are indexed from 0 to 127 and are stored in data RAM. The Receive Narrowband Mapper Write command is used to write the table from the data RAM to the device.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_RNB_MAPPER_VALUE	0x3A	Control	65	None

Incoming Parameters

Byte	Content		Description
1	Starting Address		1-byte value indicating the table entry starting address to write (zero based). The Starting Address can be between 0–127 corresponding to a total of 128 bytes for the RNB_MAPPER table entry.
2–65	Table Entry		See bit-field description below. The default is based on the system configuration.
	Bit	**Content**	**Description**
	7:5	Number of Time Slots (NTS)	Indicates the Number of Time Slots (NTS) to be controlled in each table line (000 stands for 1 and 111 stands for 8).
	4	DROP	When enabled, asserts RNBDROP output to mark specific time slots in RNBDAT. When disabled and *RNBDAT_MODE* = 1, RPDAT is three-stated. 0 = Disable 1 = Enable
	3	BER Enable	Enables the BER meter (from DSL) for the specified time slots. 0 = Discard 1 = Enable BER Meter

	2:0	Data Source	Specifies the Receive Narrowband data source for each time slot. 000 = Receive Narrowband FIFO 001 = PRBS generator (towards Narrowband) 010 = DATA BANK Register 1 (DBANK_1) 011 = DATA BANK Register 2 (DBANK_2) 100 = DATA BANK Register 3 (DBANK_3) 101 = Data from TPINSDAT input pin used for multipair configuration 110 = Not used 111 = Not used

15.10.5 Receive Narrowband Mapper Write

Receive Narrowband Mapper Write				
Writes the Receive Narrowband Mapper table to the device. This should only be called after the Receive Narrowband Mapper table has been completely filled in. The appropriate resets are issued.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_RNB_MAPPER_WRITE	0x3B	Control	1	None

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.

15.10.6 Receive Narrowband Mapper Read

Narrowband Receive PCM Mapper Read				
This command reads back the Narrowband Receive PCM Mapper setting (this reads back the local variable, not the hardware registers).				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_RNB_MAPPER_READ	0xA8	Status	2	L

Incoming Parameters		
Byte	**Content**	**Description**
1	Starting Address	1-byte value indicating the table entry starting address to read (zero based). The Starting Address can be between 0–127 corresponding to a total of 128 bytes for the RNB_MAPPER table entry.
2	Number of table entry to read (L)	Number of table entries to read (zero based). To read back (L) bytes the table entry parameter should be set to (L-1).

Outgoing Parameters			
Byte	**Content**		**Description**
1-L	Table Entry		See bit-field description below. The default is based on the system configuration.
	Bit	**Content**	**Description**
	7:5	Number of Time Slots (NTS)	Indicates the Number of Time Slots (NTS) to be controlled in each table line (000 stands for 1 and 111 stands for 8).
	4	DROP	When enabled, asserts RPDROP output to mark specific time slots in RPDAT. When disabled and *RPDAT_MODE* = 1, RPDAT is three-stated. 0 = Disable 1 = Enable
	3	BER Enable	Enables the BER meter (from DSL) for the specified time slots. 0 = Discard 1 = Enable BER Meter
	2:0	Data Source	Specifies the Receive Narrowband data source for each time slot. 000 = Receive Narrowband FIFO 001 = PRBS generator (towards Narrowband) 010 = DATA BANK Register 1 (DBANK_1) 011 = DATA BANK Register 2 (DBANK_2) 100 = DATA BANK Register 3 (DBANK_3) 101 = Data from TPINSDAT input pin used for multipair configuration 110 = Not used 111 = Not used

15.11 DSL Interface Configuration API Commands

15.11.1 Set DSL Framer State Machine

Set DSL Framer State Machine				
This command sets the DSL Framer state machine. This command is used when the system needs to reconfigure the PCM interface or DSL Framer mappers without disrupting the DSL link. ***NOTE:*** This command should only be issued when the DSL link is in normal operation or in a loopback that includes the DSL Framer.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_FR_SET_STATE_MACHINE	0x4A	Control	1	None

Incoming Parameters		
Byte	**Content**	**Description**
1	Framer State	0 – disable framer interrupts 1 – reenable framer (enable interrupts and re-start state machines)

15.11.2 DSL Framer Write Indicator Bits

DSL Framer Write Indicator Bits

This command determines the indicator bits to be transmitted out the DSL frame. The bits will be set on the next frame boundary (typically 6ms) and will continue to be sent on all subsequent frames until this command is issued again.

This command provides a generic mechanism for setting the indicator bits. The exact bit definitions vary from frame structure to frame structure (G.shdsl, HDSL2, HDSL1, IDSL).

Note 1: For IDSL, febe bit is updated by the ZipWirePlus device. The user should set this bit to 1 in this API.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_WRITE_IND_BITS	0x60	Control	2	None

Incoming Parameters			
Byte	**Content**		**Description**
G.shdsl			
1	Low Byte Indicator Bits		Default is all 1's (OxFF)
	Bit	**Content**	**Description**
	7-4	Reserved	Set the bits to '0' for future compatibility.
	3	segd	Segment Defect
	2	ps	HTU-R Power Status (Dying Gasp)
	1	sega	Segment Anomaly
	0	losd	Loss of Signal
2	High Byte Indicator Bits		Default is all 1's (OxFF)
	Bit	**Content**	**Description**
	7-0	Reserved	Set the bits to '0' for future compatibility.
HDSL2			
1	Low Byte Indicator Bits		Default is all 1's (OxFF)
	Bit	**Content**	**Description**
	7-4	Reserved	Set the bits to '0' for future compatibility.
	3	segd	Segment Defect

	2	sega	Segment Anomaly
	1	uib	Unspecified
	0	losd	Loss of Signal
2	High Byte Indicator Bits		Default is all 1's (OxFF)
	Bit	**Content**	**Description**
	7-0	Reserved	Set the bits to '0' for future compatibility.
HDSL1			
1	Low Byte Indicator Bits		Default is all 1's (OxFF)
	Bit	**Content**	**Description**
	7	rcbe	Repeator Central Block Error
	6	Rrbe	Repeator Remote Block Error
	5	Hrp	HDSL Repeater Present
	4	Bpv	Bipolar Violation
	3	ps2	Power Status Bit 2
	2	ps1	HTU-R Power Status
	1	Febe	Far End Block Error
	0	Losd	Loss of Signal
2	High Byte Indicator Bits		Default is all 1's (OxFF)
	Bit	**Content**	**Description**
	7-5	Reserved	Set the bits to '0' for future compatibility.
	4	uib	Unspecified bit
	3	uib	Unspecified bit
	2	rtr	Ready to Receive
	1	rta	Remote Terminal Alarm
	0	rega	Repeater Alarm
IDSL			
For HTU-R/NT i.e. NT to LT(network) direction			
1	Low Byte Indicator Bits		
	Bit	**Content**	**Description**
	7	ntm	NT in test mode bit (0 indicates test mode)

	6	ps2	Power status bit 2
	5	febe	Far End Block Error (0 indicates errored superframes)
	4	1	Must be set to 1
	3	ps1	Power status bit 1
	2	1	Must be set to 1
	1	1	Must be set to 1
	0	act	Start-up bit
2	Low Byte Indicator Bits		
	Bit	**Content**	**Description**
	7-4	Reserved	Reserved
	3	1	Must be set to 1
	2	sai	S-activation indication bit (optional- Setting to 1 will activate S/T)
	1	1	Must be set to 1
	0	cso	cold-start-only bit (1 indicates cold-start-only)

15.11.3 DSL Framer Enable Automatic Update of Indicator Bits

DSL Framer Enable Auto-Update of Indicator Bits
This command enables the automatic handling of indicator bits by theZipWirePlus device. When the corrosponding control bit is set to one, the indicator bit will automatically be update by the ZipWirePlus device. When cleared(0) it is up to the _DSL_WRITE_IND_BITS API command to update the bits. (Currently, the only bit that can optionally be automatic is the RTR bit). The exact bit definitions vary from frame structure to frame structure (G.shdsl, HDSL2, HDSL1, IDSL etc). Currently this command is not supported for HDSL2, IDSL and G.shdsl Modes.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_AUTO_IND	0x62	Control	2	None

Incoming Parameters

Byte	Content		Description
HDSL1			
1	Low Byte Indicator Bits		
	Bit	**Content**	**Description**
	0-7	Reserved	Set the bits to ‘0’ for future compatibility.
2	High Byte Indicator Bits		
	Bit	**Content**	**Description**
	7-3	Reserved	Set the bits to ‘0’ for future compatibility.
	2	rtr_cntl	Ready to Receive control
	0-1	Reserved	Set the bits to ‘0’ for future compatibility.

15.11.4 DSL Framer Read Indicator Bits

DSL Framer Read Indicator Bits				
This command determines the indicator bits received from the incoming DSL frame. The exact bit definitions vary from frame structure to frame structure (G.shdsl, HDSL2, HDSL1, IDSL). Note: 1) For HDSL1 mode, "rtr" is currently the only bit that can be optionally updated automatically in the firmware. "crc_error" bit is updated ever 6 ms. 2) For G.shdsl or HDSL2 modes, this command has not been implemented. 3) For IDSL modes, If the unsolicited interrupt for this API command is enabled, the ZipWirePlus will send this command to the host to signal on change of any one of the indicator bits specified by the host.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_READ_IND_BITS	0xD0	Status	None	2

Outgoing Parameters

Byte	Content		Description
	HDSL1		
1	Low Byte Indicator Bits		
	Bit	**Content**	**Description**
	7	rcbe	Repeator Central Block Error
	6	rrbe	Repeator Remote Block Error
	5	hrp	HDSL Repeater Present
	4	bpv	Bipolar Violation
	3	ps2	Power Status Bit 2.
	2	ps1	HTU-R Power Status
	1	febe	Far End Block Error
	0	losd	Loss of Signal
2	High Byte Indicator Bits		
	Bit	**Content**	**Description**
	7-6	Reserved	Not Used
	5	crc_error	0 indicates Rx CRC Error
	4	uib	Unspecified

	3	uib	Unspecified
	2	rtr	Ready to Receive
	1	rta	Remote Terminal Alarm
	0	rega	Repeater Alarm
HDSL1			
For HTU-R/NT i.e. NT to LT(network) direction			
1	Low Byte Indicator Bits		
	Bit	**Content**	**Description**
	7	ntm	NT in test mode bit (0 indicates test mode)
	6	ps2	Power status bit 2
	5	febe	Far End Block Error (0 indicates errored superframes)
	4	1	Must be set to 1
	3	ps1	Power status bit 1
	2	1	Must be set to 1
	1	1	Must be set to 1
	0	act	Start-up bit
2	High Byte Indicator Bits		
	Bit	**Content**	**Description**
	7-4	Reserved	Reserved
	3	1	Must be set to 1
	2	sai	S-activation indication bit (optional- Setting to 1 will activate S/T)
	1	1	Must be set to 1
	0	cso	cold-start-only bit (1 indicates cold-start-only)

15.11.5 DSL Framer Write Z-Bits

DSL Framer Write Z-Bits
This command determines the Z-bits to be transmitted out the DSL frame. Each outgoing DSL frame contains 48 or 96 Z-bits. The bytes received from the host will be set on the next frame boundary (typically 6ms) and will continue to be sent on all subsequent frames until this command is issued again. This command is valid for M28950 only. ***Note1:*** Z-bits are double buffered to guarantee uninterrupted Z-bits flow. The host may initially send two blocks of Z-bits concurrently, then a single block at each 6 ms interrupt. The first block of 6 bytes will be sent in the next frame and the second block will be transmitted on all subsequent frames until this command is issued again. ***Note2:*** Based on the value of Z-bits provided in _DSL_FR_HDSL_CONFIG API command, each DSL frame contains 48 (1 bit in each sub-block) or 96 Z-bits (2 bits in each sub-block).

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_WRITE_ZBITS	0x63	Control	6, 12 or 24(See the *Note* above)	None

Incoming Parameters

1 Z-bit in each sub-block

Byte	Content	Description
1-6	Z_1 to Z_{48}	First block of Z-bits Z-bits transmitted in the next outgoing frame Z_1 is the least significant bit of the first byte and Z_{48} is the most significant bit of the last byte.
7-12	Z_1 to Z_{48}	Second block of Z-bits Z_1 is the least significant bit of the first byte and Z_{48} is the most significant bit of the last byte.

2 Z-bits in each sub-block

Byte	Content	Description
1-12	Z_1 to Z_{96}	First block of Z-bits Z-bits transmitted in the next outgoing frame Z_1 is the least significant bit of the first byte and Z_{96} is the most significant bit of the last byte.
13-24	Z_1 to Z_{96}	Second block of Z-bits Z_1 is the least significant bit of the first byte and Z_{96} is the most significant bit of the last byte.

15.11.6 DSL Framer Read Z-Bits

DSL Framer Read Z-Bits
This command determines the Z-bits received from the most recent DSL frame. Each incoming frame contains either 48 or 96 Z-bits. After issuing this command, the host will receive Z-bits from the last incoming frame. Z-bits are double buffered to guard against Rx Z-bit overflow. Note 1: This API command can be sent to the host every 6 ms unsolicitedly. The host needs to issue _DSL_INTR_HOST_MASK (0x50) and _DSL_INTR_API_SUBMASK (0x51) API commands to enable the feature. Note 2: Based on the value of Z-bits provided in _DSL_FR_HDSL_CONFIG API command, each frames contains 48 (1 bit in each sub-block) or 96 Z-bits (2 bits in each sub-block). Note 3: If unsolicited interrupt is enabled, the last byte of the incoming parameter contains the number of blocks of Z-bits, the host can send to the ZipWirePlus.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_READ_ZBITS	0xD1	Status	None	6 or 12

Outgoing Parameters		
1 Z-bit in each sub-block		
Byte	**Content**	**Description**
1-6	Z$_1$ to Z$_{48}$	Z-bits extracted from the last incoming frame Z$_1$ is the least significant bit of the first byte and Z$_{48}$ is the most significant bit of the last byte.
7	Available room in Z-bits buffer	0x00: The z-bit buffer is full 0x01: The Z-bit buffer has room for 1 block of Z-bits (6 bytes) 0x02: The Z-bit buffer has room for 2 blocks of Z-bits (6 bytes each)
2 Z-bits in each sub-block		
Byte	**Content**	**Description**
1-12	Z$_1$ to Z$_{96}$	Z-bits extracted from the last incoming frame Z$_1$ is the least significant bit of the first byte and Z$_{96}$ is the most significant bit of the last byte.
13	Available room in Z-bits buffer	0x00: The z-bit buffer is full 0x01: The Z-bit buffer has room for 1 block of Z-bits (12 bytes) 0x02: The Z-bit buffer has room for 2 blocks of Z-bits (12 bytes each)

15.11.7 DSL Framer Write EOC Bits

15.11.7.1 HDSL1

DSL Framer Write EOC Bits
This command determines the EOC bits to be transmitted out the HDSL1. The bits will be set on the next frame boundary (typically 6mS for HDSL1) and will continue to be sent on all subsequent frames until this command is issued again. This command is valid for M28950 device only. In HDSL1 frame structure, EOC channel uses 13 of available 16 bits. ***Note 1:*** EOC bits are double buffered to guarantee uninterrupted EOC bits flow. The host processor may initially send two blocks of EOC bits concurrently, then a single block at each 6 ms interrupt. The first block of 13 bits will be sent in the next frame and the second block will be transmitted on all subsequent frames until this command is issued again. The host processor should write the number of EOC blocks sent to the ZipWirePlus in this command in Byte 1. ***Note 2:*** The host processor can send both EOC and Z-bits concurrently. No additional action needs to be taken by the host processor. The ZipWirePlus will be able to determine whether the Z-bits are delivered in addition to EOC bits by examining the length of the incoming parameters. Since Z-bits are also double buffered, the host processor may send two blocks of Z-bits concurrently by writing 2 to Z_BLOCK_CNT parameter and 12 bytes of Z-bits to in the next bytes after Z_BLOCK_CNT parameter. ***Note 3***: In addition to EOC and Z bits, the host processor may deliver indicator bits concurrently. No additional action needs to be taken by the host processor. The ZipWirePlus device will be able to determine whether the indicator bits are delivered in addition to EOC and Z bits by examining the length of the incoming parameters. ***Note 4***: If either of the buffers which holds EOC or Z-Bits doesn't have room to write all the EOC or Z-bits sent by the host processor, the ZipWirePlus will reject it and will notify the host processor by responding ACK_BUSY.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_WRITE_EOC	0x64	Control	3, 5, 12, 18 or 21 (See the *Note* above)	None

Incoming Parameters

HDSL1

Byte	Content	Description
1	EOC_BLOCK_CNT	Number of EOC blocks
2-3	EOC Bits	Byte 1: Low byte Byte2: High Byte (Least-significant 5 bits are used)
4-5	EOC bits	Second block of EOC bits if EOC_BLOCK_CNT is set to 2. Byte 1: Low byte Byte2: High Byte (Least-significant 5 bits are used)
6	Z_BLOCK_CNT	Number of Z blocks
7-12	Z_1 to Z_{48}	Z_1 is the least significant bit of the first byte and Z_{48} is the most significant bit of the last byte.

13-18	Z_1 to Z_{48}	Second block of EOC bits if Z_BLOCK_CNT is set to 2. Z_1 is the least significant bit of the first byte and Z_{48} is the most significant bit of the last byte.
19	IND_BLOCK_CNT	Set to 1
20-21	Indicator Bits	Byte 1: Low byte Byte2: High Byte (Least-significant 5 bits are used) (Refer to Section 15.11.2)

15.11.7.2 IDSL NT

DSL Framer Write EOC Bits				
This command is send by the host to indicate the EOC functions supported for processing by the IDSL NT modem. The value of an EOC function ranges from 0 to 255. Bytes 1-32 can be used as a bit mask to specify the supported functions. For example, if the EOC function supported by the host are 1, 5, 10, 36, 54, and 112, then the values written: 0x6 to Byte 1, 0x4 to Byte 2, 0x10 to Byte 5, 0x40 to Byte 7, and 0x1 to Byte 15.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_WRITE_EOC	0x64	Control	32	None

Incoming Parameters		
Byte	**Content**	**Description**
1-32	Bit Mask	Specifies the supported EOC functions

15.11.8 DSL Framer–DSL Configuration

DSL Framer–DSL Configuration				
Configure the DSL Framer DSL block. This only applies when the DSL Framer is enabled.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_FR_HDSL_CONFIG	0x11	Control	1	None

Incoming Parameters			
Byte	**Content**		**Description**
1	DSL Configuration		See bit-field description below.
	Bit	**Content**	**Description**

	7	Reserved	Set the bits to '0' for future compatibility.
	6	Scrambler/ Descrambler Disable	This bit can be used to disable the scrambler/descrambler in the ZipWirePlus device and only applicable after the DSP has completed training, This is reserved only for special applications where the scramble/descramble feature is done outside the ZipWirePlus device or at the ATM layer. Upon retrain, the system returns to the default value. 0 = Not Bypassed (default) 1 = Bypassed (Disabled)
	5:4	ZBits to Host	When enabled DSL OH will increase to allow Zbits to be fed from the Host interface. 0x00 = Disable, runs normal with a 8kbps DSL Overhead (default). 0x01 = Enable, 1 Zbit via Host can be accessed (with a 16 kbps DSL Overhead). 0x02 = Enable, 2 Zbits via the Host can be accessed (with a 24 kbps DSL Overhead).
	3	Dying Gasp Control	0 = S/W Control (default). Whenever using the software control option, the user issues the _DSL_WRITE_IND_BITS (0x60) to set the Dying Gasp status. 1 = H/W control–Whenever using the hardware control option, the software polls the DEVADR2 pin to set the Dying Gasp status. The Dying Gasp indicator, and therefore DEVADR2, is an active low signal. The DEVADR2 pin should have a weak pull-up (10 kOhm). In addition, the Destination field must be set to 0xFF during download.
	2	I Bits to Auxillary Channel	When set, the i-bits in the DSL will route to/from the Auxillary. 0 = iBits route to/from PCM interface.(default) 1 = iBits route to/from AUX interface. (Requires AUX_Enable = 1)..
	1	AUX Enable	AUX Enable is used to enable or disable the Auxiliary channel. 0 = Use DBANK (default) 1 = Use AUX Channel (driver will automatically set this option when the _ATM_PHY_IF_MODE (0x1E) is set to the DSL Framer Aux option.
	0	AUX Mode	Auxiliary Channel Mode. 0 = THLOAD and RHMARK pins are active-high during the relevant data 1 = THLOAD and RHMARK pins are clock-gated. Coincides with the relevant data (default)

15.11.9 DSL Data Rate

DSL Data Rate				
The Data Rate command sets the bit pump DSL Data Rate. This command is only provided for internal testing. The customer should not issue this command. However, the customer could issue the _DSL_READ_CONTROL (opcode 0x80) with this parameter to read back the current DSL data rate. The Data Rate parameter is determined by the following equation: X = Data Rate / 8,000 The supported range of X is 8–580 (or data rate = 64–4,640 kbps respectively).				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_DATA_RATE	0x0E	Control	2	None

Incoming Parameters		
Byte	**Content**	**Description**
1-2	Data Rate	Contains the lower 10 bits of the Symbol Rate parameter X. The low byte is programmed first
		NOTE: 16-bit value = (high byte << 8) + (low byte).

15.11.10 DSL Configuration Pair Identification Feature

DSL Configuration Pair Identification				
This command is valid for HDSL1 mode only. _DSL_CONFIG_PID will turn on/off the Pair-Identification functionality (default off). This is required to be on when training against a PID enabled far-end, or the 8973/53 release code of 4.0 or higher. The PID is used in the HDSL1 system to set a loop to a particular number, so the far-end know how to map the data.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_CONFIG_PID	0x2B	Control	1	None

Incoming Parameters			
Byte	**Content**		**Description**
1	PID Configuration		See the bit-field description below.
	Bit	**Content**	**Description**
	7:4	Reserved	Reserved (set to 0x00)
	3:1	Loop Id	0x01 – HDSL1 pair 1 (default) 0x02 – 0x07 Reserved
	0	Enable	The Enable bit is used to enable or disable transmission and receive requirement of the Pair Identification protocol. 0 = Disable PID(default). 1 = Enable PID.

15.11.11 DSL Framer Configuration

DSL Framer Configuration				
This command is valid for HDSL1 mode only. _DSL_FR_2B1Q_CONFIG provides a generic mechanism to set the SyncWord in 2B1Q mode and to set the framer interface to switch between SERIAL_SWAP (default) and SERIAL. The default value for SyncWord is 0x72.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_FR_2B1Q_CONFIG	0x65	Control	1	None

Incoming Parameters			
Byte	**Content**		**Description**
1	2B1Q Configuration		See the bit-field description below.
	Bit	**Content**	**Description**
	7:4	Reserved	Reserved (set to 0x00)
	3	2 Level OH Control	0: (default) At 2 level the OH is set to all ones. 1: At 2 level the OH is valid overhead (non standard).
	2	4 Level OH Control	0: After framer in sync framer enables overhead and normal data. Ignore rtr during training. 1:(default) Waits for rtr bit per standard to switch to normal data.
	1	Sync Word	0 = (default) 0x72 1 = 0x27
	0	Framer Config	0: (default)SERIAL_SWAP 1: SERIAL

15.12 DSL Mapper Configuration and Status API Commands

15.12.1 Transmit DSL Mapper Value

Transmit DSL Mapper Value				
The Transmit DSL Mapper Value command allows the user to control the data transmitted to the DSL channel. The Transmit DSL Payload Mapper table is implemented with an 8 x 64 RAM that can accommodate up to 128 table entries. Each byte entry is comprised of 2 table entries labeled A and B where A is processed first. Each table entry can then process from 1 to 4 time slots, thus providing up to 512 (4 x 128) time slots per Transmit DSL frame. The 64-byte entries are indexed from 0 to 63 and are stored in data RAM. The Transmit DSL Mapper Write command is used to write the table from the data RAM to the device.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_TH_MAPPER_VALUE	0x34	Control	65	None

Incoming Parameters		
Byte	**Content**	**Description**
1	Starting Address	1-byte value indicating the table entry starting address to write (zero based). The Starting Address can be between 0–63 corresponding to a total of 64 bytes for the TH_MAPPER table entry.
2–65	Table Entry	See bit-field description below. The default is based on the system configuration.

	Bit	Content	Description
	7:6	Number of Time Slots (NTS) B	Indicates the number of time slots to be controlled in each table entry (00 stands for 1 and 11 stands for 4).
	5:4	Data Source B	Specifies the Transmit DSL data source for each time slot. 00 = Override/Insert Data Bank register 1 (DBANK_1) 01 = PCM Payload–Read data from PCM Transmit FIFO 10 = Narrowband Payload–Read data from Narrowband Transmit FIFO 11 = If *AUX_EN* = 0—Override/Insert Data Bank register 2 (DBANK_2). If *AUX_EN* = 1—Payload 3 (Auxiliary channels). The AUX_EN bit is set using the _DSL_FR_HDSL_CONFIG (0x11) API command.
	3:2	Number of Time Slots (NTS) A	Indicates the number of time slots to be controlled in each table entry (00 stands for 1 and 11 stands for 4).
	1:0	Data Source A	Specifies the Transmit DSL data source for each time slot. 00 = Override/Insert Data Bank register 1 (DBANK_1) 01 = PCM Payload–Read data from PCM Transmit FIFO 10 = Narrowband Payload–Read data from Narrowband Transmit FIFO 11 = If *AUX_EN* = 0—Override/Insert Data Bank register 2 (DBANK_2). If *AUX_EN* = 1—Payload 3 (Auxiliary channels). The AUX_EN bit is set using the _DSL_FR_HDSL_CONFIG (0x11) API command.

15.12.2 Transmit DSL Mapper Write

Transmit DSL Mapper Write				
Writes the Transmit DSL Mapper table to the device. This should be called only after the Transmit DSL Mapper table has been completely filled in. The appropriate resets are issued.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_TH_MAPPER_WRITE	0x35	Control	1	None

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.

15.12.3 Transmit DSL Mapper Read

Transmit DSL Mapper Read				
This command reads back the Transmit DSL Payload Table setting (this reads back the local variable, not the hardware registers)				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_TH_MAPPER_READ	0xA5	Status	2	L

Incoming Parameters			
Byte	**Content**		**Description**
1	Starting Address		1-byte value indicating the table entry starting address to read (zero based). The Starting Address can be between 0–63 corresponding to a total of 64 bytes for the TH_MAPPER table entry.
2	Number of table entry to read (L)		Number of table entries to read (zero based). To read back (L) bytes the table entry parameter should be set to (L-1).
Outgoing Parameters			
Byte	**Content**		**Description**
1-L	Table Entry		See bit-field description below. The default is based on the system configuration.
	Bit	**Content**	**Description**
	7:6	Number of Time Slots (NTS) B	Indicates the number of time slots to be controlled in each table entry (00 stands for 1 and 11 stands for 4).
	5:4	Data Source B	Specifies the Transmit DSL data source for each time slot. 00 = Override/Insert Data Bank register 1 (DBANK_1) 01 = PCM Payload–Read data from PCM Transmit FIFO 10 = Narrowband Payload–Read data from Narrowband Transmit FIFO 11 = If *AUX_EN* = 0—Override/Insert Data Bank register 2 (DBANK_2). If *AUX_EN* = 1—Payload 3 (Auxiliary channels). The AUX_EN bit is set using the _DSL_FR_HDSL_CONFIG (0x11) API command.
	3:2	Number of Time Slots (NTS) A	Indicates the number of time slots to be controlled in each table entry (00 stands for 1 and 11 stands for 4).
	1:0	Data Source A	Specifies the Transmit DSL data source for each time slot. 00 = Override/Insert Data Bank register 1 (DBANK_1) 01 = PCM Payload–Read data from PCM Transmit FIFO 10 = Narrowband Payload–Read data from Narrowband Transmit FIFO 11 = If *AUX_EN* = 0—Override/Insert Data Bank register 2 (DBANK_2). If *AUX_EN* = 1—Payload 3 (Auxiliary channels). The AUX_EN bit is set using the _DSL_FR_HDSL_CONFIG (0x11) API command.

15.12.4 Receive DSL Mapper Value

Receive DSL Mapper Value						
The Receive DSL Mapper Value command allows the user to control the data coming from the DSL channel. The Receive DSL Payload Mapper table is implemented with an 8 x 64 RAM that can accommodate up to 128 table entries. Each byte entry is comprised of 2 table entries labeled A and B where A is processed first. Each table entry can then process from 1 to 4 time slots, thus providing up to 512 (4 x 128) time slots per Receive DSL frame. The 64-byte entries are indexed from 0 to 63 and are stored in data RAM. The Receive DSL Mapper Write command is used to write the table from the data RAM to the device.						
C Constant			**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_RH_MAPPER_VALUE			0x36	Control	65	None
Incoming Parameters						
Byte	**Content**		**Description**			
1	Starting Address		1-byte value indicating the table entry starting address to write (zero based). The Starting Address can be between 0–63 corresponding to a total of 64 bytes for the RH_MAPPER table entry.			
2–65	Table Entry		See bit-field description below. The default is based on the system configuration.			
	Bit	**Content**	**Description**			
	7:6	Number of Time Slots (NTS) B	Indicates the number of time slots to be controlled in each table entry (00 stands for 1 and 11 stands for 4).			
	5:4	Data Source B	Specifies the Receive DSL data source for each time slot. 00 = Discard. 01 = Insert into Receive PCM FIFO. 10 = Insert into Receive Narrowband FIFO. 11 = Output to RHAUX pin, auxiliary channel.			
	3:2	Number of Time Slots (NTS) A	Indicates the number of time slots to be controlled in each table entry (00 stands for 1 and 11 stands for 4).			
	1:0	Data Source A	Specifies the Receive DSL data source for each time slot. 00 = Discard. 01 = Insert into Receive PCM FIFO. 10 = Insert into Receive Narrowband FIFO. 11 = Output to RHAUX pin, auxiliary channel.			

15.12.5 Receive DSL Mapper Write

Receive DSL Mapper Write				
Writes the receive DSL Mapper table to the device. This should be called only after the receive DSL Mapper table has been completely filled in. The appropriate resets are issued.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_RH_MAPPER_WRITE	0x37	Control	1	None

Incoming Parameters		
Byte	Content	Description
1	Reserved	Set to 0x00 for future compatibility.

15.12.6 Receive DSL Mapper Read

Receive DSL Mapper Read				
This command reads back the Receive DSL Payload Table setting (this reads back the local variable, not the hardware registers)				
C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_RH_MAPPER_READ	0xA6	Status	2	L

Incoming Parameters		
Byte	Content	Description
1	Starting Address	1-byte value indicating the table entry starting address to read (zero based). The Starting Address can be between 0–63 corresponding to a total of 64 bytes for the RH_MAPPER table entry.
2	Number of table entry to read (L)	Number of table entries to read (zero based). To read back (L) bytes the table entry parameter should be set to (L-1).

Outgoing Parameters			
Byte	Content		Description
1-L	Table Entry		See bit-field description below. The default is based on the system configuration.
	Bit	Content	Description
	7:6	Number of Time Slots (NTS) B	Indicates the number of time slots to be controlled in each table entry (00 stands for 1 and 11 stands for 4).
	5:4	Data Source B	Specifies the Receive DSL data source for each time slot. 00 = Discard. 01 = Insert into Receive PCM FIFO. 10 = Insert into Receive Narrowband FIFO. 11 = Output to RHAUX pin, auxiliary channel.
	3:2	Number of Time Slots (NTS) A	Indicates the number of time slots to be controlled in each table entry (00 stands for 1 and 11 stands for 4).
	1:0	Data Source A	Specifies the Receive DSL data source for each time slot. 00 = Discard. 01 = Insert into Receive PCM FIFO. 10 = Insert into Receive Narrowband FIFO. 11 = Output to RHAUX pin, auxiliary channel.

15.13 AFE Configuration and Status API Commands

15.13.1 Analog Front End (AFE) Configuration

Analog Front End (AFE) Configuration				
Configures the ZipWirePlus AFE chip.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_AFE_CONFIG	0x02	Control	1	None

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility

15.13.2 Read AFE Setting

Read AFE Setting				
Reads the current setting of the AFE and hybrid selection. The DSP startup algorithm determines these results. The AFE value should only be read during normal operation.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_AFE_SETTING	0x90	Status	1	1

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility
Outgoing Parameters		
1	AFE Value	1-Byte unsigned (see AFE Bit Definitions below).

AFE Bit Definitions		
AFE Bit	**Description**	**Bit(s) Definition**
7	Reserved	Reserved
6–4	AFC setting corresponding to the Absolute Gain (dB) setting	0 = 0.0 dB 1 = 3.5 dB 2 = 6.0 dB 3 = 7.9 dB 4 = 10.0 dB 5 = 11.6 dB 6 = 13.3 dB 7 = 15.0 dB

		3–0	Reserved	Reserved

15.13.3 Write AFE Transmit Gain

Write AFE Transmit Gain

Writes the AFE Line Driver Transmitter Gain register. This command can be used to increase or decrease the AFE Transmit Gain up to ±1.6dB in 0.1dB steps. The customer may need to adjust the default settings to compensate for application specific hardware differences: such as surge protection, remote power feeding, and so on. The following procedure lists how the user would modify the transmit gain.

- Put the modem In-Service. (DSL System State = In-Service)
- Read the default value of _AFE_TX_GAIN, because you may want to reload the default. (Opcode 0x80 = 0x13 => 0x10, for example)
- Increase the AFE Transmit Power by loading a smaller number. (*e.g.*, Opcode 0x13 = 0x0D)
- Decrease the AFE Transmit Power by loading a larger number. (*e.g.*, Opcode 0x13 = 0x12)

The transmit gain change takes effect immediately.

If this command is issued while in end-to-end operation, the link will most likely drop.

The transmit gain offset is maintained even when switching between different data rates, modes (such as Symmetric vs. Asymmetric), and so on.

The initial transmit gain is set only after a software or hardware reset. No other event will cause it to change.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_AFE_TX_GAIN	0x13	Control	1	None

Incoming Parameters

Byte	Content	Description
1	TX Gain	5-bit unsigned integer value. The default is 0x10.

15.14 Performance Monitoring API Commands

15.14.1 Line Attenuation

Line Attenuation

Requests a value of the far-end signal attenuator. This value is based on measuring the average far-end signal level after echo cancellation. The return value is already adjusted to match the analog gain value (AAGC). The attenuation is calibrated against a 150 Khz sine wave.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_FAR_END_ATTEN	0x82	Status	1	1

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.
Outgoing Parameters		
Byte	**Content**	**Description**
1	Attenuation	1-byte unsigned integer X (0 ≤ × ≤ 255) indicating the signal power attenuation in units of 0.5 dB. For example, a value of 35 means a total cable attenuation of 17.5 dB.

15.14.2 Noise Margin

Noise Margin				
Requests a value of the Noise Margin of the Receiver (NMR). The noise margin is defined as the maximum tolerable increase in external noise power that still allows for BER of less than 1x 10 $^{-7}$. The value is based on measuring the average absolute level of the noise at the input to the slicer. The noise margin format matches the definition in the G.shdsl, HDSL2 and HDSL1 standards.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_NOISE_MARGIN	0x83	Status	1	1

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.
Outgoing Parameters		
Byte	**Content**	**Description**
1	NMR	1-byte signed integer X (–128 ≤ × ≤ 127) indicating the noise margin in units of 0.5 dB. For example, a value of –4 means a noise margin of –2.0 dB. ***NOTE:*** NMR is limited to +30dB to avoid large steps.

15.14.3 Power Back-Off Result

Power Back-Off Result				
This command returns the power back-off result. This only applies to HDSL2 OPTIS and G.shdsl mode.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_POWER_BACK_OFF_RESULT	0x94	Status	1	2

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.
Outgoing Parameters		
Byte	**Content**	**Description**
1	Transmit Power Back Off	1-byte unsigned integer indicating the transmit (near end) power back-off result in units of 1.0 dB.
2	Receive Power Back Off	1-byte unsigned integer indicating the receive (far end) power back-off result in units of 1.0 dB.

15.14.4 Read ZipWirePlus System Performance Error Counters

Read ZipWirePlus System Performance Error Counters				
Queries the ZipWirePlus System Performance Error counters. These error counters are accumulated when the ZipWirePlus device reaches normal operation or when the Clear Error Counter command was issued. The System performance error counters are 2-bytes wide.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_SYSTEM_PERF_ERR_CTRS	0xA2	Status	1	4

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.
Outgoing Parameters		
Byte	**Content**	**Description**
1–2	Startup Attempts	16-bit value specifying the number of startup attempts. The low byte is sent first followed by the high byte.
3–4	Startup Successful	16-bit value specifying the number of successful startup attempts. The low byte is sent first followed by the high byte.
NOTE: 16-bit value = (high byte << 8) + (low byte)		

15.14.5 Read ZipWirePlus DSL Performance Error Counters

Read ZipWirePlus DSL Performance Error Counters				
Queries the ZipWirePlus DSL Performance Error counters. These error counters are accumulated when the ZipWirePlus device reaches normal operation or when the Clear Error Counter command was issued. The DSL performance error counters are 2-bytes wide.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_HDSL_PERF_ERR_CTRS	0x9E	Status	1	10

Incoming Parameters		
Byte	**Content**	**Description**
1	Clear Error Counter	0 = Read Error Counter Only 1 = Read and Clear Error Counter
Outgoing Parameters		
Byte	**Content**	**Description**
1–2	Out-of-Sync (LOSW)	16-bit value specifying the number of Out-of-Sync errors. The low byte is sent first followed by the high byte.
3–4	SEGD	16-bit value specifying the number of SEGD errors. The low byte is sent first followed by the high byte.
5–6	CRC	16-bit value specifying the number of CRC errors. The low byte is sent first followed by the high byte.
7–8	SEGA	16-bit value specifying the number of SEGA errors. The low byte is sent first followed by the high byte.
9–10	LOSD	16-bit value specifying the number of LOSD errors. The low byte is sent first followed by the high byte.
NOTE: 16-bit value = (high byte << 8) + (low byte)		

15.14.6 Read ATM PHY Performance Error Counters

Read ATM PHY Performance Error Counters				
Queries the ATM PHY Performance Error Counters. These error counters are accumulated from when the ZipWirePlus device reaches normal operation or when the Clear Error Counter command was issued. The ATM PHY performance error counters are 2-bytes wide.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_ATM_PHY_PERF_ERR_CTRS	0xB9	Status	1	8

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.
Outgoing Parameters		
Byte	**Content**	**Description**
1–2	Out of Sync (LOCD)	16-bit value specifying the number of Loss of Cell Delineation errors. The low byte is sent first followed by the high byte.
3–4	Corrected HEC	16-bit value specifying the number of corrected HEC errors. The low byte is sent first followed by the high byte.

5–6	Uncorrected HEC	16-bit value specifying the number of uncorrected HEC errors. The low byte is sent first followed by the high byte.
7-8	nlcd Defect Total Count	16-bit unsigned value specifying the count of the total number of nlcd defects.
		NOTE: 16-bit value = (high byte << 8) + (low byte)

15.14.7 Read ATM PHY Cell Counters

Read ATM PHY Cell Counters

Queries the ATM PHY Cell Counters. These error counters are accumulated from when the ZipWirePlus device reaches normal operation or when the Clear Error Counter command was issued. The ATM PHY performance error counters are 4-bytes wide.

When operating at the maximum data rate of 4640kbps, the counters would overflow after 109 hours

53 bytes per cell = 53 x 8 bits per cell = 424 bits per cell

4640 kbits per second ÷ 424 bits per cell = 10943 cells per second

A 32-bit counter can hold 2^{32} cells

2^{32}cells /. 10943 cells per second = 392485 seconds = 109 hours

Opcode C constant	Opcode C constant	Opcode Type	Incoming Bytes	Outgoing Bytes
_ATM_PHY_CELL_CTRS	0xBA	Status	1	17

Incoming Parameters

Byte	Content	Description
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters

Byte	Content	Description
1	Overflow Status	1-byte indicating the overflow status. This is a read-clear indicator. See the *ATM* PHY *Overflow Status Bit Definitions* table below.

ATM PHY Overflow Status Bit Definitions

Bit	Description	Bit(s) Definition
7–4	Reserved	Reserved
3	Receive idle cell count overflow	0 = No 1 = Yes
2	Transmit idle cell count overflow	0 = No 1 = Yes
1	Receive cell count overflow	0 = No 1 = Yes

		0	Transmit cell count overflow	0 = No 1 = Yes
2–5	Transmit Cell Count	32-bit unsigned value specifying the number of transmitted cells, including idle cells. The low byte is sent first and the most significant byte sent last.		
6–9	Receive Cell Count	32-bit unsigned value specifying the number of received cells, including idle cells. The low byte is sent first and the most significant byte sent last.		
10–13	Transmit Idle Cells	32-bit unsigned value specifying the number of idle cells transmitted. The low byte is sent first followed by the high byte.		
14–17	Receive Idle Cells	32-bit unsigned value specifying the number of idle cells received. The low byte is sent first followed by the high byte.		

NOTE: 32-bit value = (byte 4 << 24) + (byte 3 << 16) + (byte 2 << 8) + (byte 1)

15.14.8 Available Seconds and Total Seconds

Available Seconds and Total Seconds

Requests the available seconds and total seconds since power-on or reset.

Available Seconds refers to the total accumulated time for which the system is in normal operation (active transmit/receive state) and is passing data across the link.

Total Seconds refers to the total time the system has been in operation.

Errored Seconds refer to the number of seconds in which a CRC Error was detected. Errored Seconds only accumulate while the system is in normal operation.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_TIME	0x9D	Status	1	16

Incoming Parameters

Byte	Content	Description
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters

Byte	Content	Description
1–4	Available Seconds	32-bit value specifying the total number of Available seconds since power-on or reset; low byte first, most significant byte last which corresponds to byte 0, byte 1, byte 2, and byte 3.
5–8	Total Seconds	32-bit value specifying the total number seconds elapsed since power-on or reset; low byte first, most significant byte last which corresponds to byte 0, byte 1, byte 2, and byte 3.
9–16	reserved	reserved

NOTE: 32-bit value = (byte 4 << 24) + (byte 3 << 16) + (byte 2 << 8) + (byte 1)

15.15 EOC API Commands

15.15.1 EOC Transmit Send Message

EOC Transmit Send Message

This command sends an EOC message across the EOC channel. The host processor builds up the appropriate request or response message (excluding the FCS and data transparencies). The internal processor will queue the message to be sent on subsequent DSL frames. The ACK_STATUS byte will return one of the following codes:

_ACK_PASS–message was successfully placed in EOC transmit queue.

_ACK_NOT_AVAILABLE–EOC channel is unavailable. DSL link not up.

_ACK_NO_RESULT–Not enough available memory in the transmit queue. Message discarded.

NOTE: The internal processor will continuously transmit the sync octets (flag - 0x7E) whenever there is no EOC message to send.

NOTE: After 2 to 73 bytes of EOC message data the software will add 2 more Bytes for Frame Check Sum (FCS).

NOTE: This command works for G.shdsl and HDSL2 modes only. This is not supported for HDSL1 Mode.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_EOC_TX_SEND_COMMAND	0xB0	Status	up to 73 Bytes (L)	3

Incoming Parameters

Byte	Content	Description
1 - L	EOC Message	EOC Message. See Section 8.2.2.1 for EOC message format. The EOC message includes the address byte (source and destination address), message ID and message data. The ZipWirePlus is responsible for the sync octets, data transparency and frame check sum bytes.

Outgoing Parameters

Byte	Content	Description
1	EOC Address Byte	Echoed address byte from message
2	EOC Message ID	Echoed message ID byte from the message
3	Message Handle	Non-zero message handle uniquely identifies the message and is needed to query the status of the message or delete the message.

NOTE: The outgoing bytes are only present when the ACK_STATUS byte is _ACK_PASS.

15.15.2 Get EOC Transmit Message Status

Get EOC Receive Message							
This command is used to get the status of an EOC Transmit message.							
C Constant			Opcode	Type	Incoming Bytes	Outgoing Bytes	
_EOC_TX_GET_MSG_STATUS			0xB2	Status	1	2	
Incoming Parameters							
Byte	**Content**	**Description**					
1	Message Handle	Message Handle is unique message identifier returned when the _EOC_TX_SEND_COMMAND (0xB0) is issued.					
Outgoing Parameters							
Byte	**Content**	**Description**					
1	Message Handle	Echoed message handle					
2	Message Status	Message Status Definition Below.					
		Value	**Option**	**Description**	**Parameter (C Constant)**		
		0x00	API Not Applicable	Status Not Applicable (Message has been sent and/or deleted from the queue)	_EOC_STATUS_NA		
		0x01	EOC Message Pending	Message is still in the queue waiting to be sent	_EOC_STATUS_PENDING		

15.15.3 Get EOC Receive Message

15.15.3.1 For G.shdsl & HDSL2 Modes

Get EOC Receive Message				
This command is used to get the next available received EOC message (excluding FCS and data transparencies). The ACK_STATUS byte will return one of the following codes: _ACK_PASS–message available; contents located in outgoing bytes. _ACK_NOT_AVAILABLE–EOC channel is unavailable. DSL link not up. _ACK_NO_RESULT–No message available in EOC receive queue.				
NOTE: The software removes 2 Bytes for the Frame Check Sum (FCS) from the receive EOC message automatically.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_EOC_RX_GET_MSG	0xB1	Status	1	up to 73 Bytes (L)

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.
Outgoing Parameters		
Byte	**Content**	**Description**
1 - L	EOC Message	EOC Message. See Section 8.2.2.1 for EOC message format. The EOC message includes the address byte (source and destination address), message ID and message data. The ZipWirePlus performs data transparency on the received bytes and does not return the frame check sum bytes.

15.15.3.2 For HDSL1 Mode (M28950 device only)

The host can asynchronously access the the state of most recently received EOC bits by sending _EOC_RX_GET_MSG command. The M28950 device delivers the EOC bits in the following format using _EOC_RX_GET_MSG solicited API command .

DSL Framer Read EOC Bits				
This command determines the EOC bits received from the incoming DSL frame. This command provides a mechanism to read the EOC bits in HDSL1 frames. EOC channel uses 13 of available 16 bits.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_EOC_RX_GET_MSG	0xB1	Status	1	2

Outgoing Parameters		
Byte	**Content**	**Description**
1-2	EOC bits	Byte 1: Low byte Byte2: High Byte (Least-significant 5 bits are used)

This API command can be sent to the host every 6 ms unsolicitedly. The host needs to issue _DSL_INTR_HOST_MASK (0x50) and _DSL_INTR_API_SUBMASK (0x51) API commands to enable the feature. The format of the EOC_RX_GET_MSG API command (if the unsolicited interrupt is enabled) is described in the following Table.

DSL Framer Read EOC Bits
This command provides a mechanism to read the EOC bits in HDSL1 frames using an unsolicited interrupt every 6 ms. In addition to sending EOC bits, the ZipWirePlus reports how many blocks of EOC bits can be delivered by the host in the next _DSL_WRITE_EOC API command. EOC channel uses 13 of available 16 bits. ***Note 1***: If the host attempts to enable the unsolicited interrupt for this API command, by setting Z-Bits (Bit 1) of Byte 3 in the incoming parameters of _DSL_INTR_API_SUBMASK API command, the host can request that both EOC and Z-bits delivered concurrently. This will reduce the number of unsolicited interrupts. Therefore, Bytes 4-9 shall contain Z-bits where Z_1 is the least significant bit of Byte 4 and Z_{48} is the most significant bit of Byte 9. The maximum number of blocks of Z-bits which can be delivered by the host , is written to Byte 10. ***Note 2***: In addition to EOC and Z bits, the host can request that indicator bits to be deliver concurrently by writing 7 to Byte 3 of the incoming parameters of _DSL_INTR_API_SUBMASK API command when enabling the unsolicited interrupt for _EOC_RX_GET_MSG API command.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_EOC_RX_GET_MSG	0xB1	Status	N/A	3, 10, or 12 (See the *Note* above)

Outgoing Parameters		
Byte	**Content**	**Description**
1-2	EOC bits	Byte 1: Low byte Byte2: High Byte (Least-significant 5 bits are used)
3	TX_EOC_BLOCK_CNT	Maximum number of blocks of EOC bits should be delivered by the host in the next _DSL_WRITE_EOC API command.
4-9	Z-bits Z_1 to Z_{48}	Z_1 is the least significant bit of the first byte and Z_{48} is the most significant bit of the last byte.
10	TX_Z_BLOCK_CNT	Maximum number of blocks of Z-bit should be delivered by the host in the next _DSL_WRITE_ZBITS API command.
11-12	Indicator bits	Byte 1: Low byte Byte2: High Byte (Least-significant 5 bits are used)

15.15.3.3 For IDSL Mode (M28950 device only)

Get EOC Receive Message

This command returns the EOC bits received from the incoming DSL frame.
This command provides a mechanism to read the EOC bits in IDSL frames for the NT modem. This command provides a mechanism to read the EOC bits in IDSL frames. The EOC frame is delivered to the host using an unsolicited interrupt.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_EOC_RX_GET_MSG	0xB1	Status	1	3

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters		
Byte	**Content**	**Description**
1	RX_EOC_BLOCK_CNT	Is set to 2 i.e. 2 Bytes
2-3	EOC bits	EOC_1-EOC_{12} : 1 EOC frame EOC_1 is the least significant bit of the first byte

15.15.3.4 HDSL1 Status

HDSL1 Status						
This command returns the pair ID information when in HDSL1 mdoe.						
C Constant			**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_HDSL1_STATUS			0xB4	Status	1	1
Incoming Parameters						
Byte	**Content**		**Description**			
1	Reserved		Set to 0x00 for future compatibility.			
Outgoing Parameters						
Byte	**Content**		**Description**			
1	HDSL1 Status		See the bit-field description below.			
	Bit	**Content**	**Description**			
	7:4	Reserved	Reserved (set to 0x00)			
	3:1	pid_expected	The loop value. 001 – loop 1, 010 – loop 2, 100 – loop 3.			
	0	pid_valid	Set when pair identification is successfully completed by the HDSL1 framer.			

15.15.4 Get EOC Receive Statistics

Get EOC Receive Statistics					
_EOC_RX_GET_STATS will return the accumulated statistics for the EOC receive channel since the last _EOC_RX_GET_STATS request. The counters are read-clear and will stop accumulation at 255. This command does not work for HDSL1 or IDSL Mode.					
C Constant		**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_EOC_RX_GET_STATS		0xAE	Control	1	7
Incoming Parameters					
Byte	**Content**	**Description**			
1	Reserved	Set to 0x00 for future compatibility.			
Outgoing Parameters					
Byte	**Content**	**Description**			
1	CRC Counter	Number of received EOC messages discarded due to CRC Errors.			

2	Overflow Counter	Number of received EOC messages that exceeded the maximum 75 octets–the message was discarded.
3	In Sync Counter	Number of times the EOC has synced up (Valid only for G.shdsl)
4	No Memory Counter	Number of received EOC messages that were discarded because the receive memory queue was full. This will occur if _EOC_RX_GET_MSG (0xB1) wasn't called quickly enough.
5	Invalid Counter	Number of invalid control sequences received. The two valid control sequences use for Data Transparency are 0x7D 0x5E and 0x7D 0x5D.
6	Dropped Frame Counter	Number of dropped frames when the receive EOC buffer is not available.
7	Under Flow Counter	Number of frames with message length less than four. (Includes 2 bytes FCS)

15.15.5 EOC Reset

EOC Reset					
This command is used to reset the EOC transmit and receive queues. This command could be used when the host processor detects a large amount of EOC message errors.					
NOTE: The ZipWirePlus device will automatically reset the EOC whenever the modem successfully trains.					
C Constant		**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_EOC_RESET		0x4D	Control	1	None
Incoming Parameters					
Byte	**Content**	**Description**			
1	Reserved	Set to 0x00 for future compatibility.			

15.16 E1 Framer Configuration and Status API Commands (Valid on M28947 device only)

15.16.1 E1 Framer Configuration

E1 Framer Configuration				
This command configures the Internal E1 Framer.				
Opcode C Constant	**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_E1_PRA_CONFIG	0x44	Control	5	None

Incoming Parameters			
Byte	**Content**		**Description**
1	E1 PRA Config		See bit-field description below. Default = 0x00
	Bit	**Content**	**Description**
	7:5	Reserved	Set the bits to '0' for future compatibility.
	4	Tx TS16 Read Enable	1 = Enable reading TimeSlot 16 and searching CAS Multiframe. 0 = Disable
	3	Rx TS16 Read Enable	1 = Enable reading TimeSlot 16 and searching CAS Multiframe. 0 = Disable
	2	Tx Sync Select Value	1 = Self-align to the multiframe 0 = Align to the external 2ms multiframe reference input (default) (This feature is currently not supported)
	1	Rx Sync Select Value	1 = Set align to the multiframe 0 = Align to the external DSL 6ms reference (default) (This feature is currently not supported)
	0	Mode	Enables the E1 Framer 0–Disabled (default) 1–Enabled
2	Tx Path Config		See bit-field description below. Default = 0x00
	Bit	**Content**	**Description**
	7:5	Reserved	Set the bits to '0' for future compatibility.
	4	TS 16 Generator Mode	0 = Transparent (default) 1 = Manual
	3	A-bit Generator Mode	0 = Transparent (default) 1 = Manual
	2:1	E-bit Generator Mode	00 = Transparent (default) 01 = Manual 10 = Automatic 11 = Reserved
	0	CRC-4 Generator Mode	0 = Transparent (default) 1 = Automatic (recalculate)
3	Tx Path Sa Config		See bit-field description below. Default = 0x00
	Bit	**Content**	**Description**

	7:6	Reserved	Set the bits to ‘0’ for future compatibility.
	5	Sa8 Generator Mode	0 = Transparent (default) 1 = Manual
	4	Sa7 Generator Mode	0 = Transparent (default) 1 = Manual
	3:2	Sa6 Generator Mode	00 = Transparent (default) 01 = Manual 10 = Reserved 11 = Reserved
	1	Sa5 Generator Mode	0 = Transparent (default) 1 = Manual
	0	Sa4 Generator Mode	0 = Transparent (default) 1 = Manual
4	Rx Path Config		See bit-field description below. Default = 0x00
	Bit	**Content**	**Description**
	7:5	Reserved	Set the bits to ‘0’ for future compatibility.
	5	TS 0 Generator Mode	0 = Disable (default) 1 = When Rx DSL is out of sync, generate TS0 FAS and MFAS
	4	TS 16 Generator Mode	0 = Transparent (default) 1 = Manual
	3	A-bit Generator Mode	0 = Transparent (default) 1 = Manual
	2:1	E-bit Generator Mode	00 = Transparent (default) 01 = Manual 10 = Automatic 11 = Reserved
	0	CRC-4 Generator Mode	0 = Transparent (default) 1 = Automatic (recalculate)
5	Rx Path Sa Config		See bit-field description below. Default = 0x00
	Bit	**Content**	**Description**

	Bit	Content	Description
	7:6	Reserved	Set the bits to '0' for future compatibility.
	5	Sa8 Generator Mode	0 = Transparent (default) 1 = Manual
	4	Sa7 Generator Mode	0 = Transparent (default) 1 = Manual
	3:2	Sa6 Generator Mode	00 = Transparent (default) 01 = Manual 10 = Reserved 11 = Reserved
	1	Sa5 Generator Mode	0 = Transparent (default) 1 = Manual
	0	Sa4 Generator Mode	0 = Transparent (default) 1 = Manual

15.16.2 E1 Framer Tx Path Generator Values

E1 Framer Tx Path Generator Values

This command programs the E1 Framer transmit path generator values. These are only applicable when the specified functionality is running in manual mode.

NOTE: These values are stored in the Host Port RAM. To reduce processor latency, the host should access the Host Port RAM rather then use the API.

Opcode C Constant	Opcode Value	Opcode Type	Incoming Bytes	Outgoing Bytes
_E1_PRA_TX_GEN_VALUES	0x45	Control	6 or 22	None

Incoming Parameters

Byte	Content		Description
1	Tx Path Value		See bit-field description below. Default = 0x00
	Bit	**Content**	**Description**
	7:3	Reserved	Set the bits to '0' for future compatibility.
	2	A-bit Generator Value	1-bit field specifying the manual A-Bit value
	1	E2-bit Generator Value	1-bit field specifying the manual E2-Bit value
	0	E1-bit Generator Value	1-bit field specifying the manual E1-Bit value
2	Sa4 Value		1-byte field specifying the manual Sa4 value

3	Sa5 Value	1-byte field specifying the manual Sa5 value
4	Sa6 Value	1-byte field specifying the manual Sa6 value
5	Sa7 Value	1-byte field specifying the manual Sa7 value
6	Sa8 Value	1-byte field specifying the manual Sa8 value
7-22	TS16 byte 0-15	16 bytes of TimeSlot 16 of Frames 0-15. These bytes should be sent only if the Tx TS 16 generator Mode is set to manual using E1_PRA_CONFIG.

15.16.3 E1 Framer Rx Path Generator Values

E1 Framer Rx Path Generator Values

This command programs the E1 Framer receive path generator values. These are only applicable when the specified functionality is running in manual mode.

NOTE: These values are stored in the Host Port RAM. To reduce processor latency, the host should access the Host Port RAM rather then use the API.

Opcode C Constant	Opcode Value	Opcode Type	Incoming Bytes	Outgoing Bytes
_E1_PRA_RX_GEN_VALUES	0x46	Control	6 or 22	None

Incoming Parameters

Byte	Content		Description
1	Rx Path Value		See bit-field description below. Default = 0x00
	Bit	**Content**	**Description**
	7:3	Reserved	Set the bits to '0' for future compatibility.
	2	A-bit Generator Value	1-bit field specifying the manual A-Bit value
	1	E2-bit Generator Value	1-bit field specifying the manual E2-Bit value
	0	E1-bit Generator Value	1-bit field specifying the manual E1-Bit value
2	Sa4 Value		1-byte field specifying the manual Sa4 value
3	Sa5 Value		1-byte field specifying the manual Sa5 value
4	Sa6 Value		1-byte field specifying the manual Sa6 value
5	Sa7 Value		1-byte field specifying the manual Sa7 value
6	Sa8 Value		1-byte field specifying the manual Sa8 value
7-22	TS16 byte 0-15		16 bytes of TimeSlot 16 of Frames 0-15. These bytes should be sent only if the Rx TS 16 generator Mode is set to manual using E1_PRA_CONFIG.

15.16.4 Inject E1 Framer CRC Error

Inject E1 Framer CRC Error				
Continuously inject CRC in all frames in the transmit or receive direction.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_E1_PRA_INJECT_CRC_ERROR	0x47	Control	2	None

Incoming Parameters

Byte	Content	Description		
1	Inject Tx CRC error options	See the Inject CRC Error Options table below.		
		Inject CRC Error Options		
		Option	**Description**	**Parameter**
		Off	Normal CRC value (Default Value).	0x00
		Continuous Error	Continuously inject CRC error in all E1 multiframes	0xFF
2	Inject Rx CRC error options	See the Inject CRC Error Options table below.		
		Inject CRC Error Options		
		Option	**Description**	**Parameter**
		Off	Normal CRC value (Default Value).	0x00
		Continuous Error	Continuously inject CRC error in all E1 multiframes.	0xFF

15.16.5 E1 Framer Transmit Code

E1 Framer Transmit Code				
Transmit framed AIS, unframed AUXP or normal payload in the Tx or Rx path.				
Opcode C Constant	**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_E1_PRA_TX_CODE	0x48	Control	1	None

Incoming Parameters

Byte	Content		Description
1	Code value		See bit-field description below. Default = 0x00
	Bit	**Content**	**Description**

	Bit	Content	Description
	7:4	Reserved	Set the bits to '0' for future compatibility.
	3:2	Tx Generator Data Mode	00 = Normal payload (default) 01 = Framed AIS 10 = Unframed AUXP 11 = Reserved
	1:0	Rx Generator Data Mode	00 = Normal payload (default) 01 = Framed AIS 10 = Unframed AUXP 11 = Reserved

15.16.6 E1 Framer Tx Path Monitor Change

E1 Framer Tx Path Monitor Change

This command indicates when an E1 Framer transmit path monitor values has changed.

NOTE: This command is available as an Unsolicited Interrupt.

Opcode C Constant	Opcode Value	Opcode Type	Incoming Bytes	Outgoing Bytes
_E1_PRA_TX_MON_CHANGE	0xC0	Status	1	2

Incoming Parameters

Byte	Content	Description
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters

Byte	Content		Description
1	Tx Path Change		See bit-field description below.
	Bit	**Content**	**Description**

	7	Sa8 Monitor Change	0 = No 1 = Yes
	6	Sa7 Monitor Change	0 = No 1 = Yes
	5	Sa6 Monitor Change	0 = No 1 = Yes
	4	Sa5 Monitor Change	0 = No 1 = Yes
	3	Sa4 Monitor Change	0 = No 1 = Yes
	2	A-bit Monitor Change	0 = No 1 = Yes
	1	E-bit Monitor Change	0 = No 1 = Yes
	0	MF Sync Status	0 = No 1 = Yes
2	Tx Ts16 CAS Indication		See bit-field description below.
	Bit	**Content**	**Description**
	7-1	Reserved	Set the bits to '0' for future compatibility.
	0	Tx Ts16 CAS Indication	1 = CAS Alignment has been detected 0 = CAS Alignment has not been detected

15.16.7 E1 Framer Rx Path Monitor Change

E1 Framer Rx Path Monitor Change				
This command indicates when an E1 Framer receive path monitor values has changed. ***NOTE:*** This command is available as an Unsolicited Interrupt.				
Opcode C Constant	**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_E1_PRA_RX_MON_CHANGE	0xC1	Status	1	2
Incoming Parameters				

Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters			
Byte	**Content**		**Description**
1	Rx Path Change		See bit-field description below.
	Bit	**Content**	**Description**
	7	Sa8 Monitor Change	0 = No 1 = Yes
	6	Sa7 Monitor Change	0 = No 1 = Yes
	5	Sa6 Monitor Change	0 = No 1 = Yes
	4	Sa5 Monitor Change	0 = No 1 = Yes
	3	Sa4 Monitor Change	0 = No 1 = Yes
	2	A-bit Monitor Change	0 = No 1 = Yes
	1	E-bit Monitor Change	0 = No 1 = Yes
	0	MF Sync Status	0 = No 1 = Yes
2	Rx Ts16 CAS Indication		See bit-field description below.
	Bit	**Content**	**Description**
	7-1	Reserved	
	0	Rx Ts16 CAS Indication	1 = CAS Alignment has been detected 0 = CAS Alignment has not been detected

15.16.8 E1 Framer Tx Path Monitor Values

E1 Framer Tx Path Monitor Values				
This command returns the E1 Framer transmit path monitor values. ***NOTE:*** These values are stored in the Host Port RAM. To reduce processor latency, the host should access the Host Port RAM rather then use the API.				
Opcode C Constant	**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_E1_PRA_TX_MON_VALUES	0xC2	Status	1	6 or 22

Incoming Parameters			
Byte	**Content**		**Description**
1	Reserved		Set to 0x00 for future compatibility.
Outgoing Parameters			
Byte	**Content**		**Description**
1	Tx Path Value		See bit-field description below
	Bit	**Content**	**Description**
	7:3	Reserved	Reserved
	2	A-bit Monitor Value	1-bit field indicating the manual A-Bit value
	1	E2-bit Monitor Value	1-bit field indicating the manual E2-Bit value
	0	E1-bit Monitor Value	1-bit field indicating the manual E1-Bit value
2	Sa4 Value		1-byte field indicating the manual Sa4 value
3	Sa5 Value		1-byte field indicating the manual Sa5 value
4	Sa6 Value		1-byte field indicating the manual Sa6 value
5	Sa7 Value		1-byte field indicating the manual Sa7 value
6	Sa8 Value		1-byte field indicating the manual Sa8 value
7-22	TS16 byte 0-15		16 bytes of TimeSlot 16 of Frames 0-15. These bytes are returned by the ZipWirePlus device only if the Tx TS 16 Read Enable is set to enable using E1_PRA_CONFIG API command.

15.16.9 E1 Framer Rx Path Monitor Values

E1 Framer Rx Path Monitor Values				
This command returns the E1 Framer receive path monitor values. ***NOTE:*** These values are stored in the Host Port RAM. To reduce processor latency, the host should access the Host Port RAM rather then use the API.				
Opcode C Constant	**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_E1_PRA_RX_MON_VALUES	0xC3	Status	1	6 or 22

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters			
Byte	Content		Description
1	Rx Path Value		See bit-field description below.
	Bit	Content	Description
	7:3	Reserved	Reserved
	2	A-bit Monitor Value	1-bit field indicating the manual A-Bit value
	1	E2-bit Monitor Value	1-bit field indicating the manual E2-Bit value
	0	E1-bit Monitor Value	1-bit field indicating the manual E1-Bit value
2	Sa4 Value		1-byte field indicating the manual Sa4 value
3	Sa5 Value		1-byte field indicating the manual Sa5 value
4	Sa6 Value		1-byte field indicating the manual Sa6 value
5	Sa7 Value		1-byte field indicating the manual Sa7 value
6	Sa8 Value		1-byte field indicating the manual Sa8 value
7-22	TS16 byte 0-15		16 bytes of TimeSlot 16 of Frames 0-15. These bytes are returned by the ZipWirePlus device only if the Rx TS 16 Read Enable is set to enable using E1_PRA_CONFIG API command.

15.16.10 E1 Framer Error Counters

E1 Framer Error Counters					
This command returns the E1 Framer transmit and receive path error counters. These are read/clear error counters.					
Opcode C Constant		Opcode Value	Opcode Type	Incoming Bytes	Outgoing Bytes
_E1_PRA_ERROR_CTRS		0xC4	Status	1	8
Incoming Parameters					
Byte	Content	Description			
1	Reserved	Set to 0x00 for future compatibility.			
Outgoing Parameters					
Byte	Content	Description			
1-2	Tx CRC-4	2-byte unsigned value specifying the transmit path CRC-4 errors. Low byte programmed first.			
3-4	Tx E-Bit	2-byte unsigned value specifying the transmit path E-bit errors. Low byte programmed first.			
5-6	Rx CRC-4	2-byte unsigned value specifying the receive path CRC-4 errors. Low byte programmed first.			
7-8	Rx E-Bit	2-byte unsigned value specifying the receive path E-bit errors. Low byte programmed first.			

15.16.11 E1 Framer Multi-frame Alignment

E1 Framer Multiframe Alignment				
This command returns the E1 Framer multiframe alignment status.				
Opcode C Constant	**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_E1_PRA_MF_STAT	0xC5	Status	1	None

Incoming Parameters

Byte	Content	Description
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters

Byte	Content		Description
1	Multiframe alignment		See bit-field description below. Default = 0x00
	Bit	**Content**	**Description**
	7:4	Reserved	
	3	Tx CAS Value	1 = Transmit path is in CAS multiframe alignment 0 = Transmit path is not in CAS multiframe alignment
	2	Rx CAS Value	1 = Receive path is in CAS multiframe alignment 0 = Receive path is not in CAS multiframe alignment
	1	Tx MF Value	1 = Transmit path is in CRC4 multiframe alignment 0 = Transmit path is not in CRC4 multiframe alignment
	0	Rx MF Value	1 = Receive path is in CRC4 multiframe alignment 0 = Receive path is not in CRC4 multiframe alignment

15.16.12 E1 Framer Alarm Status

E1 Framer Alarm Status				
This command returns the Status of the E1 Framer Alarms				
Opcode C Constant	**Opcode Value**	**Opcode Type**	**Incoming Bytes**	**Outgoing Bytes**
_E1_PRA_ALARM_STATUS	0xC6	Status	1	2

Byte	Content	Description
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters

Byte	Content		Description
1	Rx Alarm		See bit-field description below. Default = 0x00 means no alarm
	Bit	**Content**	**Description**
	7-6	Reserved	
	5	TS16AIS Alarm	1: Set 0: Reset
	4	LMFA Alarm	1: Set 0: Reset
	3	RAI Alarm	1: Set 0: Reset
	2	Y Alarm	1: Set 0: Reset
	1	AIS Alarm	1: Set (NOT SUPPORTED) 0: Reset
	0	RLOF Alarm	1: Set 0: Reset
2	Tx Alarm		See bit-field description below. Default = 0x00 means no alarm
	Bit	**Content**	**Description**
	7-6	Reserved	

	5	TS16AIS Alarm	1: Set 0: Reset
	4	LMFA Alarm	1: Set 0: Reset
	3	RAI Alarm	1: Set 0: Reset
	2	Y Alarm	1: Set 0: Reset
	1	AIS Alarm	1: Set (NOT SUPPORTED) 0: Reset
	0	RLOF Alarm	1: Set 0: Reset

15.17 Regenerator APIs

15.17.1 DSL G.hs Start Regenerator Silence

DSL_GHS_START_REGEN_SILENCE				
This command is issued to the SRU-C by the SRU Host, prior to activation, to start Regenerator Silence Period with the STU-R.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_GHS_START_REGEN_SILENCE	0x5C	Control	1	None
Incoming Parameters				
Byte	**Content**	**Description**		
1	Reserved	Set the bits to ‘0’ for future compatibility.		

15.17.2 DSL G.hs Stop Regenerator Silence

DSL_GHS_STOP_REGEN_SILENCE				
This command is issued to the SRU-C by the SRU Host to take the link between SRU-C and STU-R out of Regenerator Silence Period.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_GHS_STOP_REGEN_SILENCE	0x5B	Control	1	None

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set the bits to '0' for future compatibility.

15.17.3 DSL G.hs Regenerator Rate Override

DSL_GHS_REGEN_RATE_OVERRIDE				
This Command is used by the SRU Host to override the N and I rate settings in in the SRU-C. The new N and I settings are gathered by the SRU Host from the SRU-R, then transferred to the SRU-C prior re-activation while STU-R is in Regenerator Silence Period. The clocking mode of the SRU-R is also transferred to the SRU-C. SRU-C places down and up rates and clock mode in G.HS MS message to configure the STU-R. Please note N is the number of time slots and I is the number of i-bits.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_GHS_REGEN_RATE_OVERRIDE	0x5D	Control	5	None

Incoming Parameters		
Byte	**Content**	**Description**
1	Final down rate	Represented in N rate. Rate = N *64.
2	Final down sub-rate	Represented in I sub-rate. Sub-rate = I * 8.
3	Final up rate	Represented in N rate. Rate = N *64. Must be the same as down rate.
4	Final up sub-rate	Represented in I sub-rate. Sub-rate = I * 8. Must be the same as down sub-rate.
5	Clock mode	Clock mode: 0x00 = Plesiosynchronous. 0x01 = Plesiosynchronous with NTR. 0x02 = Synchronous

15.17.4 DSL G.hs Get Final Rate

DSL_GHS_GET_FINAL_RATE				
This Command is used by the SRU host to get the final rate achieved by the link between the STU-C and SRU-R. These rates will be used to program the SRU-C to STU-R link.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
DSL_GHS_GET_FINAL_RATE	0x81	Status	0	4

Outgoing Parameters		
Byte	**Content**	**Description**
1	Final down rate	Represented in N rate. Rate = N *64.
2	Final down sub-rate	Represented in I sub-rate. Sub-rate = I * 8.
3	Final up rate	Represented in N rate. Rate = N *64. Must be the same as down rate.
4	Final up sub-rate	Represented in I sub-rate. Sub-rate = I * 8. Must be the same as down sub-rate.

15.17.5 DSL G.hs Regenerator Diagnostics

_DSL_GHS_REGEN_DIAGNOSTIC					
This command is a control API and allows the SRU host processor to take an HTUR or SRU-R out of Regenerator Silence Period (RSP) and begin the training startup process for diagnostic purposes. To use this command the HTUR or SRU-R must have previously been set (via G.HS) to the RSP mode.					
C Constant		**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_GHS_REGEN_DIAGNOSTIC		0x5E	Control	1	None
Incoming Parameters					
Byte	**Content**	**Description**			
1	Reserved	Set the bits to ‘0’ for future compatibility.			

15.18 Loopbacks, Test Modes and Diagnostics API Commands

15.18.1 Loopbacks (1 of 3)

Loopbacks						
Operates the device in loopback modes (see Section 12.2). To turn off any of the loopback modes, use the loopback command with the _EXIT_LOOPBACK (0x00 value) parameter. The AFE Analog and Bit Pump Transmit Loopbacks are destructive to the current bit pump link, that is, they bring the link down because the ZipWirePlus system performs a mini-startup to adapt the DSP receiver section. The Normal Operation and Activation Failure bits in the DSL status should be used to determine the status of the loopback (similar concept to a normal startup). On exit of these loopbacks, the bit pump is initialized to a reset state and goes to the idle state, where it awaits further commands. A complete activation procedure should be repeated if normal operation is required. The other loopbacks can be issued (or exited) at any time during normal operation without affecting the bit pump link. Only throughput data is affected to match the desired loopback condition. The software automatically handles swapping scrambler and de-scrambler taps on entry and exit of certain loopbacks. All loopbacks can be issued when the ASM is disabled to facilitate development and debugging of other devices in the system. For example, the ZipWirePlus can be placed in the _FR_PCM_ON_PCM_LB to develop (and debug) code for the T1/E1 framer. The _EXIT_LOOPBACK parameter must be issued before issuing any other test mode, loopback, or end-to-end training; failure to comply will result in unpredictable behavior. When transitioning from activation (training) to a destructive loopback, the user must disable the Activation Request (ASM) and issue the Force Deactivate command.						
C Constant			**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_LOOPBACK			0x09	Control	1	None
Incoming Parameters						
Byte	**Content**	**Description**				
1	Loopback Option	See Section 12.2.3.				
		Loopback Options				
		Option	**Description**		**Parameter (C Constant)**	

		Exit Loopback	Cancel current loopback.	0x00 (_EXIT_LOOPBACK)
		Destructive to DSL Link		
		Transmitting Analog Loopback	The data is transmitted out the AFE line driver and looped back into the AFE hybrid input. The AFE Receiver inputs are bypassed.	0x01 (_AFE_HYBRID_LB)
		Silent Analog Loopback	The data is looped back internally before the AFE line driver into the AFE A/D converter. The AFE Receiver and Hybrid inputs are bypassed. The AFE Line Driver is disabled.	0x03 (_AFE_SILENT_LB)
		Bit Pump Transmit Loopback	The data is looped back internally before the Bit Pump to AFE interface back into the Bit Pump AFE serial inputs. The AFE line driver will still output the proper levels. The AFE Receiver and Hybrid inputs are bypassed.	0x05 (_BP_TX_LB)
		Not Destructive to DSL Link		
		Bit Pump Near Loopback (1)	The data is looped back internally before the Bit Pump DSP transmit section back into the DSL Framer DSL receive section. The DSL Framer scrambler and de-scrambler are set to use the same tap.	0x06 (_BP_DIGITAL_NEAR_LB)
		DSL Framer PCM on DSL Loopback (1)	The data is looped back internally before the DSL Framer DSL transmit section back into the DSP Framer Receive section. The DSL Framer scrambler and de-scrambler are set to use the same tap.	0x09 (_FR_PCM_ON_HDSL_LB)
		DSL Framer DSL on PCM Loopback (2)	The data, and sync signals are looped back internally before the DSL Framer PCM receive inputs back into the DSL Framer PCM transmit inputs. This API is not applicable to Multi=Pair Applications.	0x0A (_FR_HDSL_ON_PCM_LB)
		DSL Framer PCM on PCM Loopback (3)	The data, and sync signals are looped back internally from the DSL Framer PCM transmits inputs to the DSL Framer PCM receive inputs. This API is not applicable to Multi=Pair Applications.	0x0B (_FR_PCM_ON_PCM_LB)
		DSL Framer NB on NB Loopback (3)	The data, and sync signals are looped back internally from DSL Framer NB transmits inputs to the DSL Framer NB receive inputs.	0x0C (_FR_NB_ON_NB_LB)
		DSL Framer DSL on NB Loopback (2)	The data, and sync signals are looped back internally before the DSL Framer NB receive inputs back to the DSL Framer NB transmit inputs.	0x0D (_FR_HDSL_ON_NB_LB)
		ATM PHY Source Loopback	The ATM PHY transmit data is looped back before the DSL Framer interface back into the ATM PHY receive section.	0x0E (_ATM_SOURCE_LB)

NOTE: (1) The bit-pump digital near-end and DSL Framer PCM on DSL loopbacks will automatically be disabled (exited) whenever the modem trains.
(2) Exiting this loopback will exit both the DSL on PCM and the DSL on NB simultaneously.
(3) Exiting this loopback will exit both the PCM on PCM and the NB on NB simultaneously.

15.18.2 Test Modes (1 of 2)

Test Modes

Operates the device in special test modes. Executing any test mode will be destructive to the current DSL link (bring the link down). The _EXIT_TEST_MODE (value 0x00) parameter is used to disable the test mode in progress. The _EXIT_TEST_MODE parameter must be issued before issuing any other test mode, loopback, or end-to-end training; failure to comply will result in unpredictable behavior. When exiting any of these tests, the device is initialized to a reset state (IDLE state) where it awaits further commands.

When transitioning from activation (training) to a test mode, the user must disable the Activation Request (ASM) and issue the Force Deactivate command.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_TEST_MODE	0x0D	Control	1	None

Incoming Parameters

Byte	Content	Description
1	Test Mode Option	See Test Mode Option.

Test Mode Option

Option	Description	Parameter (C Constant)
Exit Test Mode	Cancel current test mode. (default)	0x00 (_EXIT_TEST_MODE)
Transmit Isolated Pulse	Transmit (repeatedly) an isolated pulse. This is useful for testing the transmit pulse template. Use the _DSL_TX_ISO_PULSE (Opcode: 0x16) API command to set the desired pulse code.	0x01 (_TM_TX_ISOLATED_PULSE)
Transmit G.hs C-Tone	Transmit a C-Tone (20 kHz) signal. This is useful for measuring the transmit power. Only applicable in the G.shdsl image.	0x02 (_TM_TX_ CTONE)
Transmit G.hs R-Tone	Transmit a R-Tone (12 kHz) signal. This is useful for measuring the transmit power. Only applicable in the G.shdsl image.	0x03 (_TM_TX_ RTONE)
Transmit Continuous Coded 32-Level	Transmit continuous coded 32-Level PAM scrambled 1s. This is useful for measuring PSD and transmit power.	0x05 (_TM_C32_LEVEL_SCR)

		Transmit Continuous Coded 16-Level	Transmit continuous coded 16-level PAM scrambled 1s. This is useful for measuring PSD and transmit power.	0x06 (_TM_C16_LEVEL_SCR)
		Transmit Continuous Coded 8-Level	Transmit continuous coded 8-Level PAM scrambled 1s. This is useful for measuring PSD and transmit power.	0x07 (_TM_C8_LEVEL_SCR)
		Transmit Continuous Coded 4-Level	Transmit continuous coded 4-Level PAM scrambled 1s. This is useful for measuring PSD and transmit power.	0x08 (_TM_C4_LEVEL_SCR)
		Transmit Continuous Uncoded 16-Level	Transmit continuous Uncoded 16-Level PAM scrambled 1s. This is useful for measuring PSD and transmit power.	0x09 (_TM_U16_LEVEL_SCR)
		Transmit Continuous Uncoded 8-Level	Transmit continuous Uncoded 8-Level PAM scrambled 1s. This is useful for measuring PSD and transmit power.	0x0A (_TM_U8_LEVEL_SCR)
		Transmit Continuous Uncoded 4-Level	Transmit continuous uncoded 4-Level PAM scrambled 1s. This is useful for measuring PSD and transmit power.	0x0B (_TM_U4_LEVEL_SCR)
		Transmit Continuous Uncoded 2-Level	Transmit continuous uncoded 2-Level PAM scrambled 1s. This is useful for measuring PSD and transmit power.	0x0C (_TM_U2_LEVEL_SCR)
		ERLE Test	Start the ERLE test with the current ERLE options. See Section 12.3 for complete details regarding ERLE.	0x10 (_TM_ERLE)
		LOSW DISABLE	After link is stable and in showtime. then this command should be issued. it will put ASM state machine into IDLE_STATE and link will stay up forever until EXIT_TEST_MODE or SYSTEM_ACTIVATE Command is issued.	0x11 (_TM_LOSW_DISABLE)

15.18.3 Transmit Isolated Pulses Test Bit Pump Mode

Bit Pump Transmit Isolated Pulses Test Mode				
This command selects the desired output level while in the Transmit Isolated Pulses test modes. The command is only required in 2B1Q mode				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_TX_ISO_PULSE	0x16	Control	1	None

Incoming Parameters		
Byte	**Content**	**Description**
1	Isolated Pulse Option	Set the Isolated Pulse level 0 = -3 pulse 1 = -1 pulse 2 = +3 pulse 3 = -1 pulse

15.18.4 ERLE Test Mode

ERLE Test Mode	C	R

This command activates the ERLE Test Mode. To abort the ERLE test mode before completion, use the _DSL_TEST_MODE (0x0D) API command with the _EXIT_TEST_MODE (0x00) parameter. See Section 12.3 for complete details regarding the ERLE procedure.

NOTE: The ERLE test is typically run with the AAGC set to 0.0 dB.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_BP_ERLE_TEST_MODE	0x18	Control	1	None

Incoming Parameters					
Byte	**Content**		**Description**		
1	ERLE Configuration		See bit-field description below.		
	Bit	**Content**	**Description**		
	7	Reserved	Reserved		
	6:4	AGAIN[2:0]	**AGAIN [2:0]**	**Decimal**	**Gain (dB)**
			000	0	0.0
			001	1	3.5
			010	2	6.0
			011	3	7.9
			100	4	10.0
			101	5	11.6
			110	6	13.3
			111	7	15.0
	3:0	Reserved	Reserved		

15.18.5 ERLE Results

ERLE Results				
This command queries for the ERLE Test Mode results.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_BP_ERLE_RESULTS	0x93	Status	1	16

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters		
Byte	**Content**	**Description**
1–4	NOISE	32-bit value specifying the background noise floor. The low byte is sent first.
5–8	SLM	32-bit value specifying the SLM value. The low byte is sent first.
9–12	FELM	32-bit value specifying the FELM value. The low byte is sent first.
13–16	SLM2	32-bit value specifying the SLM2 value. The low byte is sent first.

NOTE: 32-bit value = (byte 4 << 24) + (byte 3 << 16) + (byte 2 << 8) + (byte 1)

The Digital ERLE and Analog ERLE measurements are determined by the following formulas:

$$DERLE = 20 \times \log\left(\frac{SLM}{FELM}\right)$$

$$AERLE = 20 \times \log\left(\frac{SLM2}{SLM}\right)$$

15.18.6 Stage Number

Stage Number				
Queries for the stage number of the various state machines.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_STAGE_NUMBER	0x8F	Status	1	10

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility

Outgoing Parameters		
Byte	**Content**	**Description**
1	Reserved	Reserved
2	ASM Stage	1-byte unsigned integer field returning the Activation State Manager (ASM) stage number.
3	DSP Startup Stage	1-byte unsigned integer field returning the DSP startup stage number.
4	DSL Framer Stage	1-byte unsigned integer field returning the DSL Framer stage number.
5	DPLL Handler Stage	1-byte unsigned integer field returning the PCM DPLL Handler stage number.
6	Stuff Manager Stage	1-Byte unsigned integer field returning the DSL Framer Stuff Manage stage number.
7	G.hs Session Manager	1-byte unsigned integer field returning the G.hs Session Manager stage number.
8	G.hs Transaction Manager	1-byte unsigned integer field returning the G.hs Transaction Manager stage number.
9	Narrowband DPLL Handler Stage	1-byte unsigned integer field returning the Narrowband DPLL Handler stage number.
10	Narrowband Time Base Manager Stage	1-byte unsigned integer field returning the Narrowband Time Base Manager stage number.

15.18.7 Clear ZipWirePlus Error Counters

Clear ZipWirePlus Error Counters				
Clears the requested ZipWirePlus Error counters to 0. This command can either clear all of the error counters or clear individual error counter blocks.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_CLEAR_ERROR_CTRS	0x40	Control	1	None

Incoming Parameters		
Byte	**Content**	**Description**
1	Clear Error Counter Option	See the Clear Error Counter Options table below.
		NOTE: The Clear All option does not clear the system performance error counters.

Clear Error Counter Options		
Option	**Description**	**Parameter (C Constant)**
Clear All	Clears all error counters.	0x00 (_CLEAR_ALL_COUNTERS)
Clear Operational	Clears the ZipWirePlus operational error counter.	0x01 (_CLEAR_OPER_ERR_CTRS)
Clear DSL Performance	Clears the ZipWirePlus DSL Performance error counter.	0x02 (_CLEAR_HDSL_ERR_CTRS)

	Clear PCM Performance	Clears the ZipWirePlus PCM Performance error counter.	0x03 (_CLEAR_PCM_ERR_CTRS)
	Clear System Performance	Clears the ZipWirePlus System Performance error counter. This is not cleared when the Clear All option is selected.	0x04 (_CLEAR_SYSTEM_ERR_CTRS)
	Clear Error History	Clears the ZipWirePlus Error History counters.	0x05 (_CLEAR_ERROR_HISTORY)
	Clear ATM Operational	Clears the ATM PHY operational error counters.	0x06 (_CLEAR_ATM_OPER_ERR_CTRS)
	Clear ATM Performance	Clears the ATM PHY performance error counters.	0x07 (_CLEAR_ATM_PERF_ERR_CTRS)
	Clear ATM Cell	Clears the ATM PHY cell counters.	0x08 (_CLEAR_ATM_CELL_CTRS)

15.18.8 Read ZipWirePlus Operational Error Counters

Read ZipWirePlus Operational Error Counters				
Queries the ZipWirePlus Operational Error counters. These error counters are accumulated when the ZipWirePlus device reaches normal operation or when the Clear Error Counter command was issued. The operational error counters are 1-byte wide. These counters are primarily used during system debug and development and are typically not required during normal operation.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_OPER_ERR_CTRS	0x9C	Status	1	23

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters		
Byte	**Content**	**Description**
1	RPFIFO Full	8-bit value specifying the number of Rx PCM FIFO Full errors.
2	RPFIFO Empty	8-bit value specifying the number of Rx PCM FIFO Empty errors.
3	RPFIFO Slip	8-bit value specifying the number of Rx PCM FIFO Slip errors.
4	TPFIFO Full	8-bit value specifying the number of Tx PCM FIFO Full errors.
5	TPFIFO Empty	8-bit value specifying the number of Tx PCM FIFO Empty errors.
6	TPFIFO Slip	8-bit value specifying the number of Tx PCM FIFO Slip errors.
7	Transmit Stuff	8-bit value specifying the number of Tx Stuff errors.
8	PCM DPLL	8-bit value specifying the number of PCM DPLL errors.
9	TPFIFO Water Level	8-bit value specifying the number of Tx PCM FIFO Water Level errors.

10	RPFIFO Water Level	8-bit value specifying the number of Rx PCM FIFO Water Level errors.
11	RPSLIP Positive	8-bit value specifying the number of Rx PCM Slip Buffer positive errors.
12	RPSLIP Negative	8-bit value specifying the number of Rx PCM Slip Buffer negative errors.
13	RNBFIFO Full	8-bit value specifying the number of Rx NB FIFO Full errors.
14	RNBFIFO Empty	8-bit value specifying the number of Rx NB FIFO Empty errors.
15	RNBFIFO Slip	8-bit value specifying the number of Rx NB FIFO Slip errors.
16	TNBFIFO Full	8-bit value specifying the number of Tx NB FIFO Full errors.
17	TNBFIFO Empty	8-bit value specifying the number of Tx NB FIFO Empty errors.
18	TNBFIFO Slip	8-bit value specifying the number of Tx NB FIFO Slip errors.
19	NB DPLL	8-bit value specifying the number of NB DPLL errors.
20	TNBFIFO Water Level	8-bit value specifying the number of Tx NB FIFO Water Level errors.
21	RNBFIFO Water Level	8-bit value specifying the number of Rx NB FIFO Water Level errors.
22	RNBSLIP Positive	8-bit value specifying the number of Rx NB Slip Buffer positive errors.
23	RNBSLIP Negative	8-bit value specifying the number of Rx NB Slip Buffer negative errors.

15.18.9 Read ATM PHY Operational Error Counters

Read ATM PHY Operational Error Counters				
Queries the ATM PHY Operational Error Counters. These error counters are accumulated from when the ZipWirePlus device reaches normal operation or when the Clear Error Counter command was issued. The ATM PHY performance error counters are 1-byte wide. These error counters are primarily used during system debug and development and are typically not required during normal operation.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_ATM_PHY_OPER_ERR_CTRS	0xB8	Status	1	5

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters		
Byte	**Content**	**Description**
1	Parity Error	8-bit unsigned value specifying the number of parity errors.
2	SOC Error	8-bit unsigned value specifying the number of Start of Cell errors.
3	Transmit FIFO Full	8-bit unsigned value specifying the number of transmit FIFO overflow errors.
4	Receive FIFO Full	8-bit unsigned value specifying the number of receive FIFO overflow errors.
5	Bus Conflict	8-bit unsigned value specifying the number of Bus Conflict errors.

15.18.10 Inject DSL CRC Error

Inject DSL CRC Error				
Inject a CRC error in the next N number of DSL frames or continuously inject CRC in all frames. All six CRC bits are inverted based on the calculated CRC values. The user can issue the Inject CRC Error OFF option before all N frames are completed.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_INJECT_CRC_ERROR	0x41	Control	1	None

Incoming Parameters				
Byte	**Content**	**Description**		
1	Inject CRC error options	See the Inject CRC Error Options table below.		
		Inject CRC Error Options		
		Option	**Description**	**Parameter (C Constant)**
		Off	Normal CRC value (Default Value).	0x00 (_INJECT_CRC_OFF)
		Continuous Error	Continuously inject CRC error in all DSL frames.	0xFF (_INJECT_CRC_CONT)
		Inject N Errors	Inject CRC error in next N number of DSL frames. A value of 1 equals 1 frame.	1–254

15.18.11 Inject ATM PHY HEC Error

Inject ATM PHY HEC Error				
Inject a HEC error in the next ATM cell. This command is used to verify the Corrected and Uncorrected HEC Error counters. ***NOTE:*** The M28945 has a non-conformance that causes the HEC error to only to work ~42% of the time.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_ATM_PHY_INJECT_HEC_ERROR	0x1F	Control	1	None

Incoming Parameters		
Byte	**Content**	**Description**
1	HEC Error Pattern	The HEC error pattern (ERRPAT) parameter is XORed with the calculated HEC byte. Setting a single bit in the error pattern (such as 0x01) will cause a single-bit HEC error. Setting multiple bits in the error pattern (such as 0xFF) will cause a multi-bit HEC error.

15.18.12 Data Bank Contents

Data Bank Contents					
Sets the three Data Bank patterns. Data Bank 1 fills unused PCM and DSL time slots in the PCM and DSL mappers. The default value is 0xFF, which provides an AIS (all 1s) code for these unused time slots. The AIS pattern conforms to the DSL standards.					
C Constant		**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_DBANK		0x27	Control	3	None
Incoming Parameters					
Byte	**Content**	**Description**			
1	Data Bank 1	1-byte field, least significant bit is sent first. The default is 0xFF.			
2	Data Bank 2	1-byte field, least significant bit is sent first. The default is 0x00.			
3	Data Bank 3	1-byte field, least significant bit is sent first. The default is 0x55.			

15.18.13 DSL Framer Read Minimum/Maximum Water Levels

DSL Framer Read Min/Max Water Levels					
This command returns the PCM and Narrow Band minimum and maximum water levels. This is useful in debugging custom mapping applications. The water level return value is expressed in the number of bytes where a 1 implies 1-byte.					
C Constant		**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_READ_MIN_MAX_WL		0x87	Status	1	8
Incoming Parameters					
Byte	**Content**	**Description**			
1	Reserved	Set to 0x00 for future compatibility.			
Outgoing Parameters					
Byte	**Content**	**Description**			
1	Tx PCM Max	1-byte specifying the transmit PCM maximum water level.			
2	Tx PCM Min	1-byte specifying the transmit PCM minimum water level.			
3	Rx PCM Max	1-byte specifying the receive PCM maximum water level.			
4	Rx PCM Min	1-byte specifying the receive PCM minimum water level.			
5	Tx NB Max	1-byte specifying the transmit NB maximum water level.			
6	Tx NB Min	1-byte specifying the transmit NB minimum water level.			
7	Rx NB Max	1-byte specifying the receive NB maximum water level.			
8	Rx NB Min	1-byte specifying the receive NB minimum water level.			

15.18.14 Diagnostics Get Configuration

Diagnostics Get Configuration				
The ZipWireplus upon receiving this command returns the information for the Control APIs in the order described below. The 62 byte information will be returned by the ZipWirePlus Processor in the binary format in the Data Section of the Message Structure defined in Section 13.3.4				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_API_DIAG_GET_CONFIG	0xAA	Control	1	Upto 75

Incoming Parameters

Byte	Content	Description
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters

Byte	Content	Description
1	_DSL_AFE_CONFIG	Please refer to Control API format for more details.
2-3	_DSL_TRAINING_MODE	Please refer to Control API format for more details.
4-6	_DSL_CLOCK_CONFIG	Please refer to Control API format for more details.
7-8	_DSL_PCM_MF_LEN	Please refer to Control API format for more details.
9	_DSL_SYSTEM_CONFIG	Please refer to Control API format for more details.
10-11	_DSL_DATA_RATE	Please refer to Control API format for more details.
12-23	_DSL_PREACTIVATION_CFG	Please refer to Control API format for more details.
24	_DSL_FR_PCM_CONFIG	Please refer to Control API format for more details.
25	_DSL_FR_HDSL_CONFIG	Please refer to Control API format for more details.
26	_DSL_PCM_CLK_CONF	Please refer to Control API format for more details.
27	_AFE_TX_GAIN	Please refer to Control API format for more details.
28	_DSL_MULTI_PAIR_CONFIG	Please refer to Control API format for more details.
29-34	_DSL_NB_MULTI_RATE	Please refer to Control API format for more details.
35-42	_DSL_MULTI_RATE	Please refer to Control API format for more details.
43	_ATM_PHY_MODE	Please refer to Control API format for more details.
44	_ATM_PHY_UTOPIA_CONFIG	Please refer to Control API format for more details.
45-46	_ATM_PHY_IF_MODE	Please refer to Control API format for more details.
47-48	_ATM_PHY_CONFIGURE	Please refer to Control API format for more details.
49-52	_DSL_NB_CONFIG	Please refer to Control API format for more details.
53-57	_DSL_PCM_WATER_LEVEL	Please refer to Control API format for more details.

58-62	_DSL_NB_WATER_LEVEL	Please refer to Control API format for more details.

15.18.15 Diagnostics Get Status

Diagnostics Get Status					
The ZipWireplus upon receiving this command returns the information for the Status APIs in the order described below. The 56 byte information will be returned by the ZipWirePlus Processor in the binary format in the Data Section of the Message Structure defined in Section 13.3.4.					
C Constant		**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_API_DIAG_GET_STATUS		0xAB	Status	1	Upto 75
Incoming Parameters					
Byte	**Content**	**Description**			
1	Reserved	Set to 0x00 for future compatibility.			
Outgoing Parameters					
Byte	**Content**	**Description**			
1	_DSL_FAR_END_ATTEN	Please refer to Status API format for more details.			
2	_DSL_NOISE_MARGIN	Please refer to Status API format for more details.			
3-10	_DSL_STATUS	Please refer to Status API format for more details.			
11-18	_DSL_READ_MIN_MAX_WL	Please refer to Status API format for more details.			
19-29	_DSL_VERSIONS	Please refer to Status API format for more details.			
30-39	_DSL_STAGE_NUMBER	Please refer to Status API format for more details.			
40	_DSL_AFE_SETTING	Please refer to Status API format for more details.			
41-56	_DSL_TIME	Please refer to Status API format for more details.			

15.18.16 Diagnostics Get Performance

Diagnostics Get Performance				
The ZipWireplus upon receiving this command returns the information for the Performance APIs in the order described below. The 55 byte information will be returned by the ZipWirePlus Processor in the binary format in the Data Section of the Message Structure defined in Section 13.3.4 .				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_API_DIAG_GET_PERF	0xAC	Status	1	Upto 75

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.
Outgoing Parameters		
Byte	**Content**	**Description**
1-23	DSL_OPER_ERR_CTRS	Please refer to Performance API format for more details.
24-27	_DSL_SYSTEM_PERF_ERR_CTRS	Please refer to Performance API format for more details.
28-32	_ATM_PHY_OPER_ERR_CTRS	Please refer to Performance API format for more details.
33-38	_ATM_PHY_PERF_ERR_CTRS	Please refer to Performance API format for more details.
39-55	_ATM_PHY_CELL_CTRS	Please refer to Performance API format for more details.

15.19 Miscellaneous API Commands

15.19.1 DSL DSP Configure

DSL DSP Configure

This command is a control API that modifies the DSP configuration.

The Equalizer Training mode allows the user to select which 2B1Q training mode to use. This option maybe useful for interoperability when training with non-Mindspeed modem.

The low delay mode option in G.SHDSL mode, will meet the end to end delay requirement at the expense of performance degradation. - refer to G.SHDSL spec 11.5 page 81.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_DSP_CONFIG	0x5F	Control	1	None

Incoming Parameters			
Byte	**Content**		**Description**
1	Configuration		See bit-field description below. Default = 0x00
	7-5	Reserved	
	4	CL CLR clock mode	0 – (default) Set all three mode when it send CL (CO side) or CLR (RT side) regardless of API _DSL_CLOCK_CONFIG (0x04) (Plesiosynchronous / Plesiosynchronous with timing reference/ Synchronous) 1 – Set only one clock mode when it send CL (CO side) or CLR (RT side) based on API _DSL_CLOCK_CONFIG (0x04)
	3	Skip Galf Detection	0 – (default) Configure as normal 1 – Skip Galf detection when interfaced to INF.
	2	Forced Mode Select	0 – (default) Configure as normal 1 - Force mode select to be HTUR when interfaced with INF (This is valid only when unit is RT)

	1	Equalizer training mode (only available in 2B1Q mode)	0 – (default) data driven 1 - decision driven
	0	Low Delay Mode (only available in G.shDSL mode)	0 – (default) 8975 G.SHDSL violates end to end delay requirement. G.S spec requires 1.25ms delay for data rates below 1.5Mbps. 1 – Meets the delay requirement, but reduces performance.

15.20 BER Meter Configuration and Status API Commands

15.20.1 PRBS Configure

PRBS Configure				
Configure the PRBS generator to one of the selected patterns. The Transmit PCM and Receive PCM share a common PRBS generator. This command is only necessary when either the Transmit or Receive PCM BER Configure data is sourced from the PRBS generator. Issuing this command with the same data pattern parameter value will force a reset on the PRBS polynomial.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_PRBS_CONFIGURE	0x25	Control	1	None

Incoming Parameters					
Byte	**Content**		**Description**		
1	PRBS Configuration		See bit-field description below. Default is 0x08.		
	Bit	**Content**	**Description**		
	7:6	BER Scale	BER Test Interval (or number of bits) where bit errors are accumulated. 00 = 2^{31} Bits. 01 = 2^{28} Bits. 10 = 2^{25} Bits. 11 = 2^{21} Bits.		
	5	PRBS Invert	0 = PRBS Data Normal (not inverted) 1 = PRBS Data Inverted		
	4	PRBS Source	0 = Enable random pattern, data generated from *PRBS Data Pattern* table below. 1 = Enable fixed pattern, data pattern specified using the CONST_FILL (Opcode 0x26) API command		
	3:0	PRBS Data Pattern	Set the PRBS Data Pattern as listed in *PRBS Data Pattern* table below.		
			PRBS Data Pattern		
			Data Pattern	**Description**	**Parameter (C Constant)**
			All 0s (SPACE)	Outputs an all-0s pattern.	0x00 (_DSL_PRBS_ZERO)

			All 1s (MARK)	Outputs an all-1s pattern.	0x01 (_DSL_PRBS_ONE)
			1:1	Alternating 0s and 1s.	0x02 (_DSL_PRBS_1_1)
			2^6–1	Repeats every 2^6–1 (63) bits. The polynomial is $x^6 + x^5 + 1$.	0x03 (_DSL_PRBS_2_6)
			2^9–1	Repeats every 2^9–1 (511) bits. The polynomial is $x^9 + x^5 + 1$.	0x04 (_DSL_PRBS_2_9)
			2^{11}–1	Repeats every 2^{11}–1 (2047) bits. The polynomial is $x^{11} + x^9 + 1$.	0x05 (_DSL_PRBS_2_11)
			2^{15}–1	Repeats every 2^{15}–1 bits. The polynomial is $x^{15} + x^{14} + 1$.	0x06 (_DSL_PRBS_2_15)
			2^{23}–1	Repeats every 2^{23}–1 bits. The polynomial is $x^{23} + x^{18} + 1$.	0x08 (_DSL_PRBS_2_23)
		QRSS Removed	**NOTE:** All-0s, All-1s, and alternating 0s and 1s set the PRBS source to the fixed pattern option and program the Fill Pattern to the appropriate value (see *Fill Pattern (CONST_FILL* command below).		

15.20.2 Fill Pattern (CONST_FILL)

Fill Pattern (CONST_FILL)				
Sets the Fill Pattern (Constant Pattern) used when the Transmit or Receive PCM BER Configure data is sourced from the Constant Pattern. The Constant Pattern is a useful debugging tool because the PRBS can generate a known data pattern in a given time slot to debug sync versus data alignment problems. **NOTE:** Writing the fill pattern while the device is in the PRBS mode will corrupt the PRBS pattern.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_CONST_FILL	0x26	Control	1	None

Incoming Parameters		
Byte	**Content**	**Description**
1	Fill Pattern	1-byte field, least significant bit is sent first. The default is 0x55.

15.20.3 Transmit PCM BER State

Transmit PCM BER State				
Enable or Disable the DSL Framer Transmit PCM BER meter. The Transmit and Receive PCM BER meters can be operated independently.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_TP_BER_STATE	0x23	Control	1	None

Incoming Parameters		
Byte	**Content**	**Description**
1	BER State	0 = Disable the Transmit PCM BER Meter. All results are unmodified so they can be still read (Default Value). 1 = Enable the Transmit PCM BER meter. All results are reset to 0x00. Issuing the enable option forces the BER meter to re-perform the BER sync qualification period. The error counter and elapsed time counters are reset to 0x00. The enable option can be used as a BER meter reset.

15.20.4 Transmit PCM BER Meter Results

Transmit PCM BER Meter Results				
Requests the Transmit PCM BER Meter Results. Reading the BER Meter status commands while the BER Meter is enabled does not effect the BER meter operation.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_TP_BER_RESULTS	0x8C	Status	1	5

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters				
Byte	**Content**	**Description**		
1	BER Status	1-byte value indicating the BER Meter Status (see the *DSL Framer BER Status Bits* table below).		
		DSL Framer BER Status Bits		
		Status Bit	**Description**	**Bit Definition**
		7–4	Reserved	Reserved
		3–2	Measurement Phase Status	00 = IDLE 01 = Complete 10 = Failed 11 = In Progress

		1–0	Qualification Phase Status	00 = IDLE 01 = Complete 10 = Failed 11 = In Progress
2–3	Number of Bit Errors	16-bit value specifying the number of bit-errors. The low byte is sent first followed by the high byte.		
4–5	Elapsed Time	16-bit value specifying the elapsed time in seconds. The low byte is sent first followed by the high byte.		

NOTE: 16-bit value = (high byte << 8) + (low byte).

The following formulas are used to calculate the Average BER:

$$AvgBER = \frac{\#BitErrors}{\#BitsProcessed}$$

When the DSL Framer is complete, use:

$$\#BitsProcessed = BERScale$$

When the DSL Framer BER is in progress, use:

$$\#BitsProcessed = ElapsedTime\ x\ DataRate\ x\ \frac{\#MappedBERBitsPerFrame}{\#BitsPerFrame}$$

15.20.5 Receive PCM BER State

Receive PCM BER State

Enable or disable the DSL Framer Receive PCM BER meter. The Transmit and Receive PCM BER meters can be operated independently.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_RP_BER_STATE	0x24	Control	1	None

Incoming Parameters

Byte	Content	Description
1	BER State	0 = Disable the Receive PCM BER meter. All results are unmodified so they can still be read (Default Value). 1 = Enable the Receive PCM BER meter. All results are reset to 0x00. Issuing the enable option forces the BER meter to re-perform the BER sync qualification period. The error counter and elapsed time counters are reset to 0x00. The enable option can be used as a BER meter reset.

15.20.6 Receive PCM BER Meter Results

Receive PCM BER Meter Results						
Requests the Receive PCM BER Meter status. Reading the BER Meter status commands while the BER Meter is enabled does not effect the BER meter operation.						
C Constant			Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_RP_BER_RESULTS			0x8D	Status	1	5

Incoming Parameters		
Byte	Content	Description
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters				
Byte	Content	Description		
1	BER Status	1-byte value indicating the BER Meter Status (see the *DSL Framer BER Status Bits* table below).		
		DSL Framer BER Status Bits		
		Status Bit	Description	Bit Definition
		7–4	Reserved	Reserved
		3–2	Measurement Phase Status	00 = IDLE 01 = Complete 10 = Failed 11 = In Progress
		1–0	Qualification Phase Status	00 = IDLE 01 = Complete 10 = Failed 11 = In Progress
2–3	Number of Bit Errors	16-bit value specifying the number of bit errors. The low byte is sent first followed by the high byte.		
4–5	Elapsed Time	16-bit value specifying the elapsed time in seconds. The low byte is sent first followed by the high byte.		

NOTE: 16-bit value = (high byte << 8) + (low byte).

The following formulas are used to calculate the Average BER:

$$AvgBER = \frac{\# BitErrors}{\# BitsProcessed}$$

When the DSL Framer is complete, use:

$$\# BitsProcessed = BERScale$$

When the DSL Framer BER is in progress, use:

$$\# BitsProcessed = ElapsedTime\ x\ DataRate\ x\ \frac{\# MappedBERBitsPerFrame}{\# BitsPerFrame}$$

15.20.7 Transmit NB BER State

Transmit NB BER State				
Enable or Disable the DSL Framer Transmit NB BER meter. The Transmit and Receive NB BER meters can be operated independently.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_TNB_BER_STATE	0x21	Control	1	None

Incoming Parameters		
Byte	**Content**	**Description**
1	BER State	0 = Disable the Transmit NB BER Meter. All results are unmodified so they can be still read (Default Value). 1 = Enable the Transmit NB BER meter. All results are reset to 0x00. Issuing the enable option forces the BER meter to re-perform the BER sync qualification period. The error counter and elapsed time counters are reset to 0x00. The enable option can be used as a BER meter reset.

15.20.8 Transmit NB BER Meter Results

DSL Framer Transmit PCM NB Meter Results				
Requests the Transmit Narrowband BER Meter Status. Reading the BER Meter status commands while the BER Meter is enabled does not effect the BER meter operation.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_TNB_BER_RESULTS	0x91	Status	1	5

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters

Byte	Content	Description		
1	BER Status	1-byte value indicating the BER Meter Status (see the *DSL Framer BER Status Bits* table below).		
		DSL Framer BER Status Bits		
		Status Bit	**Description**	**Bit Definition**
		7–4	Reserved	Reserved
		3–2	Measurement Phase Status	00 = IDLE 01 = Complete 10 = Failed 11 = In Progress
		1–0	Qualification Phase Status	00 = IDLE 01 = Complete 10 = Failed 11 = In Progress
2–3	Number of Bit Errors	16-bit value specifying the number of bit-errors. The low byte is sent first followed by the high byte.		
4–5	Elapsed Time	16-bit value specifying the elapsed time in seconds. The low byte is sent first followed by the high byte.		

NOTE: 16-bit value = (high byte << 8) + (low byte).

The following formulas are used to calculate the Average BER:

$$AvgBER = \frac{\#BitErrors}{\#BitsProcessed}$$

When the DSL Framer is complete, use:

$$\#BitsProcessed = BERScale$$

When the DSL Framer BER is in progress, use:

$$\#BitsProcessed = ElapsedTime\ x\ DataRate\ x\ \frac{\#MappedBERBitsPerFrame}{\#BitsPerFrame}$$

15.20.9 Receive NB BER State

Receive NB BER State

Enable or disable the Receive NB BER meter. The Transmit and Receive NB BER meters can be operated independently.

C Constant	Opcode	Type	Incoming Bytes	Outgoing Bytes
_DSL_RNB_BER_STATE	0x22	Control	1	None

Incoming Parameters		
Byte	**Content**	**Description**
1	BER State	0 = Disable the Receive NB BER meter. All results are unmodified so they can still be read (Default Value). 1 = Enable the Receive NB BER meter. All results are reset to 0x00. Issuing the enable option forces the BER meter to re-perform the BER sync qualification period. The error counter and elapsed time counters are reset to 0x00. The enable option can be used as a BER meter reset.

15.20.10 Receive NB BER Meter Results

DSL Framer Receive NB BER Meter Results				
Requests the Receive Narrowband BER Meter status. Reading the BER Meter status commands while the BER Meter is enabled does not effect the BER meter operation.				
C Constant	**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_RNB_BER_RESULTS	0x92	Status	1	5

Incoming Parameters		
Byte	**Content**	**Description**
1	Reserved	Set to 0x00 for future compatibility.

Outgoing Parameters				
Byte	**Content**	**Description**		
1	BER Status	1-byte value indicating the BER Meter Status (see the *DSL Framer BER Status Bits* table below).		
		DSL Framer BER Status Bits		
		Status Bit	**Description**	**Bit Definition**
		7–4	Reserved	Reserved
		3–2	Measurement Phase Status	00 = IDLE 01 = Complete 10 = Failed 11 = In Progress
		1–0	Qualification Phase Status	00 = IDLE 01 = Complete 10 = Failed 11 = In Progress
2–3	Number of Bit Errors	16-bit value specifying the number of bit errors. The low byte is sent first followed by the high byte.		
4–5	Elapsed Time	16-bit value specifying the elapsed time in seconds. The low byte is sent first followed by the high byte.		

NOTE: 16-bit value = (high byte << 8) + (low byte).
The following formulas are used to calculate the Average BER: $AvgBER = \frac{\#BitErrors}{\#BitsProcessed}$ When the DSL Framer is complete, use: $\#BitsProcessed = BERScale$ When the DSL Framer BER is in progress, use: $\#BitsProcessed = ElapsedTime\ x\ DataRate\ x\ \frac{\#MappedBERBitsPerFrame}{\#BitsPerFrame}$

15.21 ZipWirePlus Register Read/Write API Commands

15.21.1 Write Register

Write Register					
This command writes the specified block of data to the specified address.					
C Constant		**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_WRITE_REG		0x75	Control	3 + block size (length)	None
Incoming Parameters					
Byte	**Content**	**Description**			
1–2	Address	2-byte value specifying the address. Low byte is programmed first.			
3	Length (L)	1-byte specifying the block size up to 64 bytes. A length of 0 corresponds to one byte and a 63 corresponds to 64 bytes.			
4–(4 + L)	Data	Block of data. The first byte is written to the specified address, the second byte to the address + 1, etc.			
NOTE: Address = (high << 8) + low.					

15.21.2 Read Register

Read Register						
This command reads the specified block of data from the specified address.						
C Constant			**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_READ_REG			0xA0	Status	3	Block size (length)
Incoming Parameters						
Byte	**Content**	**Description**				
1–2	Address	2-byte value specifying the address. Low byte is programmed first.				
3	Length (L)	1-byte specifying the block size up to 64 bytes. A length of 0 corresponds to one byte and a 63 corresponds to 64 bytes.				
Outgoing Parameters						
Byte	**Content**	**Description**				
1–L	Data	Block of data. The first byte corresponds to the specified address, the second byte to the address + 1, etc.				

15.21.3 Write AFE Register

Write AFE Register						
This command writes the specified block of data to the specified address of the AFE device.						
C Constant			**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_WRITE_AFE			0x76	Control	2 + block size (length)	None
Incoming Parameters						
Byte	**Content**	**Description**				
1	Address	1-byte value specifying the address. The AFE only has 128 registers.				
2	Length (L)	1-byte specifying the block size up to 64 bytes. A length of 0 corresponds to one byte and a 63 corresponds to 64 bytes.				
3–(3 + L)	Data	Block of data. The first byte is written to the specified address, the second byte to the address + 1, etc.				

15.21.4 Read AFE Register

Read AFE Register					
This command reads the specified block of data from the specified address of the AFE device.					
C Constant		**Opcode**	**Type**	**Incoming Bytes**	**Outgoing Bytes**
_DSL_READ_AFE		0xA1	status	2	Block size (length)
Incoming Parameters					
Byte	**Content**	**Description**			
1	Address	1-byte value specifying the address. The AFE only has 128 registers.			
2	Length (L)	1-byte specifying the block size up to 64 bytes. A length of 0 corresponds to one byte and a 63 corresponds to 64 bytes.			
Outgoing Parameters					
Byte	**Content**	**Description**			
1–L	Data	Block of data. The first byte corresponds to the specified address, the second byte to the address + 1, etc.			

16.0 Appendix A: DSL Frame Structure

The ZipWirePlus chipset supports the following DSL frame structures.

- G.shdsl
- HDSL2
- HDSL1
- IDSL
- Transparent (no DSL overhead) - used in legacy SDSL applications

16.1 G.shdsl Frame Structure

Table 16-1 summarizes the SHDSL frame structure.

The size of each payload block is defined as *k*, where *k* = 12 x (*i* + *n* x 8) [bits].

The rates for single pair are given by nX64 + iX8 kbits/s. For 16-TCPAM, 3 ≤ n ≤ 60 and 0 ≤ i ≤ 7. For 16-TCPAM and *n*=60, the applicable value of *i* is 0. This corresponds to (payload) data rates from 192 kbit/s to 3840 kbit/s in increments of 8 kbit/s for 16-TCPAM. For 32-TCPAM, 12 ≤ *n* ≤ 89 and 0 ≤ *i* ≤ 7. For 32-

TCPAM and *n*=89, the applicable value of *i* is 0. This corresponds to (payload) data rates from 768 kbit/s to 5696 kbit/s in increments of 8 kbit/s for 32-TCPAM.

The DSL data rate is set by: *PCM payload* + 8 Kbps, where the 8 Kbps is the fixed DSL overhead.

In the optional 4-wire mode, two separate PMS-TC sub-layers are active–one for each wire pair. In this case, the above formula represents the payload data rate for each pair rather than the aggregate payload rate. Each pair shall operate at the same payload rate, and the transmitters for both pairs shall maintain frame alignment within specified limits.

In the STU-C, the symbol clocks for each pair shall be derived from a common source. The maximum differential delay between the start of STU-C frames shall be no greater than four symbols at the line side of each SHDSL transmitter. In the STU-R, symbol clocks may be derived from loop timing on each pair, so these clocks shall be locked in frequency but shall have an arbitrary phase relationship. The maximum differential delay between the start of STU-R frames shall be no greater than six symbols at the line side of each SHDSL transmitter.

Table 16-1 SHDSL Frame Structure

Frame Bit #	Over-head Bit #	Name	Description
1-14	1-14	sw1-sw14	Frame Sync Word
15	15	fbit1 / losd	Fixed Indicator bit #1 (Loss of Signal)
16	16	fbit2 / sega	Fixed Indicator bit #2 (Segment Anomaly)
17 -> k + 16	—	b1	Payload block #1
k + 17	17	eoc01	EOC bit #1
k + 18	18	eoc02	EOC bit #2
k + 19	19	eoc03	EOC bit #3

Frame Bit #	Over-head Bit #	Name	Description
k + 20	20	eoc04	EOC bit #4
k + 21	21	crc1	Cyclic Redundancy Check #1
k + 22	22	crc2	Cyclic Redundancy Check #2
k + 23	23	fbit3 / ps	Fixed Indicator bit #3 (Power Status)
k + 24	24	sbid1	Stuff bit ID #1
k + 25	25	eoc05	EOC bit #5
k + 26	26	eoc06	EOC bit #6
k + 27 -> 2 k + 26	—	b2	Payload block #2
2 k + 27	27	eoc07	EOC bit #7
2 k + 28	28	eoc08	EOC bit #8
2 k + 29	29	eoc09	EOC bit #9
2 k + 30	30	eoc10	EOC bit #10
2 k + 31	31	crc3	Cyclic Redundancy Check #3
2 k + 32	32	crc4	Cyclic Redundancy Check #4
2 k + 33	33	fbit4 / segd	Fixed Indicator bit #4 (Segment Defect)
2 k + 34	34	eoc11	EOC bit #11
2 k + 35	35	eoc12	EOC bit #12
2 k + 36	36	sbid2	Stuff bit ID #2
2 k + 37 –> 3 k + 36	—	b3	Payload block #3
3 k + 37	37	eoc13	EOC bit #13
3 k + 38	38	eoc14	EOC bit #14
3 k + 39	39	eoc15	EOC bit #15
3 k + 40	40	eoc16	EOC bit #16
3 k + 41	41	crc5	Cyclic Redundancy Check #5
3 k + 42	42	crc6	Cyclic Redundancy Check #6
3 k + 43	43	eoc17	EOC bit #17
3 k + 44	44	eoc18	EOC bit #18
3 k + 45	45	eoc19	EOC bit #19
3 k + 46	46	eoc20	EOC bit #20
3 k + 47 –> 4 k + 46	—	b4	Payload block #4
4 k + 47	47	stb1	Stuff bit #1
4 k + 48	48	stb2	Stuff bit #2

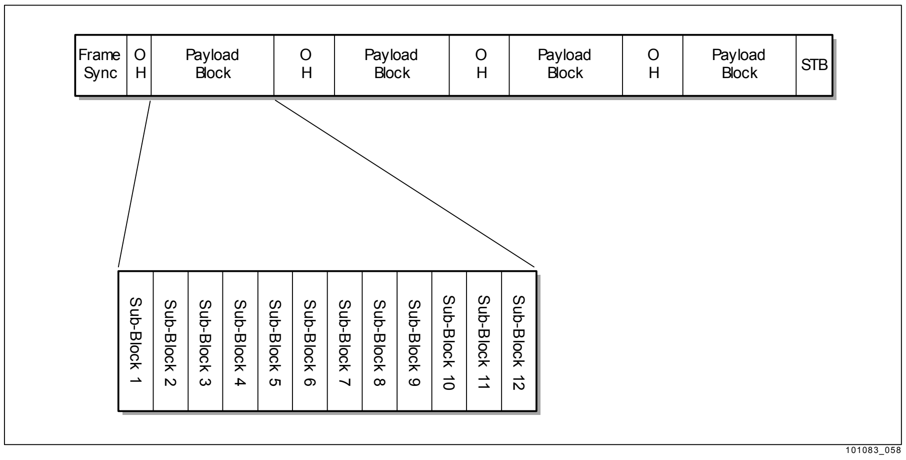
Frame Bit #	Over-head Bit #	Name	Description
4 k + 49	49	stb3	Stuff Bit #3
4 k + 50	50	stb4	Stuff Bit #4

16.1.1 Payload Block Data Structure

Each payload block shall consist of 12 Sub-blocks, and shown in Figure 16-1.

The size of each Payload Sub-Block is defined as k_S, where k_S = i + n x 8 [bits]. As stated in Section 16.1, the payload data rate is set by: n x 64 + i x 8 Kbps, where 3 ≤ n ≤ 36 and 0 ≤ i ≤ 7.

Figure 16-1 Structure Of Payload Blocks



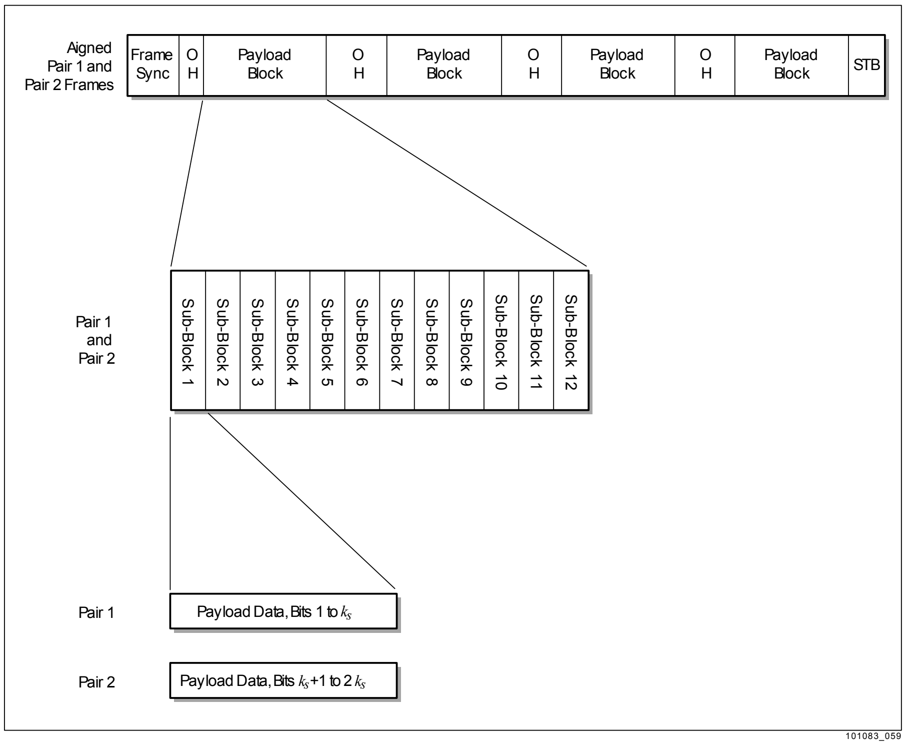
16.1.2 Data Interleaving in 4-Wire Mode

In the optional 4-wire mode, interleaving of payload data between pairs is necessary. This shall be accomplished by interleaving within payload sub-blocks between pair 1 and pair 2. k_S bits in each sub-block shall be carried on pair 1, and an additional k_S bits shall be carried on pair 2, as illustrated in Figure 16-2.

The size of each payload sub-block is defined as $2k_S$, where k_S = i + n x 8 [bits].

As stated in Section 16.1, the payload data rate per pair is set by: n x 64 + i x 8 Kbps, where 3 ≤ n ≤ 36 and 0 ≤ i ≤ 7.

Figure 16-2 Data Interleaving Within Payload Blocks



16.2 HDSL2 Configurations

The system supports several kinds of configurations for T1/E1 transmission using HDSL2 technology. Table 16-2 shows the basic structure of an HDSL2 frame where each frame is nominally 6 ms in length and consists of 48 payload blocks. Each payload block contains a single F- or Z-bit, plus an application-specific number of payload bytes. Groups of 12 payload blocks are concatenated and separated by an ordered set of HDSL2 overhead bits, where a 10-bit SYNC word pattern identifies the starting location of the HDSL2 frame. Fifty overhead bits are defined in one HDSL2 frame, but the last four STUFF bits are nominally present in alternate frames. One frame contains an average of 48 overhead bits. Figure 16-3illustrates the frame structure. The following subsections show the payload block structure for different applications.

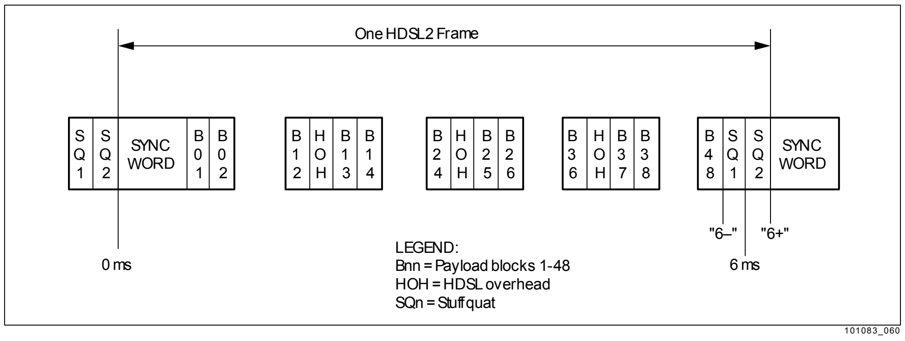
Table 16-2 HDSL2 Frame Structure

Frame Bit #	HOH Bit #	Symbol	Full Name
1–10	1–10	SW1–SW10	SYNC Word
11–2326	—	B01–B12	Payload Blocks 1–12
2327–2328	11–12	CRC1–CRC2	Cyclic Redundancy Check
2329	13	SBID1	Stuff Bit ID Copy 1
2330	14	LOSD	DS1 Loss of Signal Detect
2331–2338	1522	EOC01–EOC08	EOC Bit 1–8
2339–4654	—	B13–B24	Payload Blocks 13–24
4655–4656	23–24	CRC3–CRC4	Cyclic Redundancy Check

Frame Bit #	HOH Bit #	Symbol	Full Name
4657	25	UIB	Unspecified Indicator Bit
4658	26	SEGA	Segment Anomaly (same as FEBE)
4659–4666	27–34	EOC09–EOC16	EOC Bit 9–16
4667–6982	—	B25–B36	Payload Blocks 25–36
6983–6984	35–36	CRC5–CRC6	Cyclic Redundancy Check
6985	37	SBID2	Stuff Bit ID Copy 2
6986	38	SEGD	Segment Detect
6987–6994	39–46	EOC17–EOC24	EOC Bit 17–24
6995–9310	—	B37–B48	Payload Blocks 37–48
9311	47	SB1	Stuff Bit 1
9312	48	SB2	Stuff Bit 2
9313	49	SB3	Stuff Bit 3
9314	50	SB4	Stuff Bit 4

NOTE: The Frame Bit # field is based on the 1T1 (1552 kbps) application. Other data rates will have a different number of frame bits.

Figure 16-3 HDSL2 Frame Structure



16.2.1 HDSL2_1T1

HDSL2_1T1 runs the standard 1-loop T1 mapping at 1552 kbps with 1 loop carrying all the payloads from T1. Figure 16-4 illustrates each payload block contains 1 F-bit followed by 24 payload bytes. The relation between the payload bytes and PCM time slot is shown in

Table 16-3.

Figure 16-4 Payload Block Structure For 1T1 Application

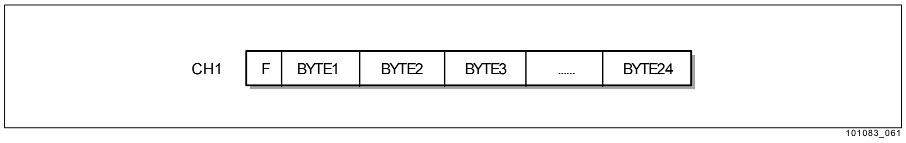


Table 16-3 1T1 Framing

Byte	1	2	3	4	5	6	7	8	9	10	11	12
Time slot	1	2	3	4	5	6	7	8	9	10	11	12
Byte	13	14	15	16	17	18	19	20	21	22	23	24
Time slot	13	14	15	16	17	18	19	20	21	22	23	24

16.3 HDSL1 Configurations

The system supports several kinds of configurations for T1/E1 transmission using DSL technology. Table 16-4 shows the basic structure of an DSL frame in which each frame is nominally 6 ms in length and consists of 48 payload blocks. Each payload block contains a single F or Z bit, plus an application specific number of payload bytes. Groups of 12 payload blocks are concatenated and separated by an ordered set of DSL overhead bits, where a 14-bit SYNC word pattern identifies the starting location of the DSL frame. Fifty overhead bits are defined in one DSL frame, but the last four STUFF bits are nominally present in alternate frames. Therefore, one frame contains an average of 48 overhead bits.

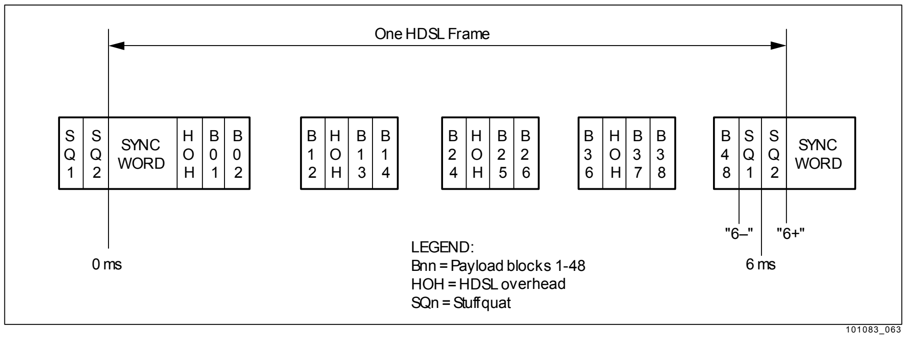
Figure 16-5 illustrates the frame structure. The payload block structures for different applications are shown in the following subsections.

Table 16-4 HDSL1 Frame Structure and Overhead Bit Allocation

HOH Bit	Symbol	Bit Name
1–14	SW1–SW14	SYNC Word
15	LOSD	Loss of signal
16	FEBE	Far End Block Error
Payload Blocks 1–12		
17–20	EOC1–EOC4	Embedded Operation Channel
21–22	CRC1–CRC2	Cyclic Redundancy Check
23	PS1	HTU-R Power Status
24	PS2	Power Status Bit 2
25	BPV	Bipolar Violation
26	EOC5	Embedded Operation Channel

HOH Bit	Symbol	Bit Name
Payload Blocks 13–24		
27–30	EOC6–EOC9	Embedded Operation Channel
31–32	CRC3–CRC4	Cyclic Redundancy Check
33	HRP	DSL Repeater Present
34	RRBE	Repeater Remote Block Error
35	RCBE	Repeater Central Block Error
36	REGA	Repeater Alarm
Payload Blocks 25–36		
37–40	EOC10–EOC13	Embedded Operation Channel
41–42	CRC5–CRC6	Cyclic Redundancy Check
43	RTA	Remote Terminal Alarm
44	RTR	Ready to Receive
45	UIB	Unspecified Indicator Bit
46	UIB	Unspecified Indicator Bit
Payload Blocks 37–48		
47	SQ1	Stuff Quat Sign
48	SQ2	Stuff Quat Magnitude
49	SQ3	Stuff Quat Sign
50	SQ4	Stuff Quat Magnitude

Figure 16-5 HDSL1 Frame Structure



16.3.1 HDSL1_1T1

This runs the standard 1-loop T1 mapping at 1552 kbps, with 1 loop carrying all the payloads from T1. Figure 16-6 illustrates each payload block contains one F-bit, followed by 24 payload bytes. The relation between the payload bytes and PCM time slot is shown in Table 16-5

Figure 16-6 Payload Block Structure for 1T1 Application

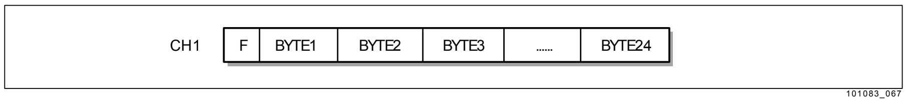


Table 16-5 1T1 Framing

Byte	1	2	3	4	5	6	7	8	9	10	11	12
Time slot	1	2	3	4	5	6	7	8	9	10	11	12
Byte	13	14	15	16	17	18	19	20	21	22	23	24
Time slot	13	14	15	16	17	18	19	20	21	22	23	24

16.3.2 HDSL1 _2T1

This runs the standard 2-loop T1 mapping at 784 kbps, with each loop carrying one-half the payloads from T1. Figure 16-7 illustrates each payload block contains one F-bit, followed by 12 payload bytes. The relation between the payload bytes and PCM time slot is shown in Table 16-6.

Figure 16-7 Payload Block Structure for 2T1 Application

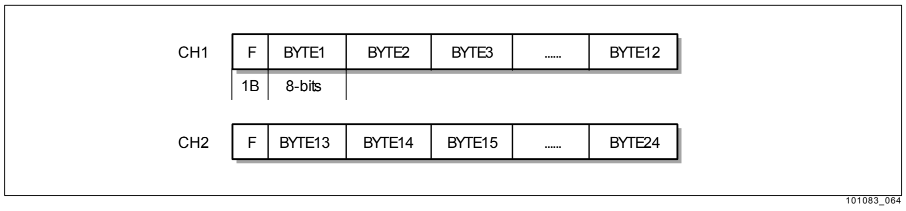


Table 16-6 2T1 Framing

Channel 1												
Byte	1	2	3	4	5	6	7	8	9	10	11	12
Time slot	1	2	3	4	5	6	7	8	9	10	11	12
Channel 2												
Byte	13	14	15	16	17	18	19	20	21	22	23	24
Time slot	13	14	15	16	17	18	19	20	21	22	23	24

16.3.3 HDSL1_1E1

This runs the standard 1-loop E1 mapping at 2320 kbps, with 1 loop carrying all the payloads from E1. Figure 16-8 illustrates each payload block contains one Z-bit, followed by 36 payload bytes. The relation between the payload bytes and PCM time slot is shown in Table 16-7

Figure 16-8 Payload Block Structure For 1E1 Application

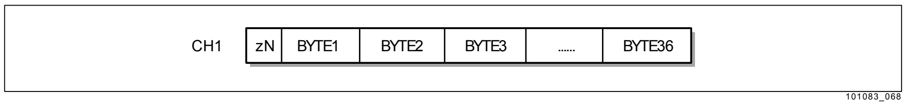


Table 16-7 1E1 Framing

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Time slot	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Byte	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Time slot	18	19	20	21	22	23	24	25	26	27	28	29	30	31	*	*	*	*

NOTE: * = DBANK

16.3.4 HDSL1_2E1

This runs the standard 2-loop E1 mapping at 1168 kbps, with each loop carrying one-half the payloads from E1. Figure 16-9 illustrates each payload block contains one Z-bit, followed by 18 payload bytes. The relation between the payload bytes and PCM time slot is shown in

Table 16-8.

Figure 16-9 Payload Block Structure For 2E1 Application

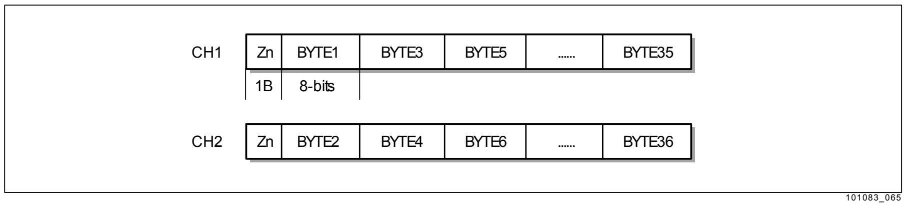


Table 16-8 2E1 Framing

Channel 1																		
Byte	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35
Time slot	0	1	3	5	7	9	11	13	15	16	18	20	22	24	26	28	30	DBANK
Channel 2																		
Byte	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36
Time slot	0	2	4	6	8	10	12	14	16	17	19	21	23	25	27	29	31	DBANK

16.3.5 HDSL1_3E1

This runs the standard 3-loop E1 mapping at 784 kbps, with each loop carrying one-third the payloads from E1. Figure 16-10illustrates each payload block contains one Z-bit, followed by 12 payload bytes. The relation between the payload bytes and PCM time slot is shown in Table 16-9.

Figure 16-10 Payload Block Structure For 3E1 Application

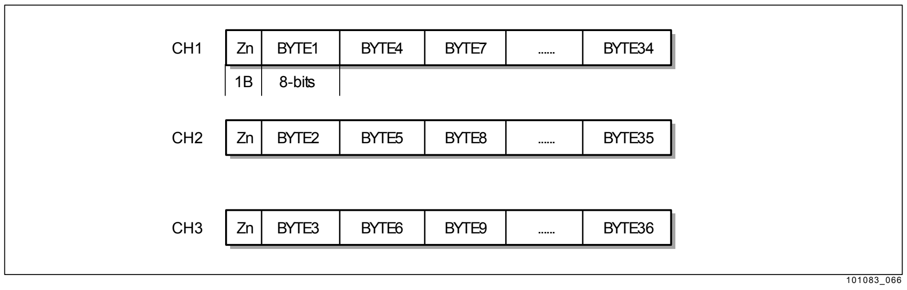


Table 16-9 3E1 Framing

Channel 1												
Byte	1	4	7	10	13	16	19	22	25	28	31	34
Time slot	0	1	4	7	10	13	16	17	20	23	26	29
Channel 2												
Byte	2	5	8	11	14	17	20	23	26	29	32	35
Time slot	0	2	5	8	11	14	16	18	21	24	27	30
Channel 3												
Byte	3	6	9	12	15	18	21	24	27	30	33	36
Time slot	0	3	6	9	12	15	16	19	21	25	28	31

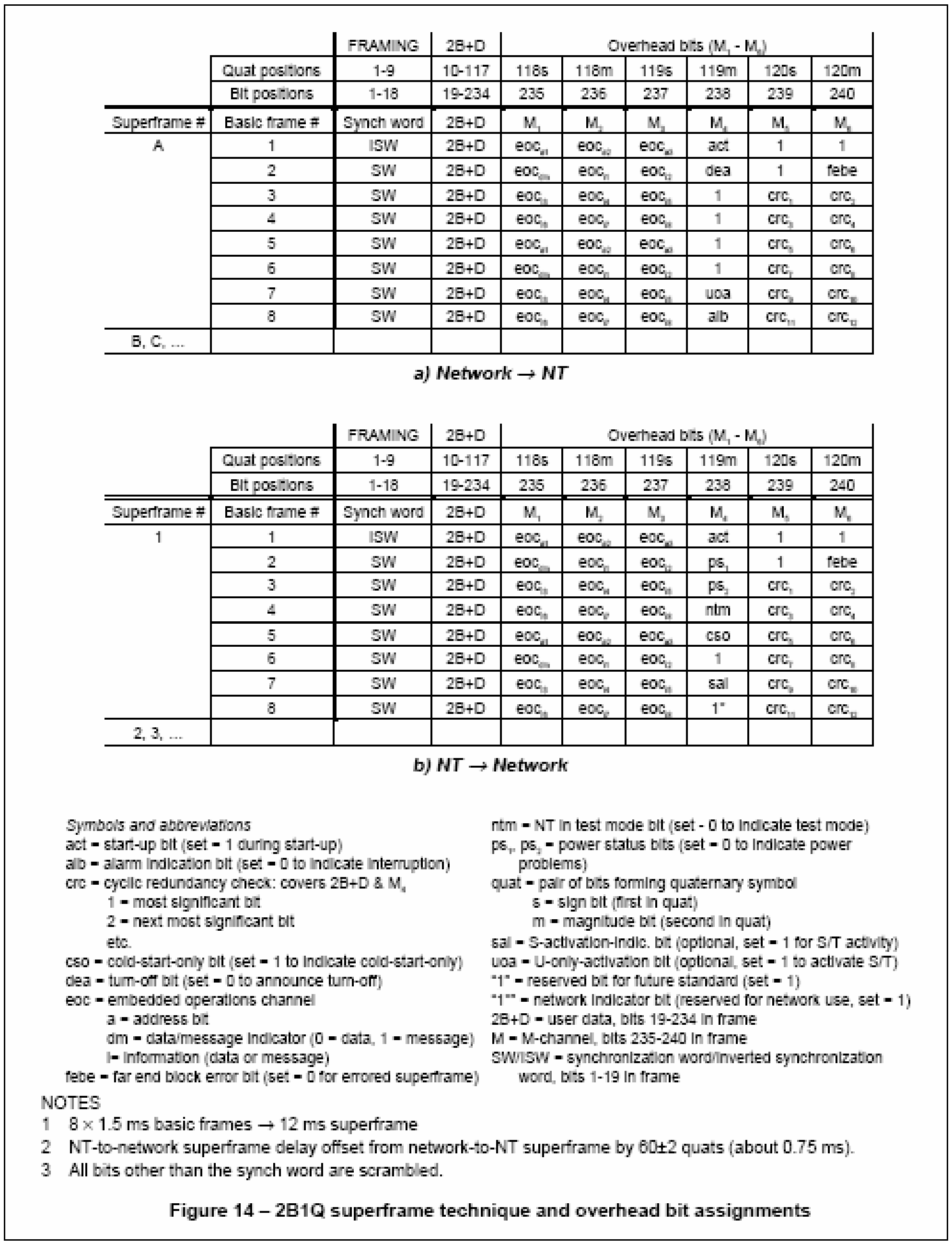
16.3.6 HDSL1_DSL_CUSTOM

To customize the DSL Framer code for applications other than standard 1T1, 2T1, 1E1, 2E1, and 3E1, the DSL Framer code needs to be modified.

16.4 IDSL Configurations

The system supports the IDSL NT configuration. Figure 16-11 shows the basic structure of an DSL frame in which each frame is nominally 1.5 ms in length and consists of 240 bits.

A superframe consists of 8 basic frames where each basic frame is nominally 1.5 ms in length. The start of a basic frame is identified by a 18-bit synchronization word (SW). In addition to the SW, the basic frame consists of 216 2B+D user data bits and 6 overhead bits including indicator bits, CRC and EOC bits. PCM rate can be set to 64Kbps (B), 128Kbps (2B) or 144Kbps (2B+D).



		FRAMING	2B+D	Overhead bits (M_1 - M_6)					
	Quat positions	1-9	10-117	118s	118m	119s	119m	120s	120m
	Bit positions	1-18	19-234	235	236	237	238	239	240
Superframe #	Basic frame #	Synch word	2B+D	M_1	M_2	M_3	M_4	M_5	M_6
A	1	ISW	2B+D	eoc_{a1}	eoc_{a2}	eoc_{a3}	act	1	1
	2	SW	2B+D	eoc_{dm}	eoc_{i1}	eoc_{i2}	dea	1	febe
	3	SW	2B+D	eoc_{i3}	eoc_{i4}	eoc_{i5}	1	crc_1	crc_2
	4	SW	2B+D	eoc_{i6}	eoc_{i7}	eoc_{i8}	1	crc_3	crc_4
	5	SW	2B+D	eoc_{a1}	eoc_{a2}	eoc_{a3}	1	crc_5	crc_6
	6	SW	2B+D	eoc_{dm}	eoc_{i1}	eoc_{i2}	1	crc_7	crc_8
	7	SW	2B+D	eoc_{i3}	eoc_{i4}	eoc_{i5}	uoa	crc_9	crc_{10}
	8	SW	2B+D	eoc_{i6}	eoc_{i7}	eoc_{i8}	aib	crc_{11}	crc_{12}
B, C, ...									

a) Network → NT

		FRAMING	2B+D	Overhead bits (M_1 - M_6)					
	Quat positions	1-9	10-117	118s	118m	119s	119m	120s	120m
	Bit positions	1-18	19-234	235	236	237	238	239	240
Superframe #	Basic frame #	Synch word	2B+D	M_1	M_2	M_3	M_4	M_5	M_6
1	1	ISW	2B+D	eoc_{a1}	eoc_{a2}	eoc_{a3}	act	1	1
	2	SW	2B+D	eoc_{dm}	eoc_{i1}	eoc_{i2}	ps_1	1	febe
	3	SW	2B+D	eoc_{i3}	eoc_{i4}	eoc_{i5}	ps_2	crc_1	crc_2
	4	SW	2B+D	eoc_{i6}	eoc_{i7}	eoc_{i8}	ntm	crc_3	crc_4
	5	SW	2B+D	eoc_{a1}	eoc_{a2}	eoc_{a3}	cso	crc_5	crc_6
	6	SW	2B+D	eoc_{dm}	eoc_{i1}	eoc_{i2}	1	crc_7	crc_8
	7	SW	2B+D	eoc_{i3}	eoc_{i4}	eoc_{i5}	sai	crc_9	crc_{10}
	8	SW	2B+D	eoc_{i6}	eoc_{i7}	eoc_{i8}	1*	crc_{11}	crc_{12}
2, 3, ...									

b) NT → Network

Symbols and abbreviations
act = start-up bit (set = 1 during start-up)
aib = alarm indication bit (set = 0 to indicate interruption)
crc = cyclic redundancy check: covers 2B+D & M_4
    1 = most significant bit
    2 = next most significant bit
    etc.
cso = cold-start-only bit (set = 1 to indicate cold-start-only)
dea = turn-off bit (set = 0 to announce turn-off)
eoc = embedded operations channel
    a = address bit
    dm = data/message indicator (0 = data, 1 = message)
    i= information (data or message)
febe = far end block error bit (set = 0 for errored superframe)
ntm = NT in test mode bit (set - 0 to indicate test mode)
ps_1, ps_2 = power status bits (set = 0 to indicate power problems)
quat = pair of bits forming quaternary symbol
    s = sign bit (first in quat)
    m = magnitude bit (second in quat)
sai = S-activation-indic. bit (optional, set = 1 for S/T activity)
uoa = U-only-activation bit (optional, set = 1 to activate S/T)
"1" = reserved bit for future standard (set = 1)
"1*" = network indicator bit (reserved for network use, set = 1)
2B+D = user data, bits 19-234 in frame
M = M-channel, bits 235-240 in frame
SW/ISW = synchronization word/inverted synchronization word, bits 1-19 in frame

NOTES
1 8 × 1.5 ms basic frames → 12 ms superframe
2 NT-to-network superframe delay offset from network-to-NT superframe by 60±2 quats (about 0.75 ms).
3 All bits other than the synch word are scrambled.

Figure 14 – 2B1Q superframe technique and overhead bit assignments

Figure 16-11 IDSL Frame Structure

17.0 Appendix B: G.Shdsl Transmit PSD Masks

17.1 Annex A

17.1.1 Symmetric

Payload Data Rate R (Kb/s)	K_{shdsl}	P_{shdsl}
$R < 1536$	7.86	$P1(R) \leq P_{shdsl} \leq 13.5$
$R = 1536$ or $R = 1544$	8.32	13.5
$R > 1544$	7.86	13.5

$$P1(R) = 0.3486 \cdot \log_2(1000 \cdot R + 8000) + 6.06 \text{ (dBm)}$$

17.1.2 Asymmetric

Payload Data Rate R (Kb/s)	Terminal Type	P_{shdsl}
$R = 1536$ or $R = 1544$	HTUC	16.8
	HTUR	16.5
$R = 768$ or $R = 776$	HTUC	14.1
	HTUR	14.1

17.2 Annex B

17.2.1 Symmetric

Payload Data Rate R (Kb/s)	K_{shdsl}	P_{shdsl}
$R < 2048$	7.86	$P1(R) \le P_{shdsl} \le 13.5$
$R \ge 2048$	9.90	14.5

$$P1(R) = 0.3486 \cdot \log_2(1000 \cdot R + 8000) + 6.06\,(\mathrm{dBm})$$

17.2.2 Asymmetric

Payload Data Rate R (Kb/s)	Terminal Type	K_{shdsl}	P_{shdsl}
$R = 2048$	HTUC	16.86	16.25
	HTUR	15.66	16.50
$R = 2304$	HTUC	12.48	14.75
	HTUR	11.74	15.25

NOTE: The transmit power in data state should be $P_{shdsl} \pm 0.5dBm$

18.0 Appendix C: Acronyms and Abbreviations

ADC (A/D)	Analog-to-Digital Converter
AFE	Analog Front End
AIS	Alarm Indication Signal
API	Application Programming Interface
BER	Bit Error Rate
BGA	Ball Grid Array
BP	Bit Pump
BT	Bit Pump Transceiver
Channel Unit	HDSL Framer (name comes from HDSL1 Framer)
CRC-N	Cyclic Redundancy Check-N
CU	Channel Unit or HDSL Framer
DAC (D/A)	Digital-to-Analog Converter
DFE	Decision Feedback Equalizer
DIP	Dual In-Line Package
Downstream	From the HTU-C towards the HTU-R (includes regenerators)
DPLL	Digital Phase Lock Loop
DSL	Digital Subscriber Line
DSL Framer	ZipWirePlus DSL Framer Block
DSP	Digital Signal Processing
EC	Echo Canceller
EOC	Embedded Operations Channel
EVM	Evaluation Module
FEBE	Far End Block Error (the far end reported a CRC error)
FEXT	Far End Cross Talk
FFE	Feed Forward Equalizer
FIFO	First-In First-Out

ADC (A/D)	Analog-to-Digital Converter
FR	Framer
H2TU	HDSL2 Terminal Unit
HDLC	High-Level Data Link Controller
HDSL	High-Bit-Rate Digital Subscriber Line
HTU	HDSL Terminal Unit
HTU-C or COT or LTU	Central Office Terminal or Local Terminal Unit
HTU-R or RT or NTU	Remote Terminal or Network Terminal Unit
IDSL	ISDN Digital Subscriber Line
LED	Light Emitting Diode
LOS	Loss of Signal
NEXT	Near End Cross Talk
OOF	Out of Frame
P2MP	Point to Multipoint
PAM	Pulse Amplitude Modulation
PCM	Pulse Code Modulation
PLL	Phase Lock Loop
PRA	Primary Rate Access
PRBS	Pseudo-Random Bit Sequence
RAM	Random Access Memory
ROM	Read Only Memory
TCM	Time Code Modulation
TQFP	Thin Quad Flat Pack
Transceiver	ZipWirePlus DSP/Transceiver Block
UART	Universal Asynchronous Receive Transmit
UIP	User Interface Program
Upstream	From the HTU-R towards the HTU-C (includes regenerators)