

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *RadiSys ENP-3511-F* at our website: *Click HERE*



ENP Software Development Kit Programmer's Guide

VERSION
2.0

www.radisys.com
World Headquarters
5445 NE Dawson Creek Drive • Hillsboro, OR
97124 USA
Phone: 503-615-1100 • Fax: 503-615-1121
Toll-Free: 800-950-0044
International Headquarters
Gebouw Flevopoort • Televisieweg 1A
NL-1322 AC • Almere, The Netherlands
Phone: 31 36 5365595 • Fax: 31 36 5365620
007-01271-0001
March 2002

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

March 2002
Copyright ©2002 by RadiSys Corporation.
All rights reserved.

EPC, INtime, iRMX, and RadiSys are registered trademarks of RadiSys Corporation. ASM, Brahma, DAI, DAQ, MultiPro, SAIB, Spirit, and ValuePro are trademarks of RadiSys Corporation.

DAVID, MAUI, OS-9, and OS-9000, are registered trademarks of RadiSys Microware Communications Software Division, Inc. FasTrak, Hawk, SoftStax, and UpLink are trademarks of RadiSys Microware Communications Software Division, Inc.

† All other trademarks, registered trademarks, service marks, and trade names are the property of their respective owners.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Before you begin

This guide explains how to install and use the ENP (Embedded Network Processor) SDK (Software Development Kit).

The ENP SDK provides programming and runtime libraries for the Intel† Internet Exchange Architecture (IXA) SDK that adds software functionality specific to RadiSys platforms. You use the ENP SDK to quickly develop optimized applications for the Intel IXP1200 network processor.

This guide assumes you understand Embedded Network Processors and the VxWorks operating system. It also assumes you are familiar with the IXA and Wind River Tornado development environments.

About this guide

Contents

Part I: Introducing the ENP SDK

This part acquaints you with the ENP Software Development Kit: its components, how they're put together, and how to develop applications for RadiSys ENP boards.

Chapter	Description
1 Overview	Describes ENP software development kit features and tells you where to find detailed information about ENP SDK topics.
2 Installation and configuration	Explains how to install and configure your ENP software development kit.
3 Developing ENP applications	Overviews typical application architectures suitable for use on ENP boards.
4 Sample applications	Describes the sample applications included with the ENP SDK.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Part II: Application programming interfaces

This part details the various API libraries included in the ENP Software Development Kit, including syntax, parameter descriptions, and return values.

Chapter	Description
5 System Monitor software	Describes the monitor software for the ENP platform, which provides a rudimentary execution environment.
6 Boot Manager software	Describes the Boot Manager application, which specifies the boot OS from those available in the ENP board's flash chip.
7 PCI communications	Describes the PCI communications library.
8 TDM routing and switching library	Describes the TDM switching library, which encapsulates and abstracts call setup and teardown for ARM code.
9 E1T1 library	Describes the E1/T1 library.
10 Interrupt library	Describes the Interrupt library.
11 ENP microcode resource manager	Describes the resource manager functions.
12 IXP1200 library	Describes the IXP1200 hardware functions.
13 IXF440 microcode library	Describes the IXF440 microcode library.
14 IXF1002 microcode library	Describes the IXF1002 microcode library.
15 TDM2IX microcode library	Describes the TDM2IX microcode library.

Part III: Appendices

The appendices include:

Appendix	Description
A Status codes	Lists and describes the ENP SDK status codes.
B Diagnostic tests	Explains how to use the diagnostic utilities provided as part of your ENP SDK software.
C Re-programming the flash chip	Explains how to update or recover programs that reside on the flash chip. You accomplish this by re-programming all or part of an ENP board's flash chip.

Notational conventions

This manual uses the following conventions:

- `Screen text and syntax strings appear in this font.`
- All numbers are decimal unless otherwise stated.
- Bit 0 is the low-order bit. If a bit is set to 1, the associated description is true unless otherwise stated.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com


Notes indicate important information about the product.


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


The globe indicates a World Wide Web address.


The book indicates a book or file.


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


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


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


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

Where to get more information

About the ENP software development kit

You can find out more about ENP SDK from these sources:

- **Readme file:** Lists features and issues that arose too late to include in other documentation.
- **World Wide Web:** RadiSys maintains an active site on the World Wide Web. The site contains current information about the company and locations of sales offices, new and existing products, contacts for sales, service, and technical support information. You can also send e-mail to RadiSys using the web site.

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

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

  Requests for sales, service, and technical support information receive prompt response.
- **Other:** If you purchased your RadiSys product from a third-party vendor, you can contact that vendor for service and support.

About related RadiSys products

ENP-2505

The ENP-2505 reference board uses a single Intel IXP1200 network processor for a variety of applications. It is PCI 2.2 compliant and works in both hosted and hostless environments.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The ENP-2505 can operate in any Windows NT workstation with a standard PCI slot, including a 32-bit, 33 MHz slot, or performs optimally in a 64 bit, 66 MHz PCI slot.

The ENP-2505 supports four 10/100 Mbps Ethernet ports, supported by an IXF440 multiport Ethernet controller, along with four LXT972 PHY transceivers, and a 21555 PCI-to-PCI bridge. The PCI expansion site allows you to add devices that extend processing power for specialized applications and development tasks.

ENP-2506

The ENP-2505 reference board uses a single Intel IXP1200 network processor for a variety of applications. It is PCI 2.2 compliant and works in both hosted and hostless environments.

ENP-3511

The ENP-3511, a 6U single-slot 64 bit CompactPCI board, is ideal for TDM-to-packet or packet-to-packet applications. It employs an IXP1200 network processor and incorporates an H.110 time-slot interchange chip, a TDM-to-IX bus conversion FPGA or ASIC, and eight 10/100 Ethernet ports on the IX bus to perform this function.

For more information about this board, see the *ENP-3511 Hardware Reference*.

References

RadiSys

ENP-3511 Hardware Reference: RadiSys Corporation, 2002.

ENP-2505 Hardware Reference: RadiSys Corporation, 2002.

ENP-2506 Hardware Reference: RadiSys Corporation, 2002.

Intel

Intelligent Platform Management Interface Specification, revision 1.0: Intel Corporation, Hewlett-Packard Company, NEC Corporation, Dell Computer Corporation, 1998.

Intel IXF0002 Dual Port Gigabit Ethernet Controller Data Sheet: Intel Corporation, May 2001.

Intel IXF440 Multiport 10/100 Mbps Ethernet Controller Data Sheet: Intel Corporation, May 2001.

Intel IXP1200 Network Processor Family Development Tools User's Guide: Intel Corporation, June 2001.

Intel IXP1200 Network Processor Family Hardware Reference Manual: Intel Corporation, June 2001.

Intel IXP1200 Network Processor Family Microcode Programmer's Reference Manual: Intel Corporation, June 2001.

Intel IXP1200 Network Processor Family Microcode Software Reference Manual: Intel Corporation, June 2001.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Other

CompactPCI Specification (PICMG 2.0 R3.0): PCI Industrial Computers Manufacturers Group, 1999.

CompactPCI Computer Telephony Specification (PICMG 2.5 R1.0): PCI Industrial Computers Manufacturers Group, 1998.

H.110 Hardware Compatibility Specification: CT Bus, revision 1.0: Enterprise Computer Telephony Forum, 1997.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Contents

Part I: Introducing the ENP SDK

Chapter 1: Overview

Features 3
ENP SDK component distribution 3
Developing an ENP application 4
ENP application architecture 4
Programming environment 5

Chapter 2: Installation and configuration

Installing the ENP Software Development Kit 7
Requirements 7
Before you begin 7
Install the software 8
Determining the ENP SDK version 8
Uninstalling the ENP SDK 8
Code organization 9
Configuring the ENP board for boot 10
Changing hardware settings 10
Using the Boot Manager 11
Using ENP software 11
Sample applications 11
ENP APIs 12

Chapter 3: Developing ENP applications

Typical ENP applications 13
Network to network data path 15
ENP-3511 15
ENP-2505 and ENP-2506 16
Network to PCI data path 17
ENP-3511 17
ENP-2505 and ENP-2506 18
Network to TDM data path 19
ENP-3511 19
TDM to TDM data path 20
ENP-3511 20
PCI-to-TDM data path 21
PCI-to-PCI data path 21

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 4: Sample applications

ENP-2505 sample application: ForwardingDemo 23
Description 23
Running the application 24
ENP-2506 sample applications 26
Simulating the applications 26
Running the sample applications on hardware 26
IXF1002IngressEgress 27
ENP2506FwdDemo 28
ENP2506LoopbackTest 29
ENP-3511 sample application: TdmDemo 30
Description 30
Data flow detail 30
Running the application 31

Part II: Application programming interfaces

Chapter 5: System Monitor software

Booting your ENP system 35
System Monitor environment 35
System Monitor commands 36
delete 37
ipmi 38
load 39
peek 40
poke 41
program 42
reboot 43
reflash 44
set 45
show 46

Chapter 6: Boot Manager software

Booting your ENP system 47
Boot Manager commands 48
b (boot) 48
c (change) 48
h (help) 49
p (print) 49

Chapter 7: PCI communications

Overview 51
PCI controller components 52
Windows NT device driver 52
Windows NT API 52
Windows NT files 52
PCI peripheral components 53
VxWorks driver 53
VxWorks API 53
VxWorks files 53
VxWorks configuration 54

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function list 55
Windows NT API 55
VxWorks API 55
Windows NT API 56
IXPHPCAsyncOpen 56
IXPHPCAsyncRead 58
IXPHPCAsyncWrite 60
IXPHPCClose 62
IXPHPCEnumerate 63
IXPHPCReset 65
IXPHPCSyncOpen 66
IXPHPCSyncRead 68
IXPHPCSyncWrite 69
VxWorks API 71
ENPClosePCIChannel 71
ENPOpenPCIChannel 72
ENPPCIAPIPrintError 73
ENPReadFromPCIChannel 74
ENPWriteToPCIChannel 75

Chapter 8: TDM routing and switching library

Overview 77
Implementation 77
Architecture 78
Concepts 78
ENP boards and T8105 79
Interrupt routines 79
Sample code 81
Sample startup sequence 81
Example: clock configuration 81
Example: time slot switching 84
Local-to-Local (T1/E1 Framer 2 to TDM2IX) 84
Local-to-CT_BUS (TDM2IX-to-CT_BUS) 84
TDM-to-TDM connection 85
Function and structure list 86
Functions 87

Chapter 8:

TDMClearClockFault 87
TDMClearMemoryFault 88
t_TDMClockConfig 89
t_TDMConnection 91
TDMGetSwitchErrorStatus 92
TDMInitSwitch 93
TDMSetClockConfig 94
TDMSetInterruptHandler 95
TDMSetStreamConfig 96
TDMSetSwitchConfig 97
t_TDMStreamConfig 98
t_TDMSwitchConfig 99

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Structures 100
t_fallback_clk 100
t_netref_clk 101
t_ref_clk 102
t_source_dest 103
t_stream_rate 104

Chapter 9: E1T1 library

Overview 105
Sample startup sequence 106
T1 Initialization example 107
Function list 109
Functions 113
T1E1initCard 113
T1E1clockConfig 114
T1E1getBoardConfig 115
T1E1setLeds 116
setT1Config 117
setT1Signaling 118
setT1Command 119
setT1ClearChannels 120
setT1IdleChannels 121
setT1ChannelConfig 122
setE1Config 123
setE1Signaling 124
getT1Signaling 125
getT1Status 126
getT1SignalingRaw 127
getE1Signaling 128
setT11SignalingHandler 129
setT1StatusHandler 130
setE1SignalingHandler 131
Structures 132
t_T1E1_framer_id 134
t_T1E1_card_type 135
t_T1E1_led_state 136
t_T1E1_user_signaling_data 137
t_T1E1_BoardConfig 138
t_T1E1_clock_mode 139
t_T1E1_rclock_ref 140
t_T1E1_rclock_rclk 141
t_T1E1_int_rcv_sys_clk 142
t_T1E1_xclock_DCOX 143
t_T1E1_int_xmit_sys_clk 144
t_T1E1_ClockConfig 145
t_T1_line_coding 146
t_T1_framing_mode 147
t_T1_line_buildout 148
t_T1_user_config_struct 149
t_T1_user_signaling_data,

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1_signaling_data 151
t_T1_user_command_data 152
t_T1_user_status_struct 153
t_T1_user_clear_channel_data 154
t_T1_user_idle_struct 155
t_T1_user_channel_config 156
t_T1_user_raw_signaling_struct 158
Example 158
T1StatusHandler 159
T1SignalingHandler 160
t_E1_line_coding 161
t_E1_signaling_mode 162
t_E1_line_buildout 163
t_E1_user_config_struct 164
t_E1_user_signaling_data 166
E1SignalingHandler 167

Chapter 10: Interrupt library

Introduction 169
Software architecture 171
CINT (MAC devices): interrupt handling 172
MicroEngine interrupt handling 173
InterruptStatus 174
Device-specific interrupt function list 175

Chapter 11: ENP microcode resource manager

Overview 177
Memory maps 177
Interrupt routines 178
CPLD interrupt routines 178
Microengine interrupt routines 179
RTC interrupt routines 179
Function list 180
CPLD_DisableAll_Interrupt 181
CPLD_Disable_Interrupt 182
CPLD_EnableAll_Interrupt 183
CPLD_Enable_Interrupt 184
CPLD_SetInterruptHandler 185
ENP2505_InitApp 186
ENP2506_InitApp 187
ENP3510_InitApp 188
ENP3511_InitApp 189
ENPCreateEpktStack 190
ENPDestroyEpktStack 191
ENPInitThreadID 192
RTC_Disable_Interrupt 193
RTC_Enable_Interrupt 194
RTC_InitClock 195
RTC_Set_Int_frequency 196
RTC_SetInterruptHandler 197

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

uEng_Enable_Interrupt 198
uEng_Disable_Interrupt 199
uEng_SetInterruptHandler 200

Chapter 12: IXP1200 library

Function list 201
IXPGetReadLockRetryStatus 202
IXPPushSramStack 203
IXPPopSramStack 204
IXPReadLockSramA32 205
IXPReadReg 206
IXPReadScratchA32 207
IXPReadSramA32 208
IXPWriteReg 209
IXPWriteScratchA32 210
IXPWriteSramA32 211
IXPWriteUnlockSramA32 212

Chapter 13: IXF440 microcode library

Overview 213
Microcode libraries 213
Programming model 213
IXF440 receive module 214
IXF440 transmit module 215
IXF440 data flow 216
Using the IFX440 microcode drivers 216
Start-up and initialization 216
Example code 217
Managing packets 218
Epkt descriptor 218
IXF440 receive queue 219
IXF440 transmit queue 219
IXF440 uPI 219
IXF440 Receive thread uPI 220
IXF440 transmit thread uPI 220
IXF440 interrupt routines 221
Function list 222
IXF440ClearNetStats 223
IXF440ClosePort 224
IXF440DisableAllInterrupt 225
IXF440DisableInterrupt 226
IXF440EnableAllInterrupt 227
IXF440EnableInterrupt 228
IXF440GetDuplexMode 229
IXF440GetRxThreadError 230
IXF440GetRxTxMode 231
IXF440GetTxThreadError 232
IXF440HookPort 233
IXF440OpenPort 234
IXF440QueryNetStats 235

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440ReadPacket 236
IXF440ReadReg 237
IXF440ResetPort 238
IXF440SetDuplexMode 239
IXF440SetInterruptHandler 240
IXF440StartPort 241
IXF440StopPort 242
IXF440UnhookPort 243
IXF440WritePacket 244
IXF440WriteReg 245

Chapter 14: IXF1002 microcode library

Programming model 247
Functional details 248
Overview 248
IFX1002 ingress module 248
Overview 248
Thread sequencing 248
Initialization 248
Receive requests 250
Fast port thread-to-thread communication 250
Buffer descriptors 251
Return conditions 251
IXF1002 egress module 251
Overview 251
Initialization 252
Transmit queues 253
Thread sequencing 253
Global GPR mailboxes 253
TFIFO element control 253
Packet counters 254
Buffer descriptors 254
Ingress macro definitions 256
IXF1002_Ingress_Init 256
IXF1002_Ingress 257
Data structures 259
Buffer descriptors 259
Epkt buffer descriptors 259
FastEpkt buffer descriptors 260
enpFqueue 260
IFX1002 µPI 262
IXF1002 ingress µPI 262
IFX1002 egress µPI 262
Using the IFX1002 microcode drivers 263
Ingress cycle time 263
Start-up and initialization 263
IXF1002 interrupt routines 264

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function list 265
IXF1002 AAPI 267
IXF1002ClearNetStats 267
IXF1002ClosePort 268
IXF1002DefineTxQueue 269
IXF1002DisableAllInterrupt 270
IXF1002DisableInterrupt 271
IXF1002EnableAllInterrupt 272
IXF1002EnableInterrupt 273
IXF1002GetRxTxMode 274
IXF1002GetTxEpktCount 275
IXF1002GetTxMpktCount 276
IXF1002HookPort 277
IXF1002OpenPort 278
IXF1002QueryNetStats 279
IXF1002ReadNetStatReg 280
IXF1002ReadPacket 281
IXF1002ReadReg 282
IXF1002ResetPort 283
IXF1002SetGMIIMode 284
IXF1002SetGPCSMode 285
IXF1002SetInterruptHandler 286
IXF1002StartPort 287
IXF1002StopPort 288
IXF1002UnhookPort 289
IXF1002WritePacket 290
IXF1002WriteReg 291

Chapter 15: TDM2IX microcode library

Overview 293
Microcode libraries 293
Programming model 293
TDM2IX Receive module 294
TDM to IX receive module 294
Receive data buffer 294
Receive buffer queue 294
Timing considerations 294
Error conditions 295
Rx TDM2IX packet count 296
TDM2IX Transmit module 297
Transmit data buffer(s) 297
Transmit buffer queue 297
Error conditions 297
Tx TDM2IX packet count 298
TDM2IX interrupt routines 299

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function list 300
TDM2IXClosePort 301
TDM2IXDisableAllInterrupt 302
TDM2IXDisableInterrupt 303
TDM2IXEnableAllInterrupt 304
TDM2IXEnableInterrupt 305
TDM2IXGetRxThreadError 306
TDM2IXGetTxThreadError 307
TDM2IXHookPort 308
TDM2IXOpenPort 309
TDM2IXReadPacket 310
TDM2IXReadReg 311
TDM2IXResetPort 312
TDM2IXSetInterruptHandler 313
TDM2IXSetRxLength 314
TDM2IXSetTxLength 315
TDM2IXStartPort 316
TDM2IXStopPort 317
TDM2IXUnhookPort 318
TDM2IXWritePacket 319
TDM2IXWriteReg 320

Part III: Appendices

Appendix A: Status codes 323

Appendix B: Diagnostic tests

test 21555 327
test cpld 328
test e1t1 330
test ethernet 331
test flash 333
test h110 334
test ipmi 335
test ixf440 336
test ixf1002 338
test macaddrs 340
test pci 341
test scratch 342
test sdram 343
test sram 344
test t8105 345
test tdm2ix 346
test teeprom 347
Loopback connectors 348
T1/E1 loopback 348
Ethernet loopback 348
Ethernet crossover loopback 348
CompactPCI H110 loopback 349

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix C: Re-programming the flash chip

About the flash chip 352
About re-programming the flash chip 353
Before you begin 354
Re-programming the flash chip from VxWorks 356
Re-programming the flash chip from the System monitor 357
Re-programming the flash chip from the Boot Manager 358
About the Flash utility setup and enumeration 358
Reprogramming the flash 358

Glossary 361

Index 365

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Figures

Figure 1-1. ENP application architecture 4
Figure 3-1. Abstracting primary I/O interfaces 13
Figure 3-2. ENP-3511: network to network control and data paths 15
Figure 3-3. ENP-2505: network to network control and data paths 16
Figure 3-4. ENP-3511: Network-to-PCI control and data paths 17
Figure 3-5. ENP-2505: Network-to-PCI control and data paths 18
Figure 3-5. ENP-3511: network-to-TDM control and data paths 19
Figure 3-6. ENP-3511: TDM-to-TDM control and data paths 20
Figure 4-1. Forwarding demo data flow 24
Figure 4-2. SmartBits screen 25
Figure 4-3. TdmDemo data flow 30
Figure 7-1. IXA PCI communications software architecture 51
Figure 8-1. Programming model 78
Figure 8-2. T8105 interfaces 79
Figure 8-3. Sample clock fallback 82
Figure 10-1. ENP interrupt sources 170
Figure 10-2. Architecture 171
Figure 10-3. CINT/Mac device interrupt handling 172
Figure 10-4. Microengine interrupt handling 174
Figure 11-1. Memory map: SRAM (32-bit addressing) 177
Figure 13-1. ENP board: IXF440 microcode programming model 213
Figure 13-2. IXF440 packet data flow 216
Figure 13-3. Epkt descriptor format 218
Figure 13-4. IXF440 receive queue format 219
Figure 13-5. IXF440 transmit queue format 219
Figure 14-1. ENP board: IFX1002 microcode programming model 247
Figure 14-2. IFX1002 Ethernet packet data flow 255
Figure 14-3. enpFqueue data structure 261
Figure 15-1. ENP board: microcode programming model 293
Figure 15-2. IXF440 packet data flow 296
Figure 15-3. IXF440 packet data flow 298
Figure C-1. Flash chip configuration 352
Figure C-2. Flash chip re-programming coverage 353
Figure C-3. Flash chip re-programming process flow 354
Figure C-4. Flash utility components 358

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Tables

Table 5-1. System Monitor commands 36
Table 6-1. Boot Manager commands 48
Table 7-1. PCI communication functions: Windows NT API 55
Table 7-2. PCI communication functions: VxWorks API 55
Table 8-1. InterruptStatus structure 80
Table 8-2. TDM functions 86
Table 8-3. TDM structures 86
Table 9-1. T1/E1 functions 109
Table 9-2. Time slot numbers 111
Table 9-3. E1/T1 structures 132
Table 10-1. IXP1200 interrupt vectors 170
Table 10-2. TDM interrupt functions (Chapter 8) 175
Table 10-3. ENP interrupt functions (Chapter 11) 175
Table 10-4. IXF440 microcode interrupt functions (Chapter 13) 175
Table 10-5. IXF1002 microcode interrupt functions (Chapter 14) 176
Table 10-6. TDM2IX microcode interrupt functions (Chapter 15) 176
Table 11-1. Memory map: scratch (32-bit addressing) 178
Table 11-2. Memory map: SDRAM (64-bit addressing) 178
Table 11-3. InterruptStatus structure 179
Table 11-4. ENP functions 180
Table 12-1. IXP1200 microcode functions 201
Table 13-1. IXF440 receiver error conditions 214
Table 13-2. IXF440 transmitter error conditions 215
Table 13-3. InterruptStatus structure 221
Table 13-4. IXF440 microcode functions 222
Table 14-1. InterruptStatus structure 264
Table 14-2. IXF1002 microcode functions 265
Table 15-1. InterruptStatus structure 299
Table 15-2. TDM2IX microcode functions 300
Table B-1. Diagnostic tests 325
Table B-2. Ethernet loopback pin-out 348
Table B-3. Ethernet loopback pin-out 348
Table B-4. Ethernet crossover loopback pin-out 348
Table B-5. CompactPCI H110 loopback pin-out 349

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

I Introducing the ENP SDK

This part acquaints you with the ENP Software Development Kit: its components, how they're put together, and how to develop applications for RadiSys ENP boards.

This part contains:

Chapter 1: Overview

Describes ENP software development kit features and tells you where to find detailed information about ENP SDK topics.



Read this chapter first. It introduces you to all the basic ENP SDK concepts and tells you where to find detailed information.

Chapter 2: Installation and configuration

Explains how to install and configure your ENP software development kit.

Chapter 3: Developing ENP applications

Overviews typical application architectures suitable for use on ENP boards.

Chapter 4: Sample applications

Describes the sample applications included with the ENP SDK.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1 Overview

The ENP SDK provides programming and runtime libraries for the Intel Internet Exchange Architecture (IXA) SDK that adds software functionality specific to RadiSys platforms. The IXA SDK serves as the primary software development environment. You use the ENP SDK to quickly develop optimized applications for the Intel IXP1200 network processor.

Features

The ENP SDK:

- Contains microcode "drivers" that abstract the primary interfaces provided by ENP hardware to the IXP1200 microengines, simplifying microcode development and providing source-level portability to other hardware platforms.
- Contains a complete set of boot and diagnostic software for each supported platform.
- Includes a reflash utility that updates all or part of ENP hardware flash images.
- Provides support for these operating systems:
  - **On a Windows NT workstation**: Windows NT 4.0, Service Pack 6.
  - **On an ENP board**: VxWorks BSP version 1.2, revision 2.



To do StrongARM† development, you must purchase an IDE such as Tornado 2.0, available from Wind River.

ENP SDK component distribution

ENP SDK components are distributed among the hardware components that comprise ENP boards:

- **IXP1200**: The processor (all platforms).
- **IXF440**: An octal port Ethernet controller (all platforms).
- **T8105**: An H.110 compliant Timeslot Interchange (TSI) chip that can switch 512 timeslots between the local TDM bus (CHI Bus) and the H.110 backplane TDM bus (ENP-3511 only).
- **TDM2IX**: A device that performs the TDM-to-IX bus interface, transferring up to 256 timeslots in each direction (ENP-3511 only).
- **E1T1**: A 4-port E1/T1 option card. Provides the hardware to support up to four 1.544 Mb/s T1/J1 or 2.048 Mb/s E1 network links.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Developing an ENP application

To develop an ENP application, you use the IXA SDK. ENP SDK software, an add-on to the IXA SDK, works in concert with the IXA SDK to supply the tools necessary to build value-added software for ENP platforms.

RadiSys does not ship the IXA SDK CD-ROM. However, RadiSys provides technical support for the IXA SDK on an ENP platform.

ENP application architecture

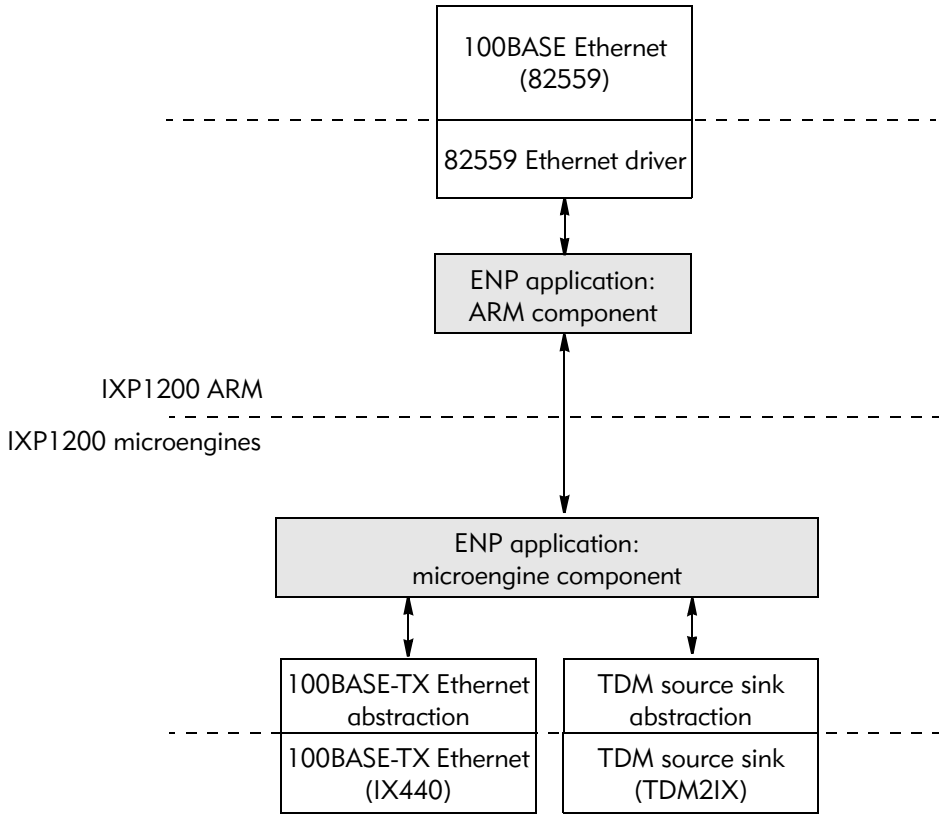
Differences in the capabilities and performance of the IXP1200 ARM processor and the IXP1200 microengines dictate the architecture and high-level functionality of a well-designed application.

Like all IXP1200 applications, ENP applications have two main components:

- **ARM component:** Executes on the IXP1200 ARM processor and primarily handles control and configuration. Involving this component in data manipulation and movement can reduce data path throughput.
- **Microengine component:** Executes on the IXP1200 microengines and primarily handles high-performance data manipulation and movement. Involving this component in control and configuration can reduce data path throughput.

This figure shows how the components work together:

Figure 1-1. ENP application architecture



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Programming environment

The IXA SDK provides the essential tools for writing, compiling, and debugging microengine-based components targeted for the ENP board.

IXA SDK requires:

- A PC that runs Windows NT 4.0 Workstation, Service Pack 6.
- Wind River Tornado, version 2.0.X, and VxWorks, version 5.4, (both for an ARM target) on the Windows NT host for development of ARM-based components that execute on the VxWorks operating system.

The ENP SDK provides additional tools for developing code specific to RadiSys ENP boards. To ensure correct operation of those tools, install the ENP SDK software *after* installing the IXA SDK.

Third parties may provide additional tools for developing both ARM-based and microengine-based components. All known third-party development tools are additions to the IXA SDK.



For more information, see the *IXP1200 Network Processor Development Tools User's Guide*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2 Installation and configuration

This chapter explains how to install your ENP Software Development Kit and configure your ENP boards for boot.

For information about...	Go to this page...
Installing the ENP Software Development Kit	7
Configuring the ENP board for boot	10

Installing the ENP Software Development Kit

Requirements

The ENP Software Development Kit requires:

- A Microsoft† Windows NT† compatible PC platform.
- A minimum 64MB of RAM.
- A minimum 30MB disk space.
- The following software installed on your computer:
  - Windows NT 4.0, Service Pack 6.
  - Wind River Tornado, version 2.0.
  - Intel† Internet Exchange Architecture (IXA) Software Developers Kit (SDK) software, version 1.3 (1.3.141) or 2.0 (2.0.83).

Before you begin

Before installing the ENP Software Development Kit:

- Ensure that you have the ENP SDK software, which is available on the ENP SDK CD-ROM.
- Ensure that you are logged on with Administrator privileges.
- Exit all programs prior to installing the ENP SDK.
- If your system has a previously installed version of the ENP SDK, remove it as described in *Uninstalling the ENP SDK*, later in this chapter.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Install the software

To install the ENP SDK:

1. Start the Setup program by inserting the ENP SDK CD-ROM into the CD-ROM drive. The Setup program automatically starts and the welcome screen displays.

   Note: If you've disabled the automatic start feature on your system, select:

   `Start>Settings>Control Panel>Add/Remove Programs`

   Click the Install button and follow the directions on the screen.

   Setup prompts you to select a destination directory.

2. Select a destination directory for the ENP SDK files. The default is:

   `C:\RadiSys\enp_sdk_2.0.`

   RadiSys recommends that you accept the default settings. Failure to do so may produce unexpected results.

   Setup creates the directory then prompts you to verify that you want the program files copied to that directory.

3. Verify that you are ready to install the files. Setup installs the ENP SDK files. For a detailed list of installed files, see *Code organization* later in this chapter.

   When finished, Setup displays the readme.txt file which contains:

   - Information about this release that occurred after other documentation was released.
   - Issues that require special attention to ensure that the ENP SDK runs properly.

Determining the ENP SDK version

To determine the version of ENP SDK installed on your system, select Start>Programs>RadiSys ENP SDK>Version Utility. The versions of all ENP SDK products installed on your system display.

Uninstalling the ENP SDK

Use one of these procedures to uninstall the ENP SDK prior to installing a new version:

- Use the Uninstall program that comes with the ENP SDK. To use this program, select `Start>Programs>RadiSys ENP SDK>Uninstall ENP SDK` and follow the instructions on the screen.
- Use the Windows NT uninstall utility:

  1. Select `Start>Settings>Control Panel>Add/Remove Programs`.
  2. Select ENP SDK, then click Remove. Uninstall removes the ENP SDK from your system.
  3. Select `Start>Shutdown`. You can either shut down or restart the computer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Code organization

When installed, the ENP SDK files reside in this directory:

```
\RadiSys\ENP_SDK_2.0
```

The ENP SDK files are arranged as follows:

Directory	File	Description
	install.log license.txt readme.txt setup.exe unwise.exe unwise.ini	
\Host	WinNT\lib	Contains all Windows NT library files.
	WinNT\include	Contains all Windows NT include files.
	WinNT\bin	Contains all Windows NT binary files.
\BoardSupport	Contains specific sources, projects, and samples for supported operating systems.	
	\ENP-2505[1]	Contains Tornado 2.0 BSP and VxWorks images for the ENP-2505 board.
	\ENP-2506[2]	Contains Tornado 2.0 BSP and VxWorks images for the ENP-2506 board.
	\ENP-3511[3]	Contains Tornado 2.0 BSP and VxWorks images for the ENP-3511 board.
	\BootROM	Contains combined BootROM images.
	\FPGA	Contains FPGA binary image for the ENP-3511 board.
\Documentation	Contains SDK documentation.	
\Include	Contains SDK header files.	
\Lib	Contains library files needed to link against to create custom versions of the applications.	
\MicroCode\	Includes microcode source and header files.	
	\include	Contains include files specific to microcode.
	\source	Contains microcode source files.
	\projects	Contains projects that demomstrate SDK usage.
\AcrobatReader	ar500enu.exe	Includes the freely distributable version of Adobe Acrobat reader.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Directory	File	Description
\DemoApps	\TdmDemo	Contains files needed to build the StrongARM portion of the TdmDemo application.
	\2505_ ForwardingDemo[1]	Contains all files needed to build the StrongARM portion of the 2505 Forwarding demo application.
	\2506_ ForwardingDemo[2]	Contains all files needed to build the StrongARM portion of the 2506 Forwarding demo application.
	\2506_Ingress/Egress[2]	Contains all files needed to build the StrongARM portion of the 2506 IngressEgress demo application.
	\2506_Loopback[2]	Contains all files needed to build the StrongARM portion of the 2506_LoopbackTest demo application.
	\E1_T1Demo	Contains all files needed to build the E1_T1 demo application.

[1] Included only if you selected ENP-2505 options during intallation.

[2] Included only if you selected ENP-2506 options during intallation.

[3] Included only if you selected ENP-3511 options during intallation.

Configuring the ENP board for boot

You can specify which application executes at boot—VxWorks, the Boot Manager, or the System Monitor—using any of the methods described below.

Changing hardware settings

Your ENP board provides jumpers or DIP switches that you use to configure the board. The position of the switches or jumpers determine which application executes at boot.



For information about jumper or switch settings for your ENP board, see the hardware reference for that board.

Your system can boot any of these:

- **VxWorks**: For more information about VxWorks, see your VxWorks documentation.
- **System Monitor**: When booted, this command prompt displays:

  ```
  enp>
  ```

  For more information about the System Monitor, see *Chapter 5, System Monitor software*.
- **Boot Manager**: When booted, this command prompt displays:

  ```
  [BootMgr]:
  ```

  For more information about the Boot Manager, see *Chapter 6, Boot Manager software*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Using the Boot Manager

The ENP Software Development Kit (SDK) includes the Boot Manager. From this application you can execute other available boot options.

To select a boot option:

1. Set your COM port to:

Baud rate:	38400
Parity:	N (no parity)
Data bits:	8
Stop bits:	1
Flow control:	N (no flow control)

2. Set your ENP board jumpers or switches for Boot Manager.
3. Power on the ENP board. The Boot Manager starts and displays this on the serial port:

   ```
   Press space bar to stop auto-boot...
   10
   ```

4. Press the space bar before the numerical value reaches zero. The Boot Manager prompt displays:

   ```
   [BootMgr]:
   ```

5. (Optional) Enter this command to list available OSs:

   ```
   p
   ```

6. Enter this command:

   ```
   b [n]
   ```

   where *n* is the number of the OS you want to boot when the countdown reaches 0 (zero). You can enter only listed OS numbers.

For detailed information about the Boot Manager and its commands, see *Chapter 6, Boot Manager software*.

Using ENP software

Sample applications

The ENP SDK includes sample applications that you can use as a model for your own application development. You can find the StrongArm portion of these applications after installation in these directories:

```
\RadiSys\ENP_SDK_2.0\DemoApps\TdmDemo
\RadiSys\ENP_SDK_2.0\DemoApps\ForwardingDemo
```

You can find the microcode portion of these applications after installation in these directories:

```
\RadiSys\ENP_SDK_2.0\MicroCode\projects\TdmDemo
\RadiSys\ENP_SDK_2.0\MicroCode\projects\ForwardingDemo
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

For information about building and running these applications, see *Chapter 4, Sample applications*.

ENP APIs

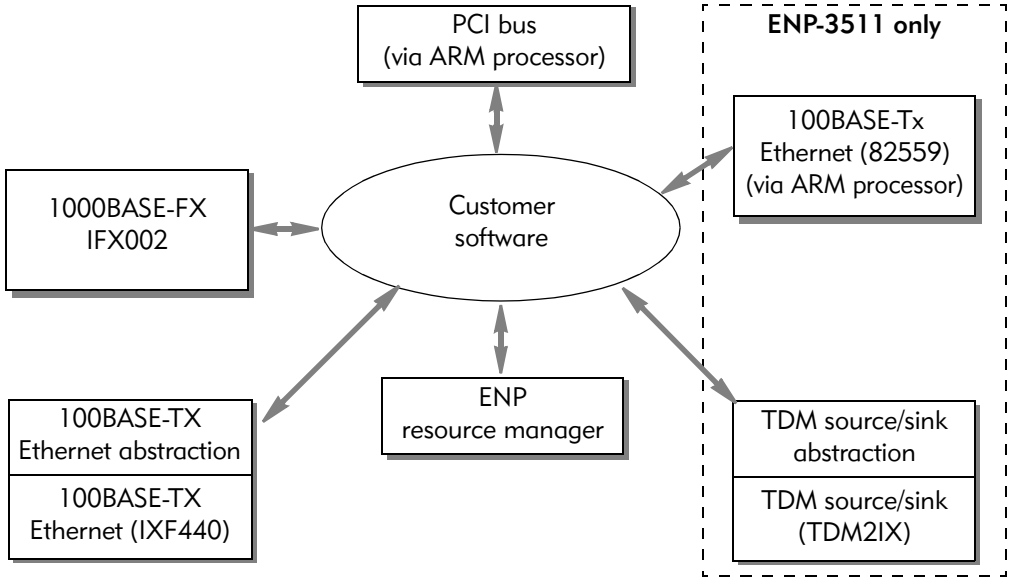
The ENP SDK includes various API libraries that you use to build ENP ENP applications. For detailed information, including syntax and parameter descriptions, see *Part II, Application programming interfaces*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3 Developing ENP applications

ENP software abstracts the primary I/O interfaces of the ENP board, presenting portable programming interfaces to customer software written for the IXP1200 ARM and microengine processors:

Figure 3-1. Abstracting primary I/O interfaces



Typical ENP applications

Any application that passes data at rates between 100 Mbits/second and 1500 Mbits/second between switched packet networks and/or the PCI bus and/or a TDM data source/sink, with moderate data manipulation, is a candidate for implementation. Typical applications include:

- Switches
- Routers
- Bandwidth aggregators
- Firewalls
- Intrusion detection devices
- TDM to packet bridge devices (ENP-3511 only)

ENP hardware and software is *not* well suited for applications that require digital signal processing or data encryption. Since the IXP1200 microengines have no significant arithmetic capability, such applications require additional acceleration

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

hardware (either multiple DSP processors or specialized encryption/decryption hardware, respectively).

A convenient mechanism for classifying ENP applications is by the data path (or paths) the applications use. Possible paths for data processed by an ENP board include:

- Network to network data path
- Network to PCI data path
- Network to TDM data path
- TDM to TDM data path
- PCI-to-TDM data path
- PCI-to-PCI data path

The following sections discuss these paths.

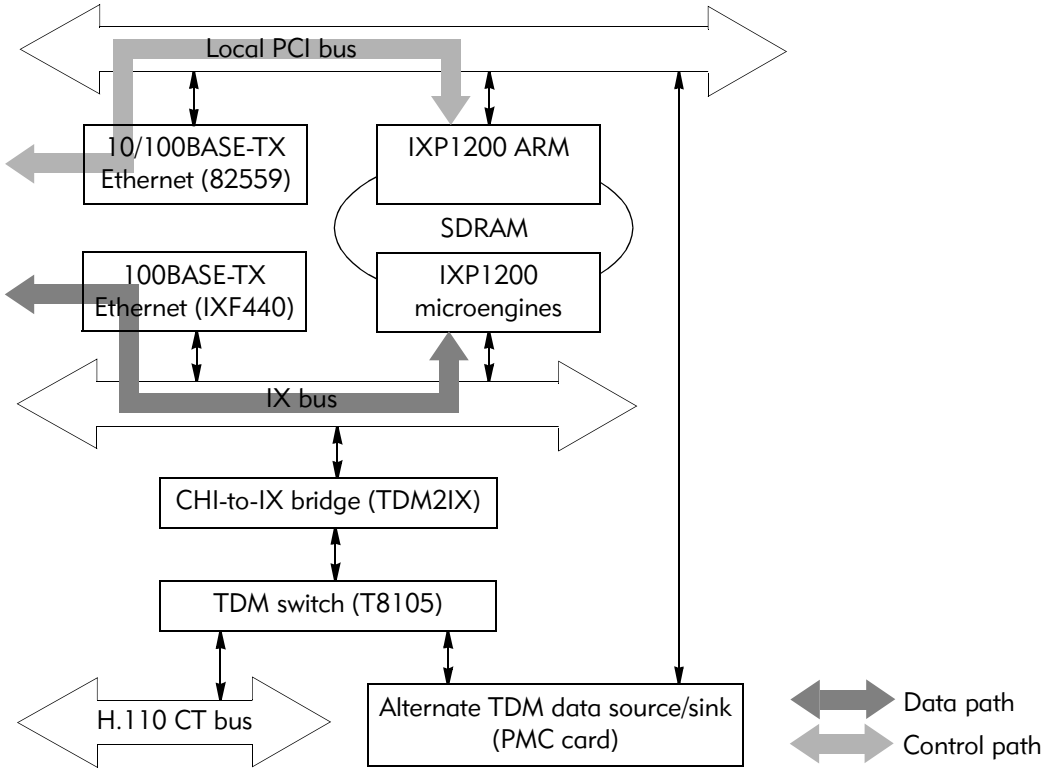
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Network to network data path

Simple routers, switches and bandwidth aggregators use the network-to-network data path, as shown in the next figure. Using the IXP1200 microengines, these applications receive packets from one or more switched networks, process the received packets and send packets back out onto one or more switched networks.

ENP-3511

Figure 3-2. ENP-3511: network to network control and data paths

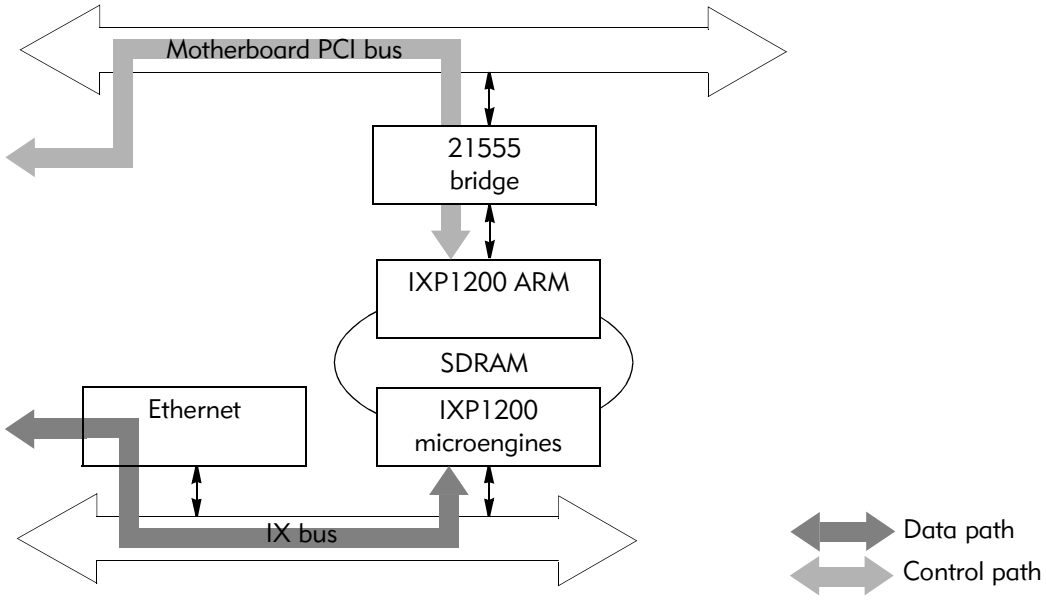


- A network-to-network application does not use the RadiSys ENP-3511 TDM capabilities; the hardware platform typically uses a board other than an ENP-3511. However, with proper abstraction of underlying hardware, ARM and microengine application source code developed on an ENP-3511 can port to the production platform.
- RadiSys recommends that you separate control and data paths, so that data path saturation does not block or delay control data and control data bursts do not degrade data path throughput.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENP-2505 and ENP-2506

Figure 3-3. ENP-2505: network to network control and data paths



RadiSys recommends that you separate control and data paths, so that data path saturation does not block or delay control data and control data bursts do not degrade data path throughput.

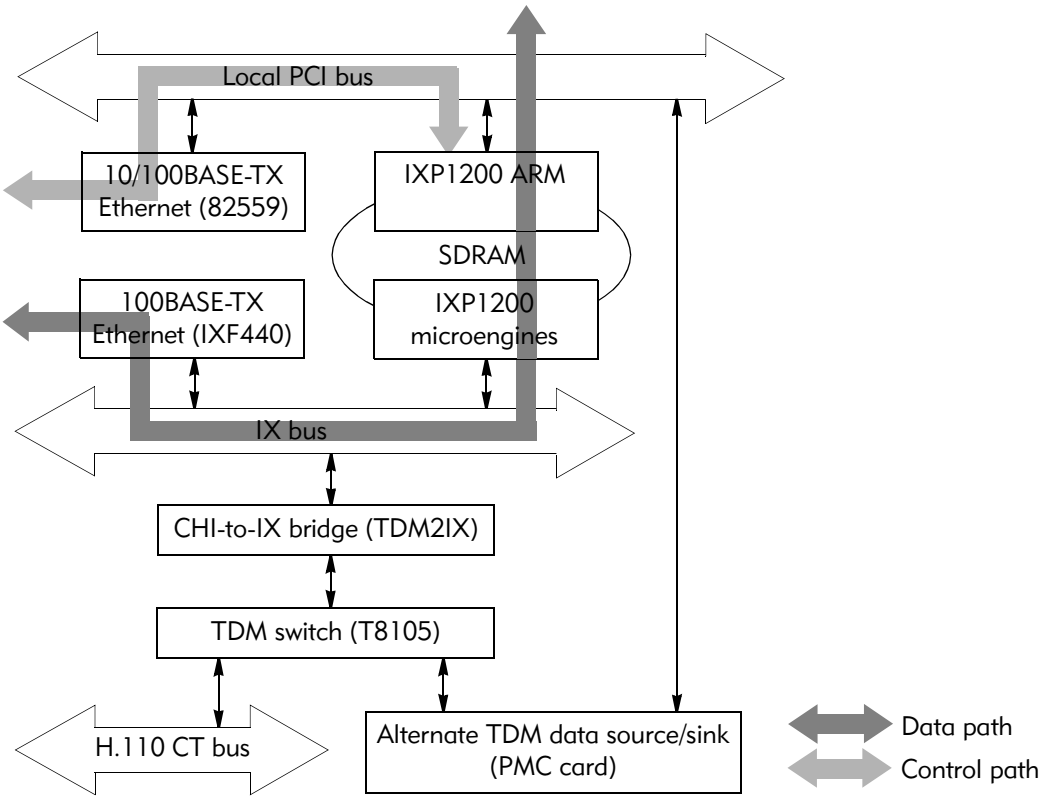
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Network to PCI data path

Multiple-board CompactPCI-based routers, switches, bandwidth aggregators, firewalls, encryption/decryption devices, and intrusion detection devices use the network-to PCI data path, as shown in the next figure. Using the IXP1200 microengines and the IXP1200 ARM processor, these applications receive packets from one or more switched networks, send the data to one or more coprocessor boards, receive processed data back from the coprocessor boards and send packets back out onto one or more switched networks.

ENP-3511

Figure 3-4. ENP-3511: Network-to-PCI control and data paths

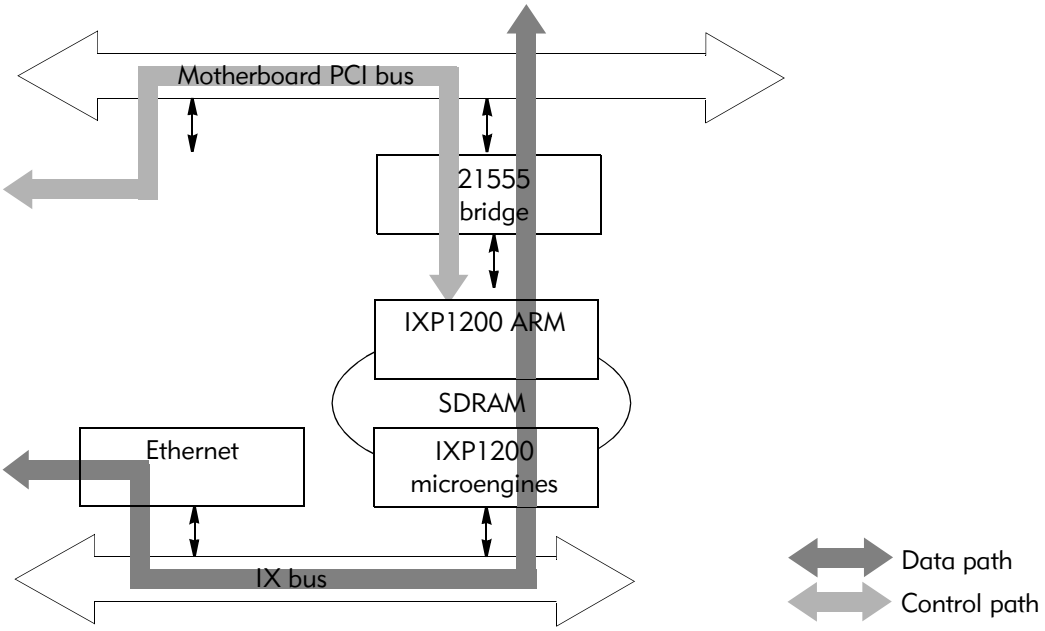


- A network-to-PCI application does not use the RadiSys ENP 3511's TDM capabilities; the hardware platform typically uses a board other than an ENP-3511. However, with proper abstraction of underlying hardware, ARM and microengine application source code developed on an ENP board can port to the production platform.
- RadiSys recommends that you separate control and data paths, so that data path saturation does not block or delay control data and control data bursts do not degrade data path throughput.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENP-2505 and ENP-2506

Figure 3-5. ENP-2505: Network-to-PCI control and data paths



RadiSys recommends that you separate control and data paths, so that data path saturation does not block or delay control data and control data bursts do not degrade data path throughput.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

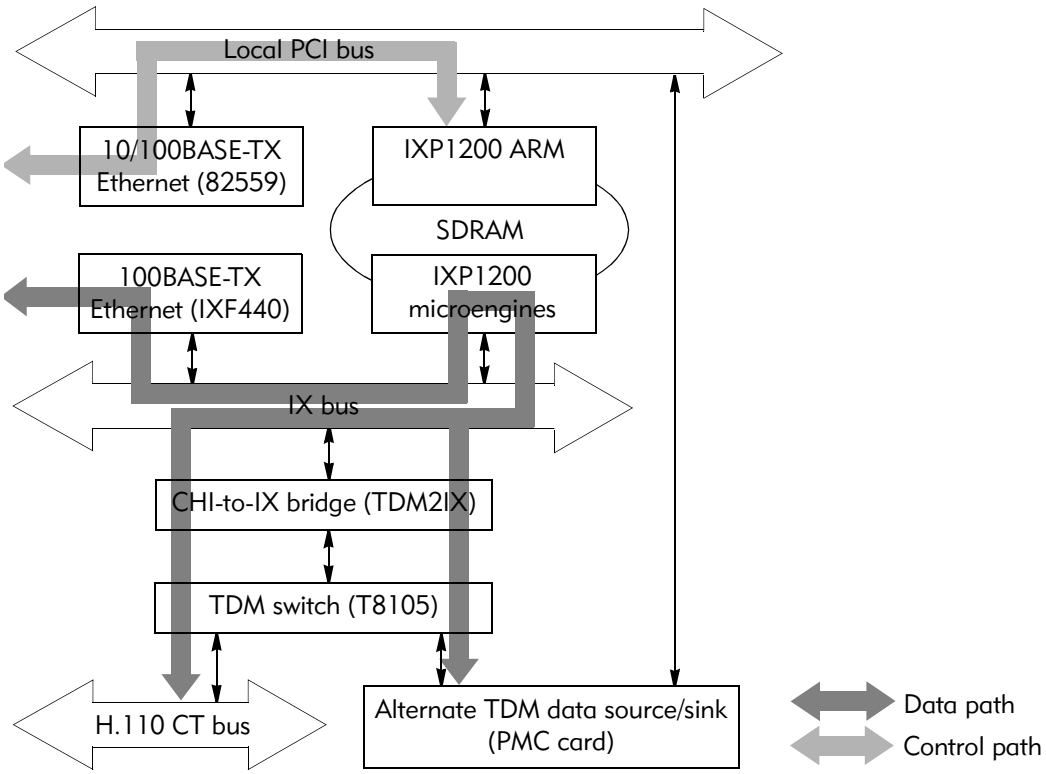
Network to TDM data path

Packetization engines use the network-to-TDM data path, as shown in the next figure. Using the IXP1200 microengines, these applications receive packets from one or more switched networks, extract TDM data from the packets, and send the TDM data to one or more TDM data sinks. The applications also use the IXP1200 microengines to receive TDM data from one or more TDM data sources, construct packets, and send the packets to one or more switched networks.

The ENP network-to-TDM bridge sample application fits this profile.

ENP-3511

Figure 3-5. ENP-3511: network-to-TDM control and data paths



- Whether data flows between the IXP1200 microengines and the H.110 CT Bus, the IXP1200 microengines and an alternate TDM data source/sink, or both depends upon the application.
- RadiSys recommends that you separate control and data paths, so that data path saturation does not block or delay control data and control data bursts do not degrade data path throughput.

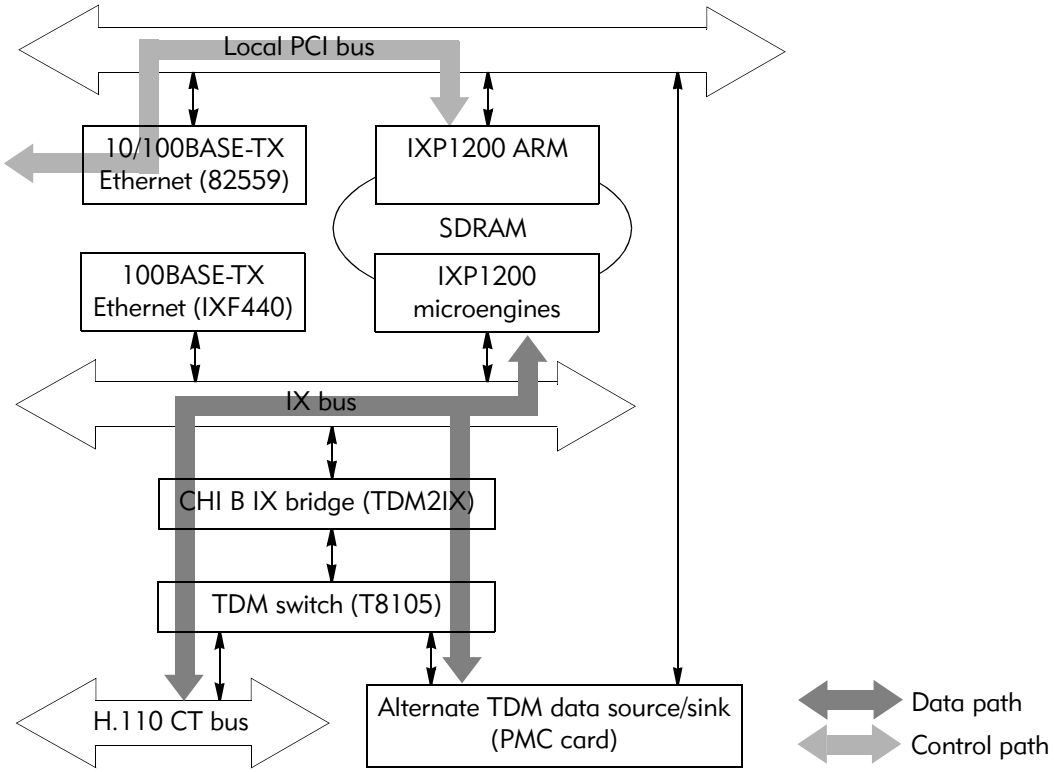
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM to TDM data path

TDM data stream manipulation applications use the TDM to TDM data path, as shown in the next figure. Using the IXP1200 microengines, these applications receive TDM data from one or more TDM data sources, manipulate the data, and send the TDM data to one or more TDM data sinks.

ENP-3511

Figure 3-6. ENP-3511: TDM-to-TDM control and data paths



- A TDM-to-TDM application does not use the RadiSys ENP board's network capabilities; the hardware platform typically uses a board other than an ENP. However, with proper abstraction of underlying hardware, ARM and microengine application source code developed on an ENP board can port to the production platform.
- RadiSys recommends that you separate control and data paths, so that data path saturation does not block or delay control data and control data bursts do not degrade data path throughput.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PCI-to-TDM data path



This data path is mentioned only for completeness.

Individual chips exist that perform this functionality.

PCI-to-PCI data path



This data path is mentioned only for completeness.

PCI bridge chips used in conjunction with general-purpose processors already provide this functionality.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4 Sample applications

This product includes the sample applications listed in the table below:

For information about...	Go to this page...
ENP-2505 sample application: ForwardingDemo	23
ENP-2506 sample applications	26
IXF1002IngressEgress	27
ENP2506FwdDemo	28
ENP2506LoopbackTest	29
ENP-3511 sample application: TdmDemo	30

ENP-2505 sample application: ForwardingDemo

This section describes the ForwardingDemo example application contained in the ENP SDK. The application demonstrates use of the ENP-2505 board as a basic router.

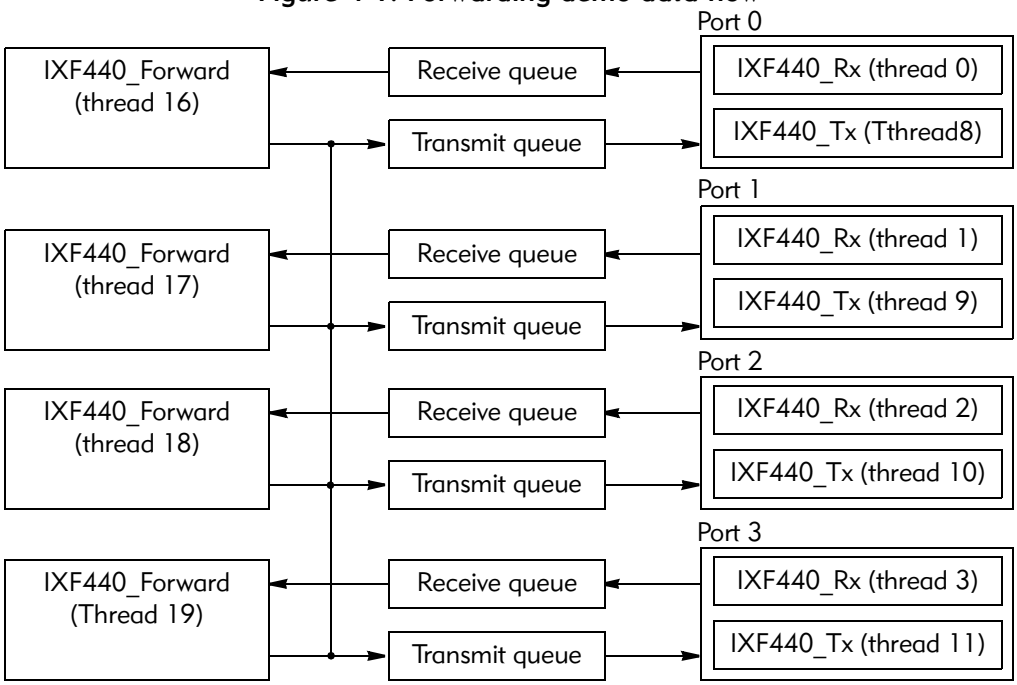
Description

The example application receives packets from the IXF440 device and places them into a queue associated with a particular IXF440_Rx thread. Each thread notifies a IXF440_Forward thread when packets await in the receive queue. The IXF440_Forward thread examines the destination IP address of each incoming packet, and determines whether it should be forwarded to another port based on a lookup in a routing

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

table. If the packet is to be forwarded to another port, the packet is modified and placed in the queue of the IXF440_Tx thread associated with the forwarding port.

Figure 4-1. Forwarding demo data flow



Running the application

To run the example application:

1. Boot the ENP-2505 to VxWorks and connect a VxWorks target server to the ENP-2505. (For details, see the *Intel Ethernet Evaluation System User's Manual*).
2. Launch NetApp_Init_PciDebug from the Tornado shell.
3. Start the developer workbench and load the ForwardingDemo project installed from the RadiSys ENP SDK.
4. Configure the workbench-to-hardware debugging and set the hardware options for connecting to the target board.
5. Launch the debugger and start the microengines.
6. From the Tornado shell, download ForwardingDemo.out.
7. Enter this command to initialize the microcode resource manager and configure the microcode:

   ```
   ForwardingDemo
   ```

8. Add route entries into the routing table by typing the following commands in Tornado Shell:

   ```
   rt_init();
   rt_add("1.1.1.1", "255.255.255.0", 1);
   rt_add("2.2.2.2", "255.255.255.0", 2);
   ```

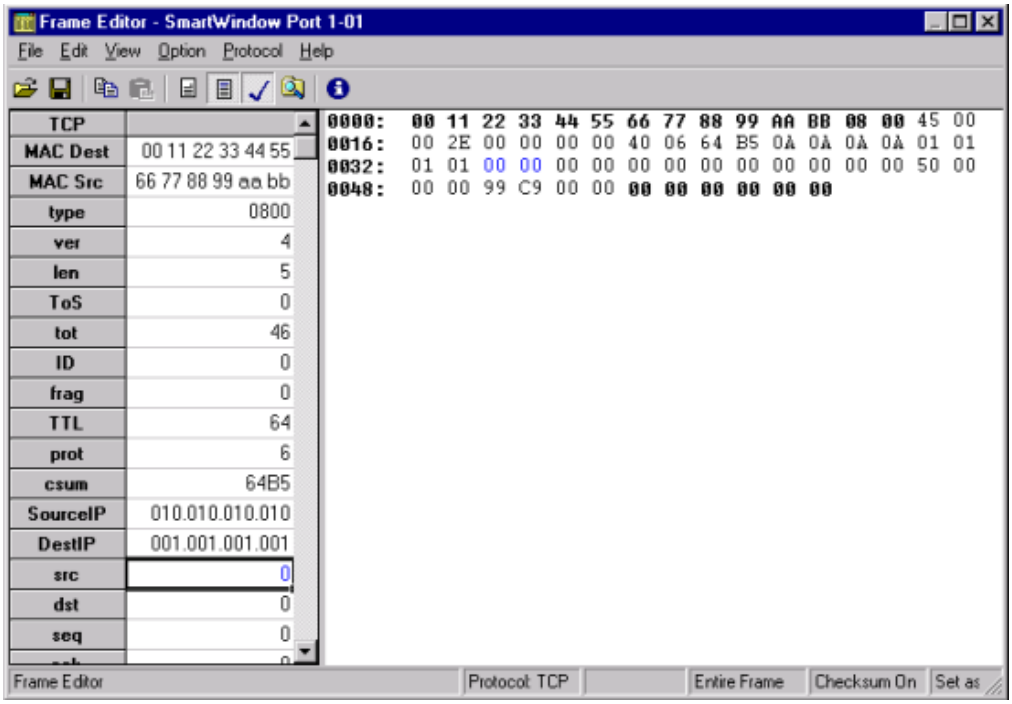

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
rt_add("3.3.3.3", "255.255.255.0", 3);
rt_add("4.4.4.4", "255.255.255.0", 0);
```

where the first parameter to rt_add is the destination IP address, the second parameter is the network mask, and the third parameter is the port to forward the packet to. For the ENP-2505 valid ports are 0 through 3.

9. Start a SmartBits or equivalent packet generator and setup packets to transmit on each port. The following is an example snapshot taken from SmartBits.

Figure 4-2. SmartBits screen



10. Start transmitting packets on each of the four ports and observe the reception of packets on the forwarding port.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENP-2506 sample applications

Simulating the applications

The example projects in this section are set up to simulate on the Developers Workbench.

- IXF1002IngressEgress
- IXF1002FwdDemo

To start the simulation process:

1. Open the project with the Developers Workbench.
2. Select the Simulation option under the Debug drop down menu.
3. If necessary, uncomment the `#define SIMULATION` statement in the project.h file.
4. If necessary, rebuild the project.
5. Start the debugger.
6. Start debugging by selecting the Start debugging option from the Debug drop-down menu, or by clicking the green GO symbol.

Running the sample applications on hardware

These sample applications run on hardware.

- IXF1002IngressEgress: Echoes packets back out on the port they were received.
- ENP2506FwdDemo: Forwards packets to a port based on the Intel IXA SDK route table set up in SRAM
- ENP2506LoopbackTest: Creates and sends test packets from the core to each IXF1002 port on the ENP-2506. Verifies that packets sent are received correctly at the opposite port. This applcication requires an external loopback cable.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Sample application: IXF1002IngressEgress

This application requires:

- A Windows NT workstation that uses PCI Debug.
- An ENP-2506 running VxWorks.

To run the application, perform these steps:

1. Start Tornado and launch a shell to connect to the ENP-2506 via the serial cable.
2. Download the VxWorks_ENP2506.o file from this ENP SDK directory:

   ```
   Radisys\ENP_SDK_2.01\BoardSupport\ENP2506\Vx_lib\
   ```

3. Enter this command:

   ```
   NetApp_GigInit_PciDebug
   ```

4. Start the Developers Workbench and open the IXF1002IngressEgress project.
5. Set the Workbench for hardware debugging and the hardware options to connect via PCI bus.
6. Comment out the `#define SIMULATION` line in the project.h microcode file.
7. Rebuild the microcode files, if necessary, and start the debugger.
8. Download the 2506_IngressEgress.out file, created using Tornado.
9. Start the microcode using the Developers Workbench.
10. Enter this command in the Tornado shell:

    ```
    IXF1002IngressEgress
    ```

The ENP-2506 is now set up to echo packets received on either IXF1002 port.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Sample application: ENP2506FwdDemo

This application requires:

- A Windows NT workstation that uses PCI Debug.
- An ENP-2506 running VxWorks.

To run the application, perform these steps:

1. Start Tornado and launch a shell to connect to the ENP-2506 via the serial cable.
2. Download the VxWorks_ENP2506.o file from this ENP SDK directory:

   ```
   Radisys\ENP_SDK_2.01\BoardSupport\ENP2506\Vx_lib\
   ```

3. Enter this command:

   ```
   NetApp_GigInit_PciDebug
   ```

4. Enter this command in the Tornado shell:

   ```
   NetApp_LoadRoutes
   ```

5. Start the Developers Workbench and open the IXF1002FwdDemo project.
6. Set the Workbench for hardware debugging and the hardware options to connect via PCI bus.
7. Comment out the `#define SIMULATION` line in the project.h microcode file.
8. Rebuild the microcode files, if necessary, and start the debugger.
9. Download the 2506_ForwardingDemo.out file, created using Tornado
10. Start the microcode using the Developers Workbench.
11. Enter this command in the Tornado shell:

    ```
    ENP2506FwdDemo
    ```

The ENP-2506 is now set up to forward packets with an IP address of 1.1.1.1 to port 1 and 4.4.4.6 to port 0.

Two example IP packets generated using a SmartBits 2000 with GX-1405B cards are shown below.

```
Packet 1: Destination = 1.1.1.1, Size = 60B without CRC
00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00
00 2E 00 00 00 00 40 04 76 C9 01 01 01 01 01 01
01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00
```

```
Packet 2: Destination = 4.4.4.6, Size = 60B without CRC
00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00
00 2E 00 00 00 00 40 04 6E BF 02 02 02 02 04 04
04 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Sample application: ENP2506LoopbackTest

This application requires:

- A Windows NT workstation that uses PCI Debug.

1. A crossover cable installed between the two Gigabit ethernet ports.

- An ENP-2506 running VxWorks.

To run the application, perform these steps:

1. Start Tornado and launch a shell to connect to the ENP-2506 via the serial cable.
2. Download the VxWorks_ENP2506.o file from this ENP SDK directory:

   `Radisys\ENP_SDK_2.01\BoardSupport\ENP2506\Vx_lib\`
3. Enter this command:

   `NetApp_GigInit_PciDebug`
4. Start the Developers Workbench and open the ENP2506LoopbackTest project.
5. Set the Workbench for hardware debugging and the hardware options to connect via PCI bus.
6. Comment out the `#define SIMULATION` line in the project.h microcode file.
7. Rebuild the microcode files, if necessary, and start the debugger.
8. Download the 2506_Loopback.out file, created using Tornado.
9. Start the microcode using the Developers Workbench.
10. Enter this command in the Tornado shell:

    `ENP2506InitLoopbackTest`
11. Enter this command to send a test packets through the loopback cable:

    `ENP2506RunLoopbackTest <number of iterations>`

    Example: `ENP2506RunLoopbackTest 100` sends 100 test packets from port 0 to port 1, and 100 test packets from port 1 to port 0).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

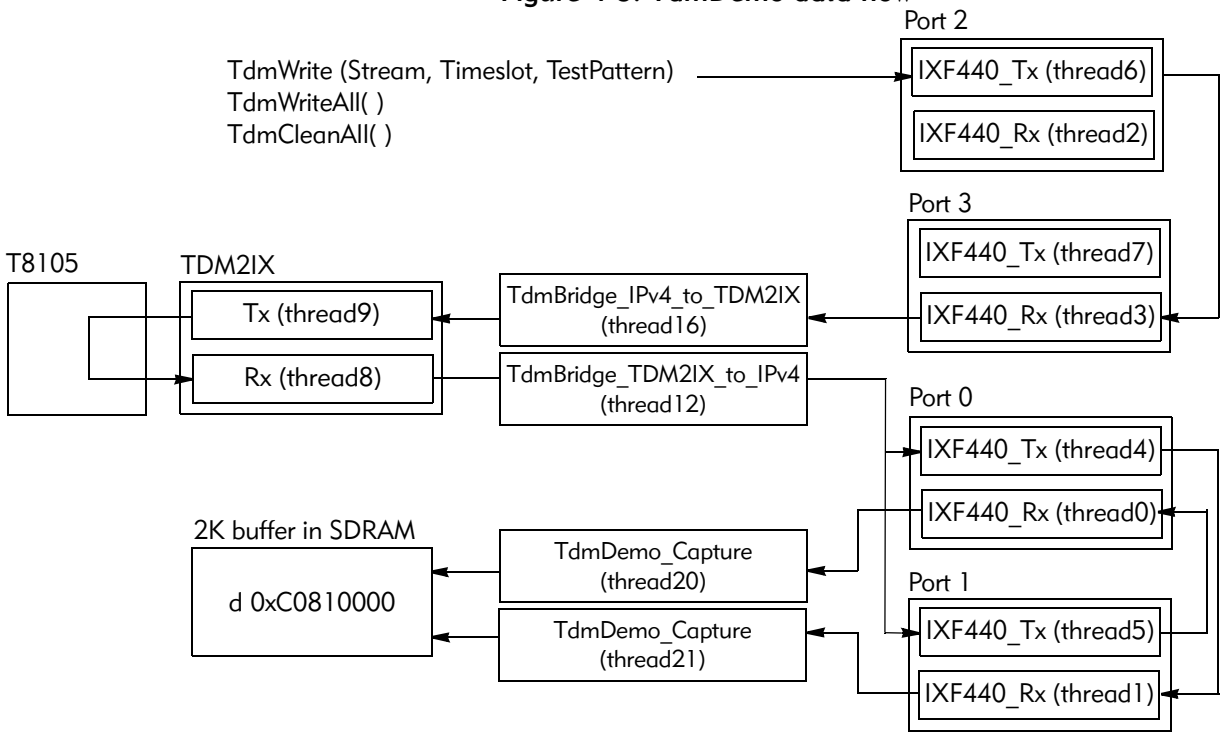
ENP-3511 sample application: TdmDemo

This section describes the TdmDemo example applications contained in the ENP SDK. The application demonstrates one way to move data between the TDM and Ethernet devices on the ENP-3511 board.

Description

The example application creates an IPv4 packet that contains TDM test data, sends the data through the Ethernet and TDM2IX devices, and eventually writes the data back into a memory buffer in SDRAM. The next figure shows the flow.

Figure 4-3. TdmDemo data flow



Data flow detail

1. An IPv4 packet, assembled on the StrongARM, contains 8 bytes of TDM data assigned to a specific stream and timeslot.
2. The packet is inserted into the Ethernet port 2 transmit queue.
3. A crossover cable between port 2 and port 3 routes the data into port 3 receive queue.
4. The target process for port 3 receive thread, TdmBridge_IPv4_to_Capone, gets the packet from port 3 receive queue, extracts the TDM data and inserts it into the TDM2IX transmit buffer.
5. The TDM data is looped back in the T8105 device with stream0 and stream1 exchanged (data sent on stream0, timeslot0 is received on stream1 timeslot0).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

6. The data is received into the TDM2IX receive queue.
7. The target process for the TDM2IX receive thread, TdmBridge_Capone_to_IPv4, extracts the data from all 256 timeslots regardless of validity and creates an IPv4 packet for each. Odd and Even timeslots are alternatively inserted into port 0 and port 1 transmit queues.
8. A crossover cable between port 0 and port 1 transmits data to port 0 and port 1 receive queues.
9. The target processes for port 0 and port 1 receive threads, TdmDemo_Capture, extract the TDM data from the packets and inserts it into a 2K buffer located in SDRAM. The stream and timeslot numbers determine the offset into the buffer and each timeslot contains 8 bytes of data, as shown in the next table.

Offset (bytes)	TDM Data Mapping
0x7F8	Stream 1, timeslot 127
…	…
0x408	Stream 1, timeslot 1
0x400	Stream 1, timeslot 0
0x3F8	Stream 0, timeslot 127
…	…
8	Stream 0, timeslot 1
0	Stream 0, timeslot 0

Note that a packet write to stream 0, timeslot 0 shows up in the buffer at offset 0x400 due to the switching setup of the T8105 device.

Running the application

To run the example application, peform these steps:

1. Boot the ENP-3511 to VxWorks and connect a VxWorks target server to the ENP-3511. (For details, see the *Intel Ethernet Evaluation System User's Manual*).
2. Launch "NetApp_Init" from the Tornado shell.
3. Start the developer workbench and load the TdmDemo project installed from the RadiSys ENP SDK.
4. Configure the workbench to hardware debugging and set the hardware options for connecting to the target board.
5. Launch the debugger and start the microengines.
6. From the Tornado shell, download DemoApps.out.
7. Enter this command to initialize the microcode resource manager and configure the microcode:

```
TdmDemo
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8. Use the following commands to send test data through the system and place it into the TDM display buffer.



To dump the TDM display buffer from the Tornado shell, enter this command:

```
d 0xc0810000
```

TdmWriteAll()

Writes a test pattern to all timeslots. Stream 0, timeslot 0 = 0x00, Stream 0, timeslot 1 = 0x01, …, Stream 1, timeslot 127 = 0xFF. Note that the streams are reversed in the TDM_DISPLAY_BUFFER. Bytes 0x00 – 0x07 are all 0x80, and bytes 0x400 – 0x407 are 0x00.

TdmClearAll()

Writes all zeros to all timeslots

TdmWrite(int Stream, int TimeSlot, Bit8 TestPattern)

Writes the value of TestPattern to the specified stream and timeslot. The data ends up in the same timeslot of the opposite stream.

9. To stop the TDM2IX device receive operation, enter "Stop" prior to stopping the microengines.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

II Application programming interfaces

This part details the various API libraries included in the ENP Software Development Kit, including syntax, parameter descriptions, and return values.

This part contains:

Chapter 5: System Monitor software

Describes the monitor software for the ENP platform, which provides a rudimentary execution environment.

Chapter 6: Boot Manager software

Describes the Boot Manager application, which specifies the boot OS from those available in the ENP board's flash chip.

Chapter 7: PCI communications

Describes the PCI communications library.

Chapter 8: TDM routing and switching library

Describes the TDM switching library, which encapsulates and abstracts call setup and teardown for ARM code.

Chapter 9: E1T1 library

Describes the E1/T1 library.

Chapter 10: Interrupt library

Describes the Interrupt library.

Chapter 11: ENP microcode resource manager

Describes the resource manager functions.

Chapter 12: IXP1200 library

Describes the IXP1200 hardware functions.

Chapter 13: IXF440 microcode library

Describes the IXF440 microcode library.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter 14: IXF1002 microcode library

Describes the IXF1002 microcode library.

Chapter 15: TDM2IX microcode library

Describes the TDM2IX microcode library.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5 System Monitor software

This chapter describes the System Monitor application. You use the System Monitor to manage the system and perform diagnostic tests.

This application is available only when specified via jumper settings on your ENP board. For information about setting jumpers, see *Configuring the ENP board for boot* on page 10.

Booting your ENP system

To boot your system:

1. Set the ENP board's switches or jumpers to the System Monitor setting. For information about jumper or switch settings for your ENP board, see the hardware reference for that board.
2. Set your COM port to:

Baud rate:	38400
Parity:	N (no parity)
Data bits:	8
Stop bits:	1
Flow control:	N (no flow control)

3. Power on your ENP board. The System Monitor starts and displays this on the serial port:

   ```
   enp>
   ```

System Monitor environment

The System Monitor provides an execution environment and includes a series of commands you use to manage the system.

The System Monitor also includes collection of tests that you use to verify the operational status of individual hardware components. For information about these tests, see *Appendix B, Diagnostic tests*.

You can recall previous commands by pressing the up and down arrow keys. You can also shorten command and keywords, when not ambiguous.

The remainder of this chapter describes the System Monitor commands.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

System Monitor commands

You use these commands to manage the System Monitor.

Use this table to identify the Monitor commands you want to use. Use the command description later in this chapter to obtain detailed information, including syntax and parameter values.

Table 5-1. System Monitor commands

Function	Description	Page
delete	Erases the flash block starting at the specified address.	37
ipmi	Provides support for the IPMI controller.	38
load	Loads an image from the network, serial port, or flash memory into main memory.	39
peek	Displays contents of the specified address.	40
poke	Writes the data into the specified address.	41
program	Programs an image located at the specified memory address into the specified flash address.	42
reboot	Generates a full system-wide reset system using the IXP_RESET register. This reboots the System Monitor.	43
reflash	Writes the entire flash with an image located in main memory.	44
set	Sets the system configuration.	45
show	Displays system information and status.	46

To display available commands, enter the following:

```
help load
```

The System Monitor displays available commands:

```
Available Commands:
help <command>
boot [address|flash]
configure [21555]
delete [flash-address]
poke [address] [data] <count>
peek [address] <count>
load [serial|flash]
program [flash address] [image size] [image location]
reboot
reflash [image location]
set [baud|width] [value]
show [cpuspeed|flash|gpio|ixf440|memory|pci|rom|version]
impi [set|get]
test [all|sdram|sram|scratch|flash|pci|leds|ethernet|teeprom|macaddrs|
   21555|ixf440|TDM2IX|t8105|H110|cpld]
enp>
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

delete

Erases the flash block starting at the specified address.

Syntax

```
delete FlashAddress
```

Parameters

FlashAddress

The start address of the flash block you want to erase. You must supply a valid starting address of a flash block.

Additional information

For more information about flash block size for the supported devices, see the ENP board's documentation.



This command does not erase the flash's bootblock area (the first 64KB).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ipmi

Provides support for the IPMI controller.

Syntax

`impi cmd type`

Parameters

cmd Specifies the command. You can use these values:

- `set` Generates an IPMI set command.
- `get` Generates an IPMI get command.

type The IPMI command. You can use these commands:

`led number state`

where:

- *number* Identifies the LED number:
  - 0Blue (Hot Swap) LED
  - 1Red (fail) LED
  - 2Green LED
- *state* Specifies the LED's current state:
  - 0Off
  - 1On
  - 255Auto

  You can use this option only with the `set` command.

- `deviceid` Returns the IPMI status and version information. This command valid only when used with the *ipmi get* command.
- `selftest` Returns the IPMI selftest status. This command valid only when used with the *ipmi get* command. The current IPMI firmware does not support this feature.
- `sensor` Gets the set up of the various IPMI sensor status. The current IPMI firmware does not support this feature.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

load

Loads an image from the network, serial port, or flash memory into main memory.

Syntax

`load type item`

Parameters

type Specifies the file type you want to load. You can use these values:

- `network` Downloads a file from the network using the bootp/tftp protocols to the memory location specified in the command. The System Monitor does not currently support this feature.
- `flash` Downloads a file from flash memory to the specified main memory address.
- `serial` Downloads a file from the serial port using the XMODEM transfer protocol to the memory location specified in the command. The host system must run a corresponding program. Both Minicom (for Linux) and Hypertem (for Windows NT) are confirmed to work.

item Specifies the item you want to load. You can use these values:

- *FileName* The name of the file you want to download. You can use this only when the *type* value is `network`.
- *MemoryAddress* The address you want to download to. Required for all values of *type*.
- *FileSize* The size of the file you want to download. You can use this only when the *type* value is `network`.
- *FlashAddress* The address you want to download the flash image to. You can use this only when the *type* value is `flash`.
- *ImageSize* The size of the flash image file you want to download. You can use this only when the *type* value is `flash`.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

peek

Displays contents of the specified address.

You can access any valid address within the IXP1200 address space. The supplied address is a physical address. The Memory Management Unit is configured for one-to-one address translation.

Set access width using the 'set width' command.

Syntax

```
peek address count
```

Parameters

address	The first address to examine. Prefix this parameter with 0x for hexadecimal format.
count	The number of subsequent addresses to read.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

poke

Writes the data into the specified address.

You can access any valid address within the IXP1200 address space. The supplied address is a physical address. The Memory Management Unit is configured for one-to-one address translation.

Set access width using the 'set width' command.

Syntax

```
poke address data count
```

Parameters

address	The first address to deposit the specified data. For hexadecimal format, prefix this parameter with 0x.
data	The data to write. For hexadecimal format, prefix this parameter with 0x.
count	The number of subsequent addresses to write.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

program

Programs an image located at the specified memory address into the specified flash address.



You must use the start address of a flash block *outside* the flash memory's boot block area (the first 64KB).

Syntax

```
program FlashAddress FileSize MemoryAddress
```

Parameters

FlashAddress
: The flash address of the block you want to program. The supplied flash address must be the starting address of a flash block and a valid flash address, otherwise the program command aborts.

FileSize
: The image size you want to program.

MemoryAddress
: The memory address of the image you want to program.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

reboot

Generates a full system-wide reset system using the IXP_RESET register. This reboots the System Monitor.

Syntax

```
reboot
```

Parameters

None.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

reflash

Writes the entire flash with an image located in main memory.

For each block in the flash array, the image is compared against the flash chip's existing contents. If the block is unchanged in the new image, the flash block does not update. This dramatically reduces the time required to reflash an image. The system must reboot for the new code to take effect.



Use caution when invoking this command. This command doesn't overwrite flash memory's boot block area (the first 64KB).

Do not power down your system while this command is running

Syntax

`reflash` *`MemoryAddress`*

Parameters

`MemoryAddress`

The memory address of the image you want to program.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

set

Sets the system configuration.

Syntax

`set item`

Parameters

item Specifies the item you want to configure. You can use one of these:

`baudrate` *speed*

Sets the UART baudrate to the rate you specify. You can set *speed* to one of these:

9600 (default)
19200
38400
57600
115200

`width` *type*

Sets the access width for examine and deposit command. You can set *type* to one of these:

`byte`

`word` (four bytes)

`longword` (default; the system resets to this width after powerup.)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

show

Displays system information and status.

Syntax

`show` *`info`*

Parameters

`info` Specifies the information type you want to display. You can use one of these:

`flash`	Displays the flash memory size and type.
`width`	Displays the access width used for the peek and poke commands. The default width is longword wide (four bytes).
`pci`	Enumerates the PCI bus.
`version`	Displays the monitor version, plus the build date and time.
`rom`	Displays versions and build dates of the ROM contents.
`memory`	Displays the size of installed memory.
cpuspeed	Displays the system's configured CPU speed.
`ixf440`	Displays the ISF440 register contents, including the PHY registers.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

6 Boot Manager software

This chapter describes the Boot Manager application. You use the Boot Manager to specify the boot OS from those available in the ENP board's flash chip.

This application is available only when specified via jumper settings on your ENP board. For information about setting jumpers, see *Configuring the ENP board for boot* on page 10.

Booting your ENP system

To boot your system:

1. Set the ENP board's switches or jumpers to the Boot Manager setting. For information about jumper or switch settings for your ENP board, see the hardware reference for that board.
2. Set your COM port to:

Baud rate:	38400
Parity:	N (no parity)
Data bits:	8
Stop bits:	1
Flow control:	N (no flow control)

3. Power on your ENP board. The Boot Manager starts and displays this on the serial port (at 38400 Baud):

   ```
   Press space bar to stop auto-boot...
   10
   ```

   The numerical value counts down to zero. If it reaches zero before you press the space bar, the current default operating system boots.

4. Press the space bar. The Boot Manager prompt displays:

   ```
   [BootMgr]:
   ```

You can then enter commands. You can recall previous commands by pressing the up and down arrow keys. You can also shorten command and keywords, when not ambiguous.

For a detailed description of these commands, continue to the next section.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Boot Manager commands

These commands manage your ENP system.

Use this table to identify the Boot Manager commands you want to use. Use the command description later in this chapter to obtain detailed information, including syntax and parameter values.

Table 6-1. Boot Manager commands

Command	Description	Page
b (boot)	Boots the OS you specify.	48
c (change)	Changes boot parameters.	48
h (help)	Lists all Boot Manager commands.	49
p (print)	Prints boot parameters.	49

b (boot)

Boots the OS you specify. Select this option when you want to change boot parameters for only this boot. Your boot parameters are not saved. You can use this, for example, to boot the flash utility without modifying the default OS.

To change boot parameters, enter this command at the Boot Manager command prompt as shown:

```
[BootMgr]: b [n]
```

where n is the number of the OS you want to boot when the countdown reaches 0 (zero). You can enter only listed OS numbers.

Enter this command with no arguments to boot the default OS.

c (change)

Changes boot parameters. Select this option when you want to change existing and boot parameters and save them.

To change boot parameters, enter this command at the Boot Manager command prompt as shown:

```
[BootMgr]: c
```

When you enter this command, the current parameters display.

```
BootMgr Version 1.0
OS list:
   0 Flash Utility
   1 Monitor
   2 Reserverd
   3 VxWorks
   4 OS-9
Countdown value: 10
Disable initial display: 0
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

You are then prompted for new values:

Default OS:

The OS that boots if you do not press the spacebar before the countdown ends. You can set the default to any OS in the Boot Manager OS list. You can enter one of these:

n	The number of the OS you want to boot. You can enter only listed OS numbers. The default OS is Vxworks (2).
Return	Press the Return key to leave the value unchanged.

Countdown value:

The number of seconds the Boot Manager waits before autobooting the default OS. You can enter one of these:

n	Any number from 1 to 60 (seconds). The default is 15. A small value means less waiting time before the boot, while a large value results in ample decision time.
Return	Press the Return key to leave the value unchanged.

Disable initial display:

The countdown display location. You can enter one of these:

0	(Default) The autoboot banner and countdown displays on the workstation console.
1	The autoboot countdown does not display. No messages display unless you interrupt autoboot by pressing the spacebar.
Return	Press the Return key to leave the value unchanged.

h (help)

Lists all Boot Manager commands.

To list commands, enter this at the Boot Manager command prompt as shown:

```
[BootMgr]: h
```

When you enter the help command, the following displays:

```
BootMgr commands:
p : Print boot parameters
c : Change boot parameters
b : Boot with current parameters
b <os> : Boot given os without changing parameters
h : Print this help message
```

p (print)

Prints (lists) boot parameters. Select this option to verify that the Boot Manager is configured the way you want.

To print boot parameters, enter this command at the Boot Manager command prompt as shown:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
[BootMgr]: p
```

When you enter this command, the following displays:

```
BootMgr Version 1.0
CPU Revision 6901C121
OS list:
    0 Flash Utility
    1 Monitor
    2 Reserverd
    3 VxWorks
    Default OS: 3
Countdown value: 10
Disable initial display: 0
```

To change the value of these parameters, use the Boot Manager's C (change) command.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

7 PCI communications

This document describes PCI communication functions.

You can find a function in this chapter either by locating it alphabetically within its group, or by using Table 7-1 and Table 7-2 on page 55.

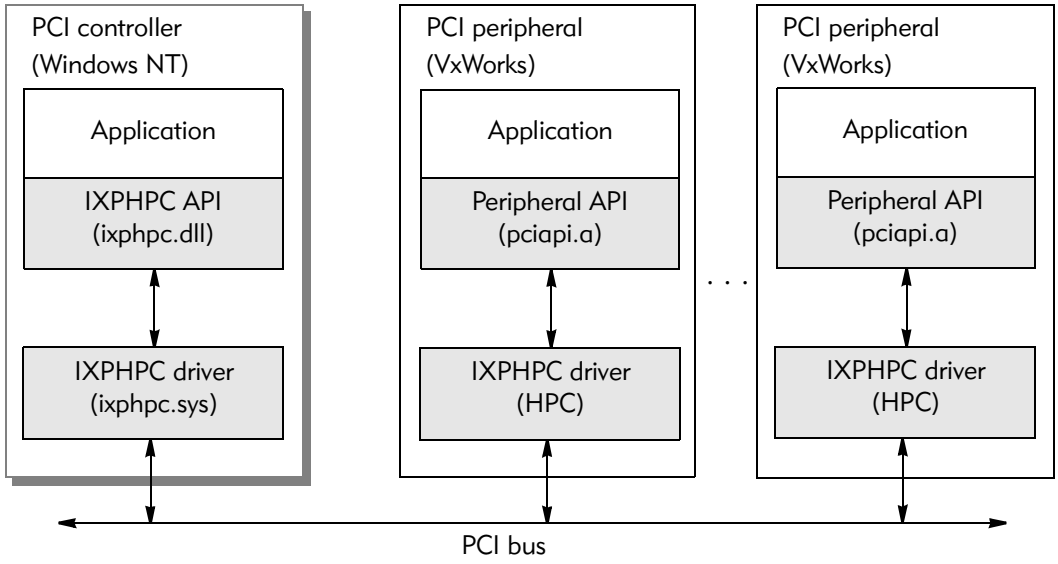
Overview

The Internet Exchange Architecture (IXA) PCI Communications software supports PCI-based communication between applications executing on a PCI controller and applications executing on multiple PCI peripherals.

IXA PCI Communications software includes:

- **PCI controller components**: Software that executes on the x86-based PCI controller.
- **PCI peripheral components**: One or more IXP12xx-based PCI peripherals.

Figure 7-1. IXA PCI communications software architecture



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PCI controller components

IXA PCI communications controller components are specific to the operating system upon which they execute. For the current release, Windows NT 4.0 (Service Pack 6) is the target operating system.

For Windows NT, IXA PCI communications components include a Windows NT device driver and a Windows NT application interface (API). Files and registry data constitute the components.



The ixphpc.sys driver provided in the ENP SDK is a RadiSys component that replaces (overwrites) the Intel ixphpc.sys driver provided in the IXA SDK.

If transactor debug via the PCI bus should fail after installing ENP SDK, contact RadiSys technical support as described in *Where to get more information* on page v.

Windows NT device driver

The Windows NT device driver (ixphpc.sys) functions as a driver-level extension of the Windows NT API. Under the direction of the Windows NT API, the driver directly manipulates and receives interrupts from the controller's underlying PCI hardware.

The Windows NT device driver reads peripheral device specification data from the registry, identifies the peripheral devices present on the PCI bus that match the specification, and allocates the resources required to communicate with the corresponding device driver executing on each peripheral device. Once low-level communication is established between the drivers on the controller and peripheral devices, the Windows NT device driver abstracts each peripheral, creating multiple bi-directional "channels" that correspond to each peripheral. Controller applications use channels to communicate with peripheral applications executing on a particular peripheral device.

The API supports concurrent use by multiple Windows NT processes and threads.

Windows NT API

The Windows NT API (ixphpc.dll, ixphpc.h, and ixphpc.lib) is a Windows NT dynamic link library (DLL), include file, and export library combination defining an API that applications can use to communicate over the PCI bus with applications that execute on a PCI peripheral device.

Windows NT files

The following files contain the Windows NT functions:

- System files:

  ```
  winnt\system32\drivers\ixphpc.sys
  <installation-directory>/Host/WinNT/lib/ixphpc.dll
  ```

- Windows NT PCI API user application functions:

  ```
  <installation-directory>/Host/WinNT/include/ixphpc.h
  <installation-directory>/Host/WinNT/lib/ixphpc.lib
  ```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

<installation-directory> represents the user-specified installation directory of the ENP SDK 2.0 product (%SystemDrive%\RadiSys\ENP SDK 2.0, by default).

%SystemDrive% is a Windows NT environment variable specifying the root directory of the volume that contains Windows NT (C:\, by default).

PCI peripheral components

VxWorks driver

The VxWorks PCI device driver initializes and maintains a software PCI mechanism providing peripheral to NT host (and vice-versa) communication. The device driver:

- Sets up downstream memory window sizes.
- Sets up global channel descriptor tables in PCI memory space.
- Connects an ISR to service the IXP1200 doorbell interrupt.
- Sets up the IXP1200 doorbell register in PCI space to trigger reading and writing.
- Spawns a task for processing requests for reading and writing across the PCI bridge.

VxWorks API

The VxWorks PCI API provides basic functions for communication with a Windows NT host application that uses the IXPHPC constructs described in *Windows NT API* on page 52. The underlying mechanism is the HPC interface, which is compatible with the IXPHPC NT device driver.

The VxWorks PCI API uses the HPC interface to handle low-level I/O. The API sources are provided as part of the ENP SDK. The VxWorks PCI API provides the basic Open, Close, Read, and Write functions that applications can use where basic PCI communication is required.

VxWorks files

The following files contain the VxWorks HPC and PCI API functions:

- 21555 non-transparent bridge initialization:

```
<installation-directory>\Boardsupport\ENPxxxx\21555drv.h
<installation-directory>\Boardsupport\ENPxxxx\21555drv.c
```

- HPC Initialization code and support functions:

```
<installation-directory>\Boardsupport\ENPxxxx\HPCif.h
<installation-directory>\Boardsupport\ENPxxxx\IXP1200HPC.h
<installation-directory>\Boardsupport\ENPxxxx\IXP1200HPC.c
```

- VxWorks PCI API user application functions:

```
<installation-directory>\include\ENPPCIAPI_VX.h
<installation-directory>\lib\pcilib.a
```

<installation-directory> represents the user-specified installation directory of the ENP SDK 2.0 product (%SystemDrive%\RadiSys\ENP SDK 2.0, by default).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

`%SystemDrive%` is a Windows NT environment variable specifying the root directory of the volume that contains Windows NT (C:\, by default).

VxWorks configuration

The optional inclusion of the VxWorks PCI driver and API is done via the Makefile definition:

`INCLUDE_HPC`

Undefining `INCLUDE_HPC` disables all HPC-related VxWorks PCI mechanisms within the VxWorks BSP. Consequently, not defining this variable disables PCI workbench debugging. This definition is, by default, defined.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function list

Use this table to identify the ENP functions you want to use. Use the function description later in this chapter to obtain detailed information, including syntax and parameter values.

Windows NT API

Table 7-1. PCI communication functions: Windows NT API

Function	Description	Page
IXPHPCAsyncOpen	Opens a channel for asynchronous I/O and returns a handle to the open channel.	56
IXPHPCAsyncRead	Asynchronously reads data from a channel.	58
IXPHPCAsyncWrite	Asynchronously writes data to a channel.	60
IXPHPCClose	Closes a channel.	62
IXPHPCEnumerate	Enumerates the PCI peripheral devices present in the system.	63
IXPHPCReset	Resets a PCI peripheral device.	65
IXPHPCSyncOpen	Opens a channel for synchronous I/O and returns a handle to the open channel.	66
IXPHPCSyncRead	Synchronously reads data from a channel.	68
IXPHPCSyncWrite	Synchronously writes data to a channel.	69

VxWorks API

Table 7-2. PCI communication functions: VxWorks API

Function	Description	Page
ENPClosePCIChannel	Closes a previously opened PCI channel.	71
ENPOpenPCIChannel	Opens a PCI channel for reading or writing.	72
ENPPCIAPIPrintError	Prints the status code.	73
ENPReadFromPCIChannel	Reads data from a PCI channel.	74
ENPWriteToPCIChannel	Writes data to a PCI channel.	75

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Windows NT API

IXPHPCAsyncOpen

Opens a channel for asynchronous I/O and returns a handle to the open channel.

Syntax

```
#include "ixphpc.h"

IXPHPC_API
IXPHPCAsyncOpen(
   UBIT32    DeviceIndex,
   UBIT32    ChannelIndex,
   HANDLE *  HandlePtr
);
```

Parameters

DeviceIndex
Specifies a PCI peripheral device capable of IXPHPC communications. Use IXPHPCEnumerate to query an array of all the PCI peripheral devices in the system. Valid values include 0 to (IXPHPC_DEVICE_CNT – 1).

ChannelIndex
Specifies a channel of the PCI peripheral device. Use IXPHPCEnumerate to determine the channels available for a particular PCI peripheral device. Valid values include 0 to (IXPHPC_CHANNEL_CNT – 1).

HandlePtr Specifies a memory location where a handle to the open channel is placed. *HandlePtr* must reference a valid memory location; you cannot use NULL.

Return value

ERROR_SUCCESS	The function completed successfully.
ERROR_INVALID_PARAMETER	The specified *DeviceIndex* and/or the specified *ChannelIndex* is invalid or the specified *HandlePtr* is NULL.



If a supporting Microsoft Platform SDK function fails, this function may also return other status values.

Comments

To read from or write to channels open for asynchronous I/O, you must use only asynchronous I/O functions (IXPHPCAsyncRead or IXPHPCAsyncWrite).

Once open, a channel's characteristics (such as whether it supports asynchronous or synchronous I/O) are fixed. Channels open for asynchronous I/O by one process or thread can be opened only for asynchronous I/O. Conversely, channels open for synchronous I/O can be opened by a process or thread only for synchronous I/O.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If this function fails for any reason, the value at the memory location specified by *HandlePtr* is undefined.



If you plan to use PCI debugging within the Intel IXA SDK workbench tool, avoid using channel 0 in your application to prevent conflicts.The Intel IXA SDK workbench tool uses channel 0 for its PCI communications.

See also

IXPHPCAsyncRead, IXPHPCAsyncWrite, IXPHPCClose, and IXPHPCEnumerate.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPHPCAsyncRead

Asynchronously reads data from a channel.

Syntax

```
#include "ixphpc.h"

IXPHPC_API
IXPHPCAsyncRead(
   HANDLE                           Handle,
   UBIT08                           * BufferPtr,
   UBIT32                           BufferSize,
   LPOVERLAPPED                     OverlappedPtr,
   LPOVERLAPPED_COMPLETION_ROUTINE FunctionPtr
);
```

Parameters

Handle Specifies the handle of a channel open for asynchronous I/O.

BufferPtr Specifies the memory buffer where you want to place data read from the channel. You must reference a valid memory location; you cannot use NULL.

BufferSize
Specifies the maximum number of bytes of data to read.

OverlappedPtr
Specifies the memory location of an OVERLAPPED structure used to control the asynchronous I/O operation. You must reference a valid memory location; you cannot use NULL.

FunctionPtr
Specifies the memory location of an asynchronous I/O completion function. You must references a valid memory location; you cannot use NULL.

Return value

ERROR_SUCCESS	The function completed successfully.
ERROR_ACCESS_DENIED	The specified channel has been opened for synchronous I/O only.
ERROR_INVALID_HANDLE	The specified Handle does not refer to an open channel.
ERROR_INVALID_PARAMETER	The specified *BufferPtr* and/or *OverlappedPtr* and/or *FunctionPtr* is NULL.



If a supporting Microsoft Platform SDK function fails, this function may also return other status values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

This function reads up to *BufferSize* bytes of data into the memory buffer specified by *BufferPtr*, then calls the completion function specified by *FunctionPtr*.

This function reads data only from a channel opened with IXPHPCAsyncOpen. You cannot perform asynchronous I/O using a channel opened with IXPHPCSyncOpen.



For a complete description of asynchronous I/O, including descriptions of the LPOVERLAPPED and LPOVERLAPPED_COMPLETION_ROUTINE types, OVERLAPPED structure, and asynchronous I/O completion functions, see ReadFileEx documentation in the Microsoft Platform SDK.

See also

IXPHPCAsyncOpen, IXPHPCAsyncWrite, and IXPHPCClose.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPHPCAsyncWrite

Asynchronously writes data to a channel.

Syntax

```
#include "ixphpc.h"

IXPHPC_API
IXPHPCAsyncWrite(HANDLE                         Handle,
   UBIT08                                       * BufferPtr,
   UBIT32                                       BufferSize,
   LPOVERLAPPED                                 OverlappedPtr,
   LPOVERLAPPED_COMPLETION_ROUTINE              FunctionPtr
);
```

Parameters

Handle Specifies the handle of a channel open for asynchronous I/O.

BufferPtr Specifies a memory buffer that contains data to write to the specified channel. You must reference a valid memory location; you cannot use NULL.

BufferSize
Specifies the number of bytes of data in the memory buffer specified by *BufferPtr*.

OverlappedPtr
Specifies the memory location of an OVERLAPPED structure used to control the asynchronous I/O operation. You must reference a valid memory location; you cannot use NULL.

FunctionPtr
Specifies the memory location of an asynchronous I/O completion function. You must reference a valid memory location; you cannot use NULL.

Return value

ERROR_SUCCESS	The function completed successfully.
ERROR_ACCESS_DENIED	The specified channel is open only for synchronous I/O.
ERROR_INVALID_HANDLE	The specified *Handle* does not refer to an open channel.
ERROR_INVALID_PARAMETER	The specified *BufferPtr* and/or *OverlappedPtr* and/or *FunctionPtr* is NULL.



If a supporting Microsoft Platform SDK function fails, this function may also return other status values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comments

This function writes up to *BufferSize* bytes of data from the memory buffer specified by *BufferPtr*, then calls the completion function specified by *FunctionPtr*.

This function writes data only to a channel opened with IXPHPCAsyncOpen. You cannot perform asynchronous I/O using a channel opened with IXPHPCSyncOpen.



For a complete description of asynchronous I/O, including descriptions of the LPOVERLAPPED and LPOVERLAPPED_COMPLETION_ROUTINE types, OVERLAPPED structure, and asynchronous I/O completion functions, see WriteFileEx documentation in the Microsoft Platform SDK.

See also

IXPHPCAsyncOpen, IXPHPCAsyncRead, and IXPHPCClose.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPHPCClose

Closes a channel.

Syntax

```
#include "ixphpc.h"

IXPHPC_API
IXPHPCClose(
   HANDLE    Handle
);
```

Parameters

Handle Specifies the handle of an open channel.

Return value

ERROR_SUCCESS	The function completed successfully.
ERROR_INVALID_HANDLE	The specified *Handle* does not refer to an open channel.



If a supporting Microsoft Platform SDK function fails, this function may also return other status values.

Comments

This function closes the channel specified by *Handle*. It closes channels open for either asynchronous or synchronous I/O.

See also

IXPHPCAsyncOpen and IXPHPCSyncOpen.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPHPCEnumerate

Enumerates the PCI peripheral devices present in the system.

Syntax

```
#include "ixphpc.h"

IXPHPC_API
IXPHPCEnumerate(
   struct IXPHPCDeviceData   * BufferPtr,
   UBIT32                    * BufferSizePtr
);
```

Parameters

BufferPtr Specifies the memory buffer where you want to place the device descriptor structures. Use one of these values:

n	The memory buffer's address.
NULL	Places the required buffer size, in bytes, in the memory location specified by *BufferSizePtr* and returns ERROR_INSUFFICIENT_BUFFER.

BufferSizePtr
Specifies a memory location that contains the specified buffer's size in bytes. *BufferSizePtr* must reference a valid memory location. It may not be NULL.

Return value

ERROR_SUCCESS	The function completed successfully.
ERROR_INSUFFICIENT_BUFFER	The specified buffer is not large enough to hold all the system's PCI peripheral device configuration data.
ERROR_INVALID_PARAMETER	The specified *BufferSizePtr* is NULL.



If a supporting Microsoft Platform SDK function fails, this function may also return other status values.

Comments

This function enumerates the PCI peripheral devices in the system that can support IXPHPC communications, stores the device configuration data in the memory buffer specified by *BufferPtr*, and returns the number of bytes stored in the buffer at the memory location specified by *BufferSizePtr*.

This function stores PCI peripheral device data in the specified buffer as an array of IXPHPCDeviceData structures. The index of a device in the array corresponds to

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

the *DeviceIndex* parameter used in the IXPHPCAsyncOpen and IXPHPCSyncOpen functions.

The IXPHPCDeviceData structure contains a channels array. The index of a channel in the array corresponds to the *ChannelIndex* parameter used in the IXPHPCAsyncOpen and IXPHPCSyncOpen functions.

When the specified buffer is not large enough to hold all of the system's PCI peripheral device enumeration data, this function places the required buffer size, in bytes, in the memory location specified by *BufferSizePtr* and returns ERROR_INSUFFICIENT_BUFFER.

If the function fails for any reason, data contained in the memory buffer specified by *BufferPtr* is undefined.

If the function fails for reasons other than insufficient buffer space, data contained in the memory location specified by *BufferSizePtr* is undefined.

See also

IXPHPCAsyncOpen and IXPHPCSyncOpen.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPHPCReset

Resets a PCI peripheral device.

Syntax

```
#include "ixphpc.h"

IXPHPC_API
IXPHPCReset(
   HANDLE    Handle
);
```

Parameters

Handle Specifies the handle of an open channel.

Return value

ERROR_SUCCESS	The function completed successfully.
ERROR_ACCESS_DENIED	The PCI peripheral device has other open channels.
ERROR_INVALID_HANDLE	The specified *Handle* does not refer to an open channel.



If a supporting Microsoft Platform SDK function fails, this function may also return other status values.

Comments

This function resets the PCI peripheral device specified by *Handle*. It resets the PCI peripheral device regardless of whether the specified channel is open for asynchronous I/O or synchronous I/O.

If any of the PCI peripheral device's other channels are open, this function does not reset the PCI peripheral device and returns an ERROR_ACCESS_DENIED status value.

If the specified channel has pending asynchronous I/O, this function cancels the I/O before proceeding with the reset operation.

Upon successful completion, the specified *Handle* is no longer valid.

See also

IXPHPCAsyncOpen and IXPHPCSyncOpen.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPHPCSyncOpen

Opens a channel for synchronous I/O and returns a handle to the open channel.

Syntax

```
#include "ixphpc.h"

IXPHPC_API
IXPHPCSyncOpen(
   UBIT32    DeviceIndex,
   UBIT32    ChannelIndex,
   HANDLE    * HandlePtr
);
```

Parameters

DeviceIndex
: Specifies a PCI peripheral device capable of IXPHPC communication. Use IXPHPCEnumerate to query an array of all the PCI peripheral devices in the system. Valid values include 0 to (IXPHPC_DEVICE_CNT – 1).

ChannelIndex
: Specifies a channel of the PCI peripheral device. Use IXPHPCEnumerate to determine the channels available for a particular PCI peripheral device. Valued values include 0 to (IXPHPC_CHANNEL_CNT – 1).

HandlePtr
: Specifies the memory location where you want to place the open channel's handle. You must reference a valid memory location; you cannot use NULL.

Return value

ERROR_SUCCESS	The function completed successfully.
ERROR_INVALID_PARAMETER	The specified *DeviceIndex* and/or the specified *ChannelIndex* is invalid or the specified *HandlePtr* is NULL.



If a supporting Microsoft Platform SDK function fails, this function may also return other status values.

Comments

A channel open for synchronous I/O can only be read from or written to using synchronous I/O functions (IXPHPCSyncRead, IXPHPCSyncWrite).

Once open, the characteristics of that channel (such as whether it supports asynchronous or synchronous I/O) are fixed. Channels open for asynchronous I/O by one process or thread can be opened only for asynchronous I/O. Conversely, channels open for synchronous I/O can be opened by a process or thread only for synchronous I/O.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If this function fails for any reason, the value at the memory location specified by *HandlePtr* is undefined.



If you plan to use PCI debugging within the Intel IXA SDK workbench tool, avoid using channel 0 in your application to prevent conflicts.The Intel IXA SDK workbench tool uses channel 0 for its PCI communications.

See also

IXPHPCClose, IXPHPCEnumerate, IXPHPCSyncRead and IXPHPCSyncWrite.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPHPCSyncRead

Synchronously reads data from a channel.

Syntax

```
#include "ixphpc.h"

IXPHPC_API
IXPHPCSyncRead(
   HANDLE    Handle,
   UBIT08    * BufferPtr,
   UBIT32    * BufferSizePtr
);
```

Parameters

Handle Specifies the handle of a channel open for synchronous I/O.

BufferPtr Specifies a memory buffer where data read from the channel is placed. You must reference a valid memory location; you cannot use NULL.

BufferSizePtr
Specifies a memory location that contains the maximum number of bytes of data to read. You must reference a valid memory location; you cannot use NULL.

Return value

ERROR_SUCCESS	The function completed successfully.
ERROR_ACCESS_DENIED	The specified channel is open only for asynchronous I/O.
ERROR_INVALID_HANDLE	The specified *Handle* does not refer to an open channel.
ERROR_INVALID_PARAMETER	The specified *BufferPtr* and/or *BufferSizePtr* is NULL.



If a supporting Microsoft Platform SDK function fails, this function may also return other status values.

Comments

This function reads up to *BufferSizePtr* bytes of data from the channel specified by *Handle* into the memory buffer specified by *BufferPtr*, places the number of bytes actually read into *BufferSizePtr* and returns to the caller.

This function reads data only from a channel opened with IXPHPCSyncOpen. You cannot perform aynchronous I/O using a channel opened with IXPHPCAsyncOpen.



For a complete description of synchronous I/O, see ReadFile documentation in the Microsoft Platform SDK.

See also

IXPHPCSyncOpen, IXPHPCSyncWrite, and IXPHPCClose.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPHPCSyncWrite

Synchronously writes data to a channel.

Syntax

```
#include "ixphpc.h"

IXPHPC_API
IXPHPCSyncWrite(
   HANDLE    Handle,
   UBIT08    * Buffer_Ptr,
   UBIT32    * Buffer_Size_Ptr
);
```

Parameters

Handle Specifies the handle of a channel open for synchronous I/O.

BufferPtr Specifies a memory buffer that contains data you want to write to the specified channel. You must reference a valid memory location; you cannot use NULL.

BufferSizePtr
Specifies a memory location that contains the number of bytes of data in the memory buffer specified by *BufferPtr*. You must reference a valid memory location; you cannot use NULL.

Return value

ERROR_SUCCESS	The function completed successfully.
ERROR_ACCESS_DENIED	The specified channel is open only for asynchronous I/O.
ERROR_INVALID_HANDLE	The specified Handle does not refer to an open channel.
ERROR_INVALID_PARAMETER	The specified *BufferPtr* and/or *BufferSizePtr* is NULL.



If a supporting Microsoft Platform SDK function fails, this function may also return other status values.

Comments

This function writes up to the number of bytes of data specified by *BufferPtr* and *BufferSizePtr* to the channel specified by *Handle*, places the number of bytes actually written in the location specified by *BufferSizePtr* and returns to the calling function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This function writes data only to a channel opened with IXPHPCSyncOpen. You cannot perform synchronous I/O using a channel opened with IXPHPCAsyncOpen.



For a complete description of synchronous I/O, see WriteFile documentation in the Microsoft Platform SDK.

See also

IXPHPCSyncOpen, IXPHPCSyncRead, and IXPHPCClose.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

VxWorks API

ENPClosePCIChannel

Closes a previously opened PCI channel.

Syntax

```
#include "ENPPCIAPI_VX.h"

HPC_ERROR_CODE
ENPClosePCIChannel(
   void      * Handle
);
```

Parameters

Handle Specifies the channel to close.

Return value

HPC_OK	The function completed successfully.
HPC_NOT_OPENED	The function failed to close the channel.

Comments

This function tries to close a PCI channel previously opened by ENPOpenPCIChannel. You must call this function to release the channel represented by *Handle* for use by other processes or threads. Any attempt to close an invalid *Handle* results in an HPC error.

See also

ENPOpenPCIChannel

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENPOpenPCIChannel

Opens a PCI channel for reading or writing.

Syntax

```
#include "ENPPCIAPI_VX.h"

HPC_ERROR_CODE
ENPOpenPCIChannel(
   unsigned int    Channel,
   void            ** Handle
);
```

Parameters

Channel	Specifies the channel to open for reading or writing. Valid values include 0 to (MAX_CHANNELS – 1).
Handle	If the call is successful, contains a unique handle for the opened channel.

Return value

HPC_OK	The function completed successfully.
HPC_ALREADY_OPENED	The channel specified is already open for I/O.
HPC_INTERNAL_ERROR	An internal system error occurred attempting to open channel.
HPC_INVALID_CHANNEL	The specified channel is out of the range of acceptable values.
HPC_NOT_OPENED	The function failed to fully open the channel.

Comments

This function tries to open a PCI channel for reading or writing. You must call this function to open the requested channel. Any attempt to read or write to an unopened channel results in an HPC error.

Once opened, a channel remains open until the application calls ENPClosePCIChannel with the *Handle* received from this function. If for any reason this function fails, the value of *Handle* is undefined.

See also

ENPClosePCIChannel

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENPPCIAPIPrintError

Prints the status code.

Syntax

```
#include "ENPPCIAPI_VX.h"

void
ENPPCIAPIPrintError (
   HPC_ERROR_CODE  ErrorCode
);
```

Parameters

ErrorCode Specifies error code message to print.

Return value

None

Comments

This function prints a meaningful message regarding *ErrorCode* to STDOUT.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENPReadFromPCIChannel

Reads data from a PCI channel.

Syntax

```
#include "ENPPCIAPI_VX.h"

HPC_ERROR_CODE
ENPReadFromPCIChannel (
   void            * Handle,
   unsigned char   * BufferPtr,
   unsigned int    * BufferLen,
   unsigned int    * Timeout
);
```

Parameters

Handle Specifies the channel to read. An invalid *Handle* results in an HPC error.

BufferPtr Specifies where to store data. You cannot use NULL.

BufferLen Specifies the number of bytes to read. After completion, *BufferLen* contains the actual number of bytes read over PCI. You cannot use NULL.

Timeout Use one of these values:

n	The number of milliseconds to wait for a write request from the controller.
0	Wait indefinitely until the read request is fulfilled.

Return value

HPC_OK	The function completed successfully.
HPC_WRONG_MODE	The channel is opened only for I/O.
HPC_NOT_OPENED	Invalid handle specified.
HPC_TIMEOUT	The number of milliseconds specified in *Timeout* have elapsed.

Comments

This function tries to read *BufferLen* bytes from the channel represented by *Handle*. After completion, *BufferLen* updates to reflect the actual number of bytes read.

If *BufferLen* bytes of data are not immediately received, this function waits up to *Timeout* milliseconds for data to be received.

See also

ENPWriteToPCIChannel

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENPWriteToPCIChannel

Writes data to a PCI channel.

Syntax

```
#include "ENPPCIAPI_VX.h"

HPC_ERROR_CODE
ENPWriteToPCIChannel(
   void            * Handle,
   unsigned char   * BufferPtr,
   unsigned int    * BufferLen,
   unsigned int    Timeout
);
```

Parameters

Handle Specifies the channel to write to.

BufferPtr Specifies where to read data from. You cannot use NULL.

BufferLen Specifies the number of bytes to write. After completion, *BufferLen* contains the actual number of bytes written over PCI. You cannot use NULL.

Timeout

n	Specifies the number of milliseconds to wait for a read request from the controller.
0	Wait indefinitely until the write request is fulfilled.

Return value

HPC_OK	The function completed successfully.
HPC_WRONG_MODE	The channel is opened only for I/O.
HPC_NOT_OPENED	Invalid handle specified.
HPC_TIMEOUT	The number of milliseconds specified in *Timeout* have elapsed.

Comments

This function tries to write *BufferLen* bytes from the channel represented by *Handle*. After completion, *BufferLen* updates to reflect the actual number of bytes written.

If *BufferLen* bytes of data are not immediately transmitted, ENPWriteToPCIChannel waits for up to *Timeout* milliseconds for data to transmit.

Any attempt to write to an invalid *Handle* results in an HPC error.

See also

ENPReadFromPCIChannel

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

8 TDM routing and switching library

This describes the TDM switching library. This chapter assumes you understand the TDM device family and the H.100/H.110 bus.

This library encapsulates and abstracts call setup and teardown for ARM code. It includes the TDM AAPI, which includes functions that create, destroy, and configure TDM channels.

You can find a function in this chapter either by locating it alphabetically within its group, or by using Table 8-2, which groups like functions together.

Overview

The TDM switching library:

- Provides a hardware-independent interface to switch TDM data between the local and TDM bus, supporting:
  - Local to local bus
  - Local to CT_bus
  - CT_Bus to CT_bus
- Configures the TDM switching device to serve as one of these:
  - An H.110 master.
  - An H.110 slave.
- Configures the stream speed of different stream groups (H.110 and local CHI).
- Provides functions that test and verify the underlying TDM device.

Implementation

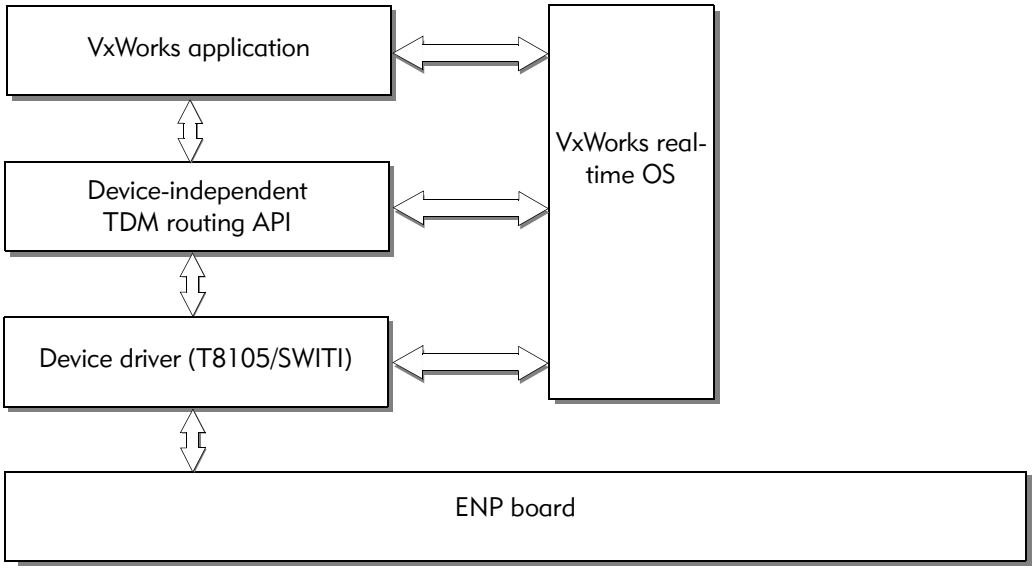
The API is under the VxWorks real-time operating system and uses a device driver to control the underlying switching device. The driver is not a true VxWorks driver as it isn't built as part of the kernel. However, it does use VxWorks system functions and therefore requires the OS.

The API is a hardware-independent abstraction of the underlying switching device's capabilities and provides the application developer with a simple, easy to use and configure the device. The API is a VxWorks library module which links with VxWorks applications.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Architecture

Figure 8-1. Programming model



Concepts

- **H.110 bus**: 32 bi-directional streams, operating at 8.192 MHz, each consisting of 128 timeslots. The maximum number of timeslots supported on H.110 bus is 4096.
- **T8105**: Serves as the ENP board's timeslot switching device. It can perform these switch connections types:
  - Local-to-local
  - Local-to-CT_BUS
  - CT_BUS-to-CT_BUS

  The maximum number of connections in all possible combinations is limited to 1024. The maximum number of Local-to-CT_BUS connections is 512, which leaves another 512 connections left for local use.

  The T8105 groups the 16 local streams as follows:

  - A (0–7)
  - B (8–11)
  - C (11–15)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

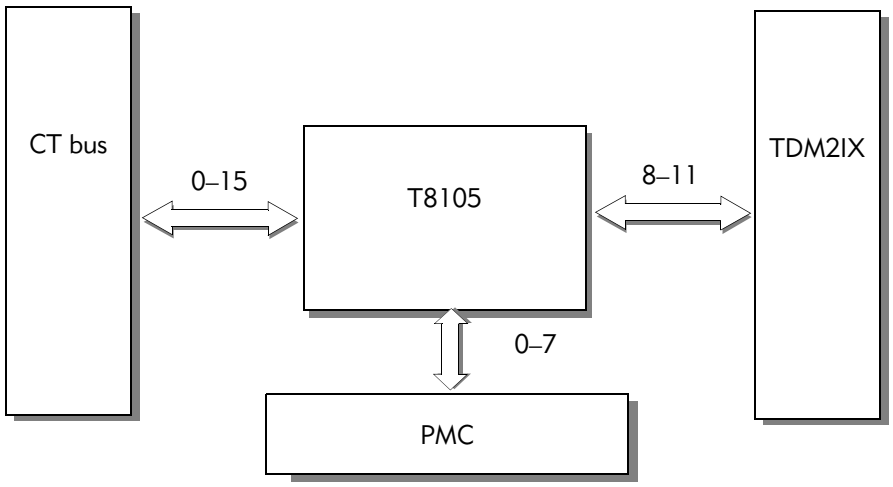
Each group can operate at any of the following stream rates, and support different number of timeslots per stream:

Stream rate	Number of time slots
2 Mbps	32
4 Mbps	64
8 Mbps	128

ENP boards and T8105

The T8105 on an ENP board interfaces with the components as shown in this figure:

Figure 8-2. T8105 interfaces



For ENP-3511 boards, the interface varies, depending upon the board configuration:

- **ENP-3511-C**: Local Streams 0 through 7 connect to the PMC and operate at 2 MHz supporting 32 timeslots per stream. Local streams 8 through 11 connect to the TDM2IX and operate at 8 MHz supporting 128 time slots per stream.
- **ENP-3511-F**: Local Streams 0 through 7 connect to the TDM2IX and operate at 8 MHz supporting 128 timeslots per stream. Local streams 8 through 15 are disabled.

Interrupt routines

Include Filetdmapi.h

Library Tdm.a

The TDM device generates a StrongARM interrupt each time an error condition occurs. These error condition can be caused by various reasons. For detailed information about error conditions, see the T8105 datasheet. ENP SDK provides an API function to install a user-specific handler to take care of these errors.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

When a TDM interrupt occurs, the TDMUserHandler is called and a structure of type InterruptStatus is passed to that function:

Table 8-1. InterruptStatus structure

Field	Value
InterruptStatus.device	INTERRUPT_FROM_TDM
InterruptStatus.status	A 32-bit word which contains the error information from the TDM device. CLKERR3|CLKERR2|CLKERR1|SYSERR forms the four bytes of InterruptStatus.status.
InterruptStatus.portStatus	Not used.

The default handler clears the interrupts and disables any further interrupts from that specific source within the device, which caused the interrupt and passes the above information to the user handler. You must re-enable any interrupt, if needed.



When output clocks are enabled using TDMSetClockConfig interrupts, event notification for those clocks are also enabled.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Sample code

Sample startup sequence

```
TDMInitSwitch(void)
TDMSetClockConfig(IN const t_TDMClockConfig *TDMClockConfig)
TDMSetSwitchConfig(IN const t_TDMSwitchConfig *TDMSwitchConfig)
```



For ENP-3511 boards, TDMInitSwitch behaves differently, depending on the board configuration:

- **ENP-3511-C**: Configures streams 0–7 at 8MHz; 8–15 are disabled.
- **ENP-3511-F**: Configures streams 0–7 at 2MHz and 8–11 at 8MHz. Although the streams can be configured at other rates, you must use 8MHz for streams 8–11.

Example: clock configuration

This example describes how the TDM switching API can configure clocks in a typical application scenario.

The system includes four ENP boards:

- **Board 1**: Serves as the primary master, which drives the H.110 A clock from the PSTN network connection.
- **Board 2**: Serves as the secondary master, which drives the H.110 B clock from the H.110 A clock, although it has a PSTN network connection which it uses as an alternate clock source when a fallback to the B clock occurs.
- **Board 3**: Drives the NETREF_1 signal from its own PSTN network connection.
- **Board 4**: Similarly drives NETREF_2.

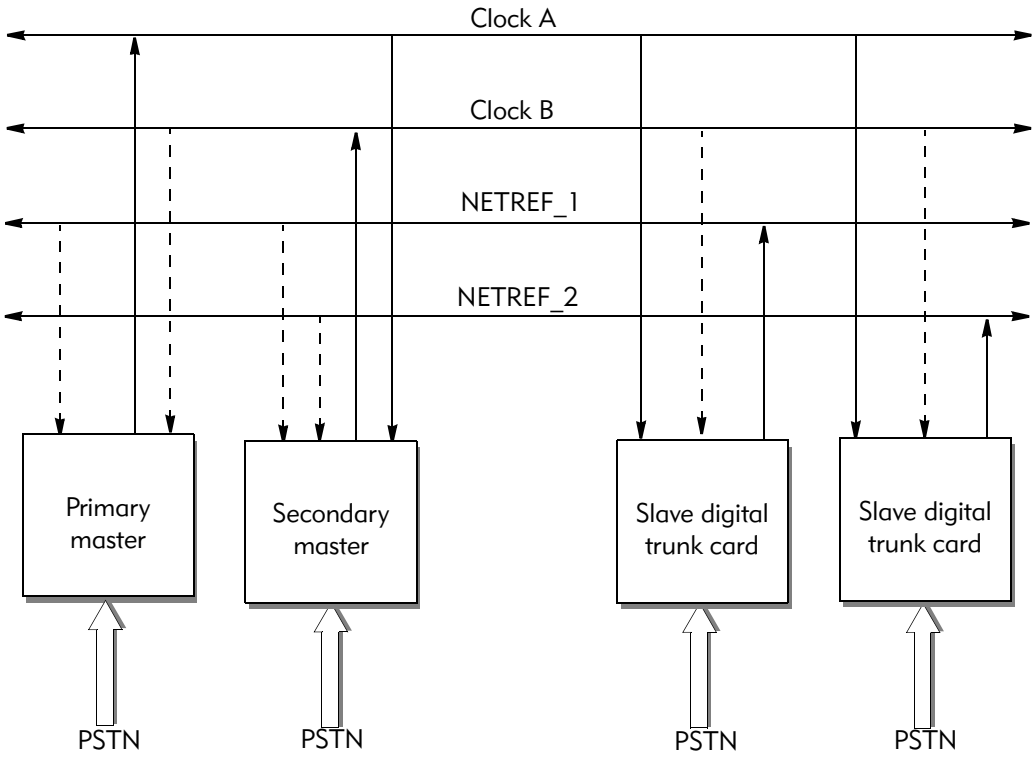
This example of clock fallback is shown in the next figure.

If the primary clock master fails, fallback to a secondary master (B clocks) can be programmed to be automatic. In this example the secondary master uses its own

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PSTN interface to drive the B clocks. It can also be programmed to use either of the NETREF signals, if so desired.

Figure 8-3. Sample clock fallback



Configuring the first board

1. Configure the board clock of the first ENP board to synchronize to the PSTN clocks and become the H.110 A clock master.
2. Declare a variable of type t_TDMClockConfig:

```
t_TDMClockConfig tdmClkConfig;
```

3. Fill the structure as follows:

```
tdmClkConfig.reference_clk_select = REF_T1E1_1
tdmClkConfig.netref_select = NETREF_CT_NETREF;
tdmClkConfig.fallback_clk_select = FB_CT_C8_B;
tdmClkConfig.netref_enable = FALSE;
tdmClkConfig.netref2_enable = FALSE;
tdmClkConfig.frame_clk_a_enable = TRUE;
tdmClkConfig.frame_clk_b_enable = FALSE;
tdmClkConfig.fallback_enable = TRUE;
```

Use TDMSetClockConfig(&tdmClkConfig); to configure the clocking as above.

Configuring the second board

1. Configures the board clock of the second ENP board to synchronize to the A clocks and become the H.110 B clock master. Its fallback source is its network connection.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2. Declare a variable of type t_TDMClockConfig:

```
t_TDMClockConfig tdmClkConfig;
```

3. Fill the structure as follows:

```
tdmClkConfig.reference_clk_select = REF_CT_C8_A
tdmClkConfig.netref_select = NETREF_CT_NETREF;
tdmClkConfig.fallback_clk_select = FB_T1E1_1;
tdmClkConfig.netref_enable = FALSE;
tdmClkConfig.netref2_enable = FALSE;
tdmClkConfig.frame_clk_a_enable = FALSE;
tdmClkConfig.frame_clk_b_enable = TRUE;
tdmClkConfig.fallback_enable = TRUE;
```

4. Use TDMSetClockConfig(&tdmClkConfig); to configure the clocking as above
5. Configure the third and fourth boards to slave off the A clocks.
6. Configure the third and fourth boards to drive NETREF_1 and NETREF_2 respectively.
7. Set the automatic fallback feature to allow for a fallback from the Primary to the Secondary Master.

Configuring the third board

1. Declare a variable of type t_TDMClockConfig:

```
t_TDMClockConfig tdmClkConfig;
```

2. Fill the structure as follows:

```
tdmClkConfig.reference_clk_select = REF_CT_C8_A
tdmClkConfig.netref_select = REF_T1E1_1;
tdmClkConfig.fallback_clk_select = FB_CT_C8_B;
tdmClkConfig.netref_enable = TRUE ;
tdmClkConfig.netref2_enable = FALSE;
tdmClkConfig.frame_clk_a_enable = FALSE;
tdmClkConfig.frame_clk_b_enable = FALSE;
tdmClkConfig.fallback_enable = TRUE;
```

3. Use t_TDMClockConfig(&tdmClkConfig); to configure the clocking as above

Configuring the fourth board

1. Declare a variable of type t_TDMClockConfig:

```
t_TDMClockConfig tdmClkConfig;
```

2. Fill the structure as follows

```
tdmClkConfig.reference_clk_select = REF_CT_C8_A
tdmClkConfig.netref_select = REF_T1E1_1;
tdmClkConfig.fallback_clk_select = FB_CT_C8_B;
tdmClkConfig.netref_enable = FALSE ;
tdmClkConfig.netref2_enable = TRUE;
tdmClkConfig.frame_clk_a_enable = FALSE;
tdmClkConfig.frame_clk_b_enable = FALSE;
tdmClkConfig.fallback_enable = TRUE;
```

3. Use t_TDMClockConfig(&tdmClkConfig); to configure the clocking as above.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example: time slot switching

Local-to-Local (T1/E1 Framer 2 to TDM2IX)

1. Switch data in timeslot 5 in T1 framer 2 to timeslot 8 in TDM2IX:

```
t_TDMSwitchConfig tdmSwitchCfg;
t_TDMConnection connectionArray [1024];

tdmSwitchCfg.number_of_connections = 1;
tdmSwitchCfg.connections = connectionArray;

connectionArray [0].connect_src.resource = RESOURCE_T1;
connectionArray [0].connect_src.mode  = CONST_DELAY;
connectionArray [0].connect_src.ctbus_connect_num= 0;
connectionArray [0].connect_src.port = 2;
connectionArray [0].connect_src.timeslot = 5;

connectionArray [0].connect_dest.resource = RESOURCE_TDM2IX;
connectionArray [0].connect_dest.mode= CONST_DELAY;
connectionArray [0].connect_dest.ctbus_connect_num = 0;
connectionArray [0].connect_dest.port = 0;
connectionArray [0].connect_dest.timeslot= 8;

setTDMSwitchConfig(&tdmSwitchCfg);
```

Local-to-CT_BUS (TDM2IX-to-CT_BUS)

1. Switch data from TDM2IX timeslot 64 to CT bus stream 6 timeslot 20:

```
t_TDMSwitchConfig tdmSwitchCfg;
t_TDMConnection connectionArray [512];

tdmSwitchCfg.number_of_connections = 1
tdmSwitchCfg.connections = connectionArray;

connectionArray [0].connect_src.resource = RESOURCE_TDM2IX;
connectionArray [0].connect_src.mode = CONST_DELAY;
connectionArray [0].connect_src.ctbus_connect_num= 1;
connectionArray [0].connect_src.port = 0;
connectionArray [0].connect_src.timeslot = 64;

connectionArray [0].connect_dest.resource = RESOURCE_CT_BUS;
connectionArray [0].connect_dest.mode = CONST_DELAY;
connectionArray [0].connect_dest.ctbus_connect_num= 1;
connectionArray [0].connect_dest.port = 6;
connectionArray [0].connect_dest.timeslot = 20;

setTDMSwitchConfig(&tdmSwitchCfg);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM-to-TDM connection

1. Switch data from CT_BUS stream 5 timeslot 10 to CT_BUS stream 8 timeslot 16:

```
t_TDMSwitchConfig tdmSwitchCfg;
t_TDMConnection connectionArray [512];

tdmSwitchCfg.number_of_connections = 1;
tdmSwitchCfg.connections = connectionArray;

connectionArray [0].connect_src.resource =RESOURCE_CT_BUS;
connectionArray [0].connect_src.mode = CONST_DELAY;
connectionArray [0].connect_src.ctbus_connect_num= 2;
connectionArray [0].connect_src.port = 5;
connectionArray [0].connect_src.timeslot = 10;

connectionArray [0].connect_dest.resource = RESOURCE_CT_BUS;
connectionArray [0].connect_dest.mode= CONST_DELAY;
connectionArray [0].connect_dest.ctbus_connect_num = 2;
connectionArray [0].connect_dest.port = 8;
connectionArray [0].connect_dest.timeslot= 16;

setTDMSwitchConfig(&tdmSwitchCfg);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function and structure list

Use these tables to identify the TDM functions and structures you want to use. Use the function or structure description later in this chapter to obtain detailed information, including syntax and parameter values.

Table 8-2. TDM functions

Function	Description	Page
TDMClearClockFault	Clears a clock fault in the TDM device.	87
t_TDMClockConfig	Used by the set_TDM_ClockConfig function to control the clocking of the TDM switching device	89
t_TDMConnection	Specifies a single connection.	91
TDMGetSwitchErrorStatus	Reads the status of the TDM switching device's error status registers.	92
TDMInitSwitch	Initializes the driver and the device.	93
TDMSetClockConfig	Configures the clocking of the TDM Switch device.	94
TDMSetInterruptHandler	Installs a user-defined interrupt handler for TDM interrupts.	95
TDMSetStreamConfig	Sets the TDM stream rates for the three on-board stream groups as well as the eight CT bus stream groups.	96
TDMSetSwitchConfig	Sets the TDM switching.	97
t_TDMStreamConfig	Used by the TDMSetStreamConfig function to control the stream rates of the TDM switching device.	98
t_TDMSwitchConfig	Specifies a series (one or more) connections to make (or break).	99

Table 8-3. TDM structures

Structure	Description	Page
t_fallback_clk	Selects the fallback reference of the TDM switching device.	100
t_netref_clk	Selects the clock reference used to generate the CT-NETREF bus clocks.	101
t_ref_clk	Selects the primary reference of the TDM switching device.	102
t_source_dest	Specifies a connection's endpoint (source or destination).	103
t_stream_rate	Selects the CT bus or local stream rate.	104

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Functions

TDMClearClockFault

Clears a clock fault in the TDM device.

Syntax

```
ENP_STATUS TDMClearClockFault(void)
```

Parameters

None.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_TDM _FAILURE

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDMClearMemoryFault

Clears a memory (CAM) fault in the TDM switch device.

Syntax

```
ENP_STATUS TDMClearMemoryFault(void)
```

Parameters

None.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_TDM _FAILURE

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_TDMClockConfig

Used by the set_TDM_ClockConfig function to control the clocking of the TDM switching device

Syntax

```
typedef struct {
   t_ref_clk reference_clk_select;
   t_netref_clk netref_select;
   t_fallback_clk fallback_clk_select;
   bool netref_enable;
   bool netref2_enable;
   bool frame_clk_a_enable;
   bool frame_clk_b_enable;
   bool fallback_enable;
} t_TDMClockConfig;
```

Elements

reference_clk_select

Selects the primary reference clock for the TDM. The TDM locks to this clock (under non-fallback conditions) and generates both its internal references as well as any external clocks, based on this clock. This clock can be any of these:

- The on-board reference.
- Any H.100/110 bus clock.
- An E1 or T1 loop timing clock.

netref_select

Selects the reference clock for generation of the CT_NETREF signal. This field selects the reference but does not enable the output.

fallback_clk_select

Selects the reference clock to which the TDM switches when an error is detected on the signal used as the *reference_clk_select*. For this to occur, fallback must be enabled.

netref_enable

Determines whether the TDM's CT_NETREF output goes on the H.100/110 bus. You can select one of these:

TRUE CT_NETREF output goes on the H.100/110 bus.

FALSE CT_NETREF output does not go on the H.100/110 bus.

netref2_enable

Determines whether the TDM's CT_NETREF_2 output goes on the H.100/110 bus. You can select one of these:

TRUE CT_NETREF_2 is available only for H.110 applications.

FALSE CT_NETREF_2 is not available for H.110 applications.

frame_clk_a_enable

Determines whether the TDM's CT_C8_A and /CT_FRAME_A outputs

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

go on the H.100/110 bus. You can select one of these:

TRUE	CT_C8_A and /CT_FRAME_A output goes on the H.100/110 bus.
FALSE	CT_C8_A and /CT_FRAME_A output does not go on the H.100/110 bus.

frame_clk_b_enable

Determines whether the TDM's CT_C8_B and /CT_FRAME_B outputs go on the H.100/110 bus. You can select one of these:

TRUE	CT_C8_B and /CT_FRAME_B output goes on the H.100/110 bus.
FALSE	CT_C8_B and /CT_FRAME_B output does not go on the H.100/110 bus.

fallback_enable

Determines whether to engage the TDM's fallback (switch) mode. You can select one of these:

1	Enables the fallback mode. The TDM falls back (switches) to the reference clock specified in *fallback_clk_select* when an error is detected on the primary reference clock specified in *reference_clk_select*. Select this option if E1 or T1 option card is installed.
2	Disables the fallback mode.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_TDMConnection

Specifies a single connection.

Syntax

```
typedef struct {
   t_source_dest connect_src;
   t_source_dest connect_dest;
} t_TDMConnection;
```

Elements

connect_src
: The connection's source.

connect_dest
: The connection's destination.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDMGetSwitchErrorStatus

Reads the status of the TDM switching device's error status registers.

Syntax

```
int TDMGetErrorStatus(void)
```

Parameters

None.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

A value that represents system and clock error registers.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDMInitSwitch

Initializes the driver and the device.

You must call this function before using any other driver functions. Following this call, the T8105(or SWITI) and the CT bus are in the following state:

- No CT bus timeslots are driven, all connections are broken.
- All CT bus bus clocks are turned off, CT bus stream rates are set to 8Mhz.
- The switching devices reference clock is set to be the on-board oscillator.
- All local/on-board connections are broken.
- The local/on-board streams are enabled.

  For ENP-3511 boards:

  - **ENP-3511-C**: Streams 0–7 are enabled at 2MHz, streams 8–11 at 8 MHz, and streams 12–15 are disabled.
  - **ENP-3511-F**: Streams 0–7 are enabled at 8MHz, streams 8–15 are disabled.
- On-board device clocks are set.

Syntax

```
ENP_STATUS TDMInitSwitch(void)
```

Parameters

None.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_TDM _FAILURE

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDMSetClockConfig

Configures the clocking of the TDM Switch device.

This includes selecting reference clocks for normal and fallback operation as well as determining which (if any) clocks to drive onto the bus. This function is typically called first following TDMInitSwitch.

Syntax

```
ENP_STATUS TDMSetClockConfig (
    const t_TDMClockConfig *TDMClockConfig
)
```

Parameters

TDMClockConfig A pointer to the t_TDMClockConfig structure.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_TDM _FAILURE

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDMSetInterruptHandler

Installs a user-defined interrupt handler for TDM interrupts.

Syntax

```
t_ENPStatus TDMSetInterruptHandler(
   void (*TDMUserHandler)(InterruptStatus)
);
```

Parameters

Pointer to the user-defined interrupt function.

Return value

ENP_SUCCESS
ENP_INT_HANDLER_INSTALL_FAILURE

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDMSetStreamConfig

Sets the TDM stream rates for the three on-board stream groups as well as the eight CT bus stream groups.

Syntax

```
ENP_STATUS TDMSetStreamConfig (
    const t_TDMStreamConfig *TDMStreamConfig
)
```

Parameters

TDMStreamConfig
A pointer to the t_TDMStreamConfig structure.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_TDM _FAILURE

Comments

Call this function prior to making timeslot connections. Changing stream rates results in reconfiguring TDM internal switch memory and may result in undesired connectivity.

This function call is optional.

The ENP-3511 must use fixed stream rates, depending upon the board configuration:

- **ENP-3511-C**: Streams 0–7 must run at 2MHz, streams 8–11 at 8 MHz, and streams 12–15 are disabled.
- **ENP-3511-F**: Streams 0–7 must run at 8MHz; streams 8–15 disabled.

TDMInitSwitch configures these when it initializes the device.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDMSetSwitchConfig

Sets the TDM switching.

The function is passed the number of connections to make (or break), and the end points for those connections. These connections can be of any type: local-to-local, local-to-CT bus or CT bus to CT bus.

Syntax

```
ENP_STATUS TDMSetSwitchConfig(
    IN const t_TDMSwitchConfig *TDMSwitchConfig
)
```

Parameters

TDMSwitchConfig
A pointer to the t_TDMSwitchConfig structure.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_TDM _FAILURE

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_TDMStreamConfig

Used by the TDMSetStreamConfig function to control the stream rates of the TDM switching device.

On-board/local stream rates are controlled in these groups:

- **TDM2IX**: TDM2IX buffered serial stream group.
- **E1/T1**: PMC option card site group.

For the H.100/110 bus, each group is a set of four consecutive H.100/110 streams.

The streams rates correspond to 32, 64 and 128 timeslots respectively.

Local streams 0–7 are configured to run at 8MHz during TDMInitSwitch; this is the only stream rate supported.

For ENP-3511 boards:

- **ENP-3511-C**: Streams 0–7 are enabled at 2MHz, streams 8–11 at 8 MHz, and streams 12–15 are disabled.
- **ENP-3511-F**: Streams 0–7 are enabled at 8MHz, streams 8–15 are disabled.

Syntax

```
typedef struct {
   t_stream_rate tdm2ix_rate;
   t_stream_rate e1t1_rate;
   t_stream_rate ct_bus_rate;
} t_TDMStreamConfig;
```

Elements

tdm2ix_rate
: Sets the stream rate for the streams connected to TDM2IX.

e1t1_rate
: Sets the stream rate for the streams connected to the PMC board's option card site. You can set the rate to one of these:

  2.048MHz — Sets the stream rate to 2.048MHz. If the RadiSys E1 or T1 option card is used, you must select this option.

  4.096MHz — Sets the stream rate to 4.096MHz.

  8.192MHz — Sets the stream rate to 8.192MHz.

  Disabled — Disables the stream.

ct_bus_yy_xx_rate
: Sets the stream rates for the H.100/110 streams xx thru yy.

Comments

This is an optional function, as the ENP-3511 works at a fixed stream rate which is initialized during TDMInitSwitch.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_TDMSwitchConfig

Specifies a series (one or more) connections to make (or break).

Syntax

```
typedef struct {
    ulong number_of_connections;
    t_TDMConnection *connections;
} t_TDMSwitchConfig;
```

Elements

number_of_connections

Indicates the total number of connections to make or break. The maximum value is:

- 1024 for a Local connection.
- 512 for a CT_BUS connection.

connections

A pointer to the number of connections specified in *number_of_connections*, each of type t_TDMConnection. Each connection includes a *connect_src* and *connect_dest*, which each include a mode, resource, port, timeslot and ctbus_connect_num (CT bus Connections only).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Structures

t_fallback_clk

Selects the fallback reference of the TDM switching device.

Syntax

```
typedef enum {
    FB_LOCAL= 0,
    FB_CT_C8_A= 1,
    FB_CT_C8_B= 2,
    FB_CT_NETREF= 3,
    FB_T1E1_1= 4,
    FB_T1E1_2= 5,
    FB_T1E1_3= 6,
    FB_CT_NETREF2= 7
} t_fallback_clk;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_netref_clk

Selects the clock reference used to generate the CT-NETREF bus clocks.

Syntax

```
typedef enum {
    NETREF_LOCAL= 0,
    NETREF_T1E1_1= 1,
    NETREF_T1E1_2= 2,
    NETREF_T1E1_3= 3,
    NETREF_T1E1_4 = 4,
    NETREF_CT_NETREF= 5,
    NETREF_CT_NETREF2= 6
} t_netref_clk;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_ref_clk

Selects the primary reference of the TDM switching device.

Syntax

```
typedef enum {
    REF_LOCAL= 0,
    REF_CT_NETREF= 1,
    REF_CT_C8_A= 2,
    REF_CT_C8_B= 3,
    REF_T1E1_1= 8,
    REF_T1E1_2= 9,
    REF_T1E1_3= 10,
    REF_T1E1_4= 11,
    REF_CT_NETREF2= 12
} t_ref_clk;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_source_dest

Specifies a connection's endpoint (source or destination).

Syntax

```
typedef struct {
   unsigned int mode;
   unsigned int resource;
   unsigned int port;
   unsigned int timeslot;
   unsigned int ctbus_connect_num
} t_source_dest;
```

Elements

mode Specifies the connection to be either constant or minimum delay.

resource Specifies the connection's resource number. For example: CT bus, T1, or TDM2IX.

port Specifies the connection's port number.

Most resources have multiple ports. For the CT bus, each stream is a port. For an E1 or T1 card, the port corresponds to the framer.

timeslot Specifies the connection's timeslot number. The maximum value for this field is a function of the stream rate selected.

ctbus_connect_num
Used only for CT bus connections. The maximum value is 512.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_stream_rate

Selects the CT bus or local stream rate.

Syntax

```
typedef enum {
    TWO_MHZ= 0,
    FOUR_MHZ= 1,
    EIGHT_MHZ= 2,
    DISABLE_GROUP= 3
} t_stream_rate;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

9 E1T1 library

This chapter describes the T1/E1 driver library.

When reading this file online, you can immediately view information about any topic by placing the mouse cursor over a topic and clicking.

For information about...	Go to this page...
Overview	105
Sample startup sequence	106
T1 Initialization example	107
Function list	109
Functions	113
Structures	132

Overview

The T1/E1 Status and Control API is your interface to the T1/E1 driver.

The T1/E1 driver provides the software interface to the T1/E1 spans of the 4-port E1/T1 option card. The driver works under the VxWorks Real-Time Operating System. The driver is not a “true” VxWorks driver in the sense that it is not built as part of the kernel. However, it makes use of VxWorks system calls and therefore requires that OS.

The driver provides control and monitoring of the two or four T1/E1 ports of the E1/T1 option card. The ports use a Infineon Quad FALC LIU/Framer as their interface device. You do not need a working knowledge of this device to use T1/E1 functions.

The driver includes:

- Interrupt handling tools
- Functions

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Sample startup sequence

```
T1E1initCard(&board_config);
```

Initializes the T1 card and driver.

```
T1E1clockConfig(&clock_config);
```

Initializes the QuadFalc receive/transmit clocks.

```
setT1Config(&config_struct);
```

Sets the configuration of the T1 line.

```
setT1Config(&config_struct);
```

Sets the configuration of the T1 line.

```
setT1Signaling(&signaling_struct);
```

Sets the initial state of the transmitted signaling bits. Not required if signaling has not been enabled.

```
setT1SignalingHandler(&userT1SignalingHandler);
```

Sets a callback function for receive signaling bit changes.

```
setT1StatusHandler(&userT1StatusHandler);
```

Sets a callback function for receive line state/status changes.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

T1 Initialization example

Here is a model code of intializing the E1T1 card in T1 operation. Two functions, E1T1_Initilize_T1 and E1T1_Initialize_E1 are provided in the E1T1 API for you to use as examples.

```
int fmr;

t_T1E1_BoardConfig board_config;
t_T1E1_ClockConfig clock_config;
t_T1_user_config_struct config_struct;
t_T1_user_signaling_data signaling_struct;

clock_config.clock_mode=SLAVE;       /* PMC in Slave mode */
clock_config.rclock_ref=LINE_1;      /* Receive clock is from Line 1 */
clock_config.rclock_rclk=DCOR;       /* Receive clock is dejittered  */
clock_config.int_rcv_sys_clk=SCLKR;  /* Internal rcv clock recovered from TDM */
clock_config.xclock_DCOX=IRF;        /* Transmit clck dejittered from internal ref clck */
clock_config.int_xmit_sys_clk=SCLKX; /* Internal TX clock recovered from TDM */

/* Initialize the driver and the framers on the PMC */
T1E1initCard(&board_config);

/* Initialize and set the respective receive and transmit clocks */
T1E1clockConfig(&clock_config);

for(fmr=0;fmr<4;++fmr)
{

if(fmr==0)
{
config_struct.framer_id=framer_1;
signaling_struct.framer_id=framer_1;
}
else if(fmr==1)
{
config_struct.framer_id=framer_2;
signaling_struct.framer_id=framer_2;
}
else if(fmr==2)
{
config_struct.framer_id=framer_3;
signaling_struct.framer_id=framer_3;
}
else if(fmr==3)
{
config_struct.framer_id=framer_4;
signaling_struct.framer_id=framer_4;
}

config_struct.line_coding=B8ZS;
config_struct.framing_mode=ESF;
config_struct.line_build_out=DSX1_0_to_133_ft;
config_struct.idle_code=0xff;
config_struct.idle_channels=0x00;
config_struct.payload_loopback_enable=FALSE;
config_struct.framer_loopback_enable=FALSE;
config_struct.local_loopback_enable=FALSE;
config_struct.remote_loopback_enable=FALSE;
config_struct.robbed_bit_signaling_enable=FALSE;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
signaling_struct.signaling_data.a_bits = 0x00000000;
signaling_struct.signaling_data.b_bits = 0x00ffffff;
signaling_struct.signaling_data.c_bits = 0x00000000;
signaling_struct.signaling_data.d_bits = 0x00ffffff;
signaling_struct.signaling_data.changed_bits = 0x00ffffff;

/* Set Card in T1 Mode and set cnfiguration of a specified port */
setT1Config(&config_struct);

/* Sets the values of the transmitted robbed bit signaling bits */
setT1Signaling(&signaling_struct);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function list

This chapter lists function descriptions alphabetically. You can find a function either by locating it alphabetically within its group, or by using Table 9-1 below, which groups like functions together.

Use this table to identify the T1/E1 driver functions you want to use. Use the function description later in this appendix to obtain detailed information, including syntax and parameter values:

Table 9-1. T1/E1 functions

Type	Function	Description
Configuration		Configure each port for T1 or E1 operation and configure the appropriate line parameters (line coding, framing mode, buildout, etc.). In addition, these functions allow for controlling of outbound robbed bit signaling, alarm generation, idle code insertion and loop-up/loop-down codes.
	T1E1initCard	Initializes the driver and initialize the framers on the option card.
	T1E1clockConfig	Initializes and sets the respective receive and transmit clocks in the Quad Falc framer chip on the option card.
	T1E1getBoardConfig	Gets information about the installed option card without performing the initialization done by T1E1initCard.
	T1E1setLeds	Controls the LEDs associated with each port of the card.
	setT1Config	Sets the card in T1 mode and sets the configuration of a specified port.
	setT1Signaling	Sets the values of the transmitted robbed bit signaling bits.
	setT1Command	Transmits “commands” over the specified T1 port.
	setT1ClearChannels	Specifies which timeslots (DS0s) on the specified T1 port are clear channel (64Kbps) and which use robbed-bit signaling, and are therefore 56Kbps channels.
	setT1IdleChannels	Specifies which timeslots (DS0s) on the specified T1 port transmit the idle code and which transmit data “normally”.
	setT1ChannelConfig	Configures individual timeslots (DS0s) on the specified T1 port.
	setE1Config	Sets the card in E1 mode and set the configuration of a specified port.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 9-1. T1/E1 functions

Type	Function	Description
	setE1Signaling	Sets the values of the transmitted CAS (channel associated signaling) bits.
Status monitoring		Polls the status of the framer and of each T1/E1 span. This status includes alarm conditions, loop code receipt and robbed bit signaling. These functions can also be used to collect status following the receipt of an interrupt event.
	getT1Signaling	Reads the values of the received robbed-bit signaling bits on the specified T1 port.
	getT1Status	Reads the current receive and transmit status of a specified T1 port.
	getT1SignalingRaw	Reads the values of the received robbed-bit signaling bits on the specified T1 port.
	getE1Signaling	Reads the values of the received CAS (channel associated signaling) bits on the specified E1 port.
Interrupt/event controls		Turns on and off alarm and signaling interrupt generation and allow the user to specify handlers (callback functions) to be called when the event occurs.
	setT11SignalingHandler	Specifies the event handler to call after detecting a change in receive T1 signaling state.
	setT1StatusHandler	Specifies the event handler to call after detecting a change in receive line state.
	setE1SignalingHandler	Specifies the event handler to call after detecting a change in receive E1 signaling state.

Function prototypes and type definitions for the driver interface are contained in the E1T1api.h header file.

The functions and their data types are described in the pages that follow. The descriptions use the terms port, framer, span, line and DS1 interchangeably to indicate 1 of 4 T1/E1 connections available on the I/O board.

Functions which contain "T1E1" in their name are used for either T1 or E1 operation. Currently the driver does not support mixed operation on a single option card. Functions with only "T1" in their name are T1 specific while functions with only "E1" are E1 specific. Basic selection between T1 and E1 operating modes is made with the setT1Config or setE1Config API calls. These mutually exclusive functions will configure the card for either T1 or E1 operation and must be called prior to calling other T1 or E1 specific functions.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Limitations

- **E1 support:** The API currently provides limited support for E1. Support is provided for span configuration and signaling generation and detection only. T1 support is much more comprehensive, however the T1 ESF Facility Data Link functionality is not supported.
- **Time slot assignments for T1 and E1 interfaces:** There are differences in the mapping of T8100 time slots depending on the interface used, as illustrated by the following table. The developer should be aware of these differences when assigning DSP data channel resources to time slots using the T8100 peripheral.

Table 9-2. Time slot numbers

T8100	T1	E1
0	1 (data 0)	0 (signaling 0)
1	2 (data 1)	1 (data 0)
2	3 (data 2)	2 (data 1)
3	4 (data 3)	3 (data 2)
4	5 (data 4)	4 (data 3)
5	6 (data 5)	5 (data 4)
6	7 (data 6)	7 (data 6)
7	8 (data 7)	7 (data 6)
8	9 (data 8)	8 (data 7)
9	10 (data 9)	9 (data 8)
10	11 (data 10)	10 (data 9)
11	12 (data 11)	11 (data 10)
12	13 (data 12)	12 (data 11)
13	14 (data 13)	13 (data 12)
14	15 (data 14)	14 (data 13)
15	16 (data 15)	15 (data 14)
16	17 (data 16)	16 (signaling 1)
17	18 (data 17)	17 (data 15)
18	19 (data 18)	18 (data 16)
19	20 (data 19)	19 (data 17)
20	21 (data 20)	20 (data 18)
21	22 (data 21)	21 (data 19)
22	23 (data 22)	22 (data 20)
23	24 (data 23)	23 (data 21)
24	N/A	24 (data 22)
25	N/A	25 (data 23)
26	N/A	26 (data 24)
27	N/A	27 (data 25)
28	N/A	28 (data 26)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 9-2. Time slot numbers

T8100	T1	E1
29	N/A	29 (data 27)
30	N/A	30 (data 28)
31	N/A	31 (data 29)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Functions

T1E1initCard

Initializes the driver and initialize the framers on the option card.

The function accepts a pointer to a structure and fills the structure with information about the installed card. You *must* call this function before using any other driver functions. As the name implies, this function is *not* specific to T1 or E1 operations.

Syntax

```
void T1E1initCard(
    t_T1E1_BoardConfig *board_config
)
```

Parameters

board_config

A pointer to the t_T1E1_BoardConfig structure. For more information about this structure, see *t_T1E1_BoardConfig* on page 138.

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

T1E1clockConfig

Initializes and sets the respective receive and transmit clocks in the Quad Falc framer chip on the option card.

The function accepts a pointer to a structure that holds the clock configuration information. This function must be called immediately after T1E1initCard and before using any other driver functions.

Syntax

```
void T1E1clockConfig (
   t_T1E1_ClockConfig  clock_config
)
```

Parameters

clock_config
: A pointer to the t_T1E1ClockConfig structure. For more information about this structure, see *t_T1E1_ClockConfig* on page 145.

Return values

SUCCESS Successful completion

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

T1E1getBoardConfig

Gets information about the installed option card without performing the initialization done by T1E1initCard.

Syntax

```
void T1E1getBoardConfig(
   t_T1E1_BoardConfig *board_config
)
```

Parameters

board_config

A pointer to the t_T1E1_BoardConfig structure. For more information about this structure, see *t_T1E1_BoardConfig* on page 138.

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

T1E1setLeds

Controls the LEDs associated with each port of the card.

See the Type Definition section for a complete description of possible LED states.

Syntax

```
void T1E1setLeds(
   t_T1E1_framer_id              framer_id,
   t_T1E1_led_state              led_state
)
```

Parameters

framer_id A pointer to the t_T1E1_framer_id structure. For more information about this structure, see *t_T1E1_framer_id* on page 134.

led_state A pointer to the structure.

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

setT1Config

Sets the card in T1 mode and sets the configuration of a specified port.

You *must* call this function after calling T1E1initCard (for T1 operation).

Syntax

```
void setT1Config(
   t_T1_user_config_struct *config_struct
)
```

Parameters

config_struct

A pointer to the t_T1_user_config_struct structure. For more information about this structure, see *t_T1_user_config_struct* on page 149.

Configuration includes:

- Line coding
- Framing mode
- Line buildout
- Idle channel and loopback specifications
- Enabling/disabling robbed bit signaling

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

setT1Signaling

Sets the values of the transmitted robbed bit signaling bits.

To transmit the bits, robbed bit signaling must be enabled via setT1Config. The *changed_bits* field of the signaling_struct determines which channel's (DS0's) bits are updated.

A 1 in Bit 0 of *changed_bits* corresponds to the first timeslot of the DS1 line. Bit 23 of *changed_bits* corresponds to the last timeslot of the DS1 line.

For details about each field, see the t_T1_user_signaling_data structure description.

Syntax

```
void setT1Signaling(
    t_T1_user_signaling_data *signaling_struct
)
```

Parameters

signaling_struct

A pointer to the t_E1_user_signaling_data structure. For more information about this structure, see *t_T1_user_signaling_data, t_T1_signaling_data* on page 151.

Outputs

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

setT1Command

Transmits “commands” over the specified T1 port.

Syntax

```
void setT1Command(
    t_T1_user_command_data *command_struct
)
```

Parameters

command_struct

A pointer to the t_T1_user_command_data structure. For more information about this structure, see *t_T1_user_command_data* on page 152

You can select one of these:

Transmit AIS

Transmit yellow (remote) alarm

Transmit loop up code

Transmit loop down code

Outputs

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

setT1ClearChannels

Specifies which timeslots (DS0s) on the specified T1 port are clear channel (64Kbps) and which use robbed-bit signaling, and are therefore 56Kbps channels.

By default, channels are assigned to be non-clear channel (56Kbps). For any channels to be used for robbed bit signaling, the *robbed_bit_signaling_enable* field in the t_T1_user_config_struct configuration command sent via setT1Config must be set to TRUE. If it is set to FALSE, all channels are clear, regardless of the configuration specified through this function.

Syntax

```
void setT1ClearChannels(
    t_T1_user_clear_channel_data clear_channels_struct
)
```

Parameters

clear_channels_struct

A pointer to the t_T1_user_clear_channel_data structure. For more information about this structure, see *t_T1_user_clear_channel_data* on page 154.

Outputs

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

setT1IdleChannels

Specifies which timeslots (DS0s) on the specified T1 port transmit the idle code and which transmit data “normally”.

By default, channels are assigned as non-idle. The function also specifies the 8-bit idle code.



Idle code data overwrites robbed bit signaling bits.

Syntax

```
void setT1IdleChannels(
    t_T1_user_idle_struct *idle_channels_struct
)
```

Parameters

idle_channels_struct

A pointer to the t_T1_user_idle_struct structure. For more information about this structure, see *t_T1_user_idle_struct* on page 155.

Outputs

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

setT1ChannelConfig

Configures individual timeslots (DS0s) on the specified T1 port.

You can specify these parameters:

- Insertion/removal of idle code
- Clear channel mode
- ABCD robbed bits (supervision)

Syntax

```
void setT1ChannelConfig(
   t_T1_user_channel_config *channel_config
)
```

Parameters

channel_config

A pointer to the t_T1_user_channel_config structure. For more information about this structure, see *t_T1_user_channel_config* on page 156.

Outputs

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

setE1Config

Sets the card in E1 mode and set the configuration of a specified port.

Configuration includes line coding, framing mode, line buildout, idle channel and loopback specifications and enabling/disabling channel associated signaling. This function *must* be called after calling T1E1initCard (for E1 operation).

Syntax

```
void setE1Config(
   t_E1_user_config_struct *config_struct
)
```

Parameters

config_struct

A pointer to the t_E1_user_config_struct structure. For more information about this structure, see *t_E1_user_config_struct* on page 164.

Outputs

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

setE1Signaling

Sets the values of the transmitted CAS (channel associated signaling) bits.

To transmit the bits, CAS must be enabled via setE1Config. The *changed_bits* field in signaling_struct determines which channel's (DS0's) bits to update. A 1 in Bit 0 of *changed_bits* corresponds to the first timeslot of the E1 line. Bit 31 of *changed_bits* corresponds to the last timeslot of the E1 line.

See t_E1_user_signaling_data structure description for details on each of the fields.

Syntax

```
void setE1Signaling(
   t_E1_user_signaling_data *signaling_struct
)
```

Parameters

signaling_struct
A pointer to the t_E1_user_signaling_data structure. For more information about this structure, see *t_E1_user_signaling_data* on page 166.

Outputs

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

getT1Signaling

Reads the values of the received robbed-bit signaling bits on the specified T1 port.

To receive bits, robbed bit signaling must be enabled via setT1Config. When signaling is read in this fashion, the *changed_bits* and *timestamp* fields of the signaling_struct are invalid and not updated.

Syntax

```
void getT1Signaling(
   t_T1_user_signaling_data *signaling_struct
)
```

Parameters

signaling_struct

A pointer to the t_E1_user_signaling_data structure. For more information about this structure, see *t_T1_user_signaling_data, t_T1_signaling_data* on page 151.

Outputs

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

getT1Status

Reads the current receive and transmit status of a specified T1 port.

Status consists of the currently transmitted "commands" and the current receive conditions (alarms and errors).

Syntax

```
void getT1Status(
    t_T1_user_status_data *status_struct
)
```

Parameters

status_struct

A pointer to the t_T1_user_status_struct structure. For more information about this structure, see *t_T1_user_status_struct* on page 153.

Outputs

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

getT1SignalingRaw

Reads the values of the received robbed-bit signaling bits on the specified T1 port.

This function differs from the getT1Signaling function in the format in which data is presented to the user. This function is specifically designed for users who want to poll signaling bits rather than use an interrupt based method.

Syntax

```
void getT1SignalingRaw(
   t_T1_user_raw_signaling_struct *signaling_struct
)
```

Parameters

signaling_struct

A pointer to the t_T1_user_raw_signaling_struct structure. For more information about this structure, see *t_T1_user_raw_signaling_struct* on page 158.

Outputs

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

getE1Signaling

Reads the values of the received CAS (channel associated signaling) bits on the specified E1 port.

To receive the bits, CAS must be enabled via setE1Config. When signaling is read in this fashion, the *changed_bits* and *timestamp* fields of *signaling_struct* are invalid and not updated.

Syntax

```
void getE1Signaling(
    t_E1_user_signaling_data *signaling_struct
)
```

Parameters

signaling_struct

A pointer to the t_E1_user_signaling_data structure. For more information about this structure, see *t_E1_user_signaling_data* on page 166.

Outputs

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

setT11SignalingHandler

Specifies the event handler to call after detecting a change in receive T1 signaling state.

This function “hooks” in the user’s event handler and enables signaling event generation.

Syntax

```
void setT1SignalingHandler(
   T1SignalingHandler userT1SignalingHandler
)
```

Parameters

userT1SignalingHandler

You can select one of these:

Event handler name
: A user-specified signaling event handler. For more information about this event handler, see *T1SignalingHandler* on page 160.

NULL
: Disables status event generation.

Outputs

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

setT1StatusHandler

Specifies the event handler to call after detecting a change in receive line state.

This function "hooks" in the user's event handler and enables status event generation.

Syntax

```
void setT1StatusHandler(
    T1StatusHandler userT1StatusHandler
)
```

Parameters

userT1StatusHandler

A pointer to the structure. For more information about this structure, see *t_T1E1_BoardConfig* on page 138.

You can select one of these:

Event handler name

Passes a pointer to t_T1_user_status_data that indicates which framer had a state change or changes, which state bits changed value, the value of each state bit, and a timestamp that indicates when the change took place.

For more information about this event handler, see *T1StatusHandler* on page 159.

NULL Disables status event generation.

Outputs

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Comments

Detected receive-line state changes include:

- AIS
- Yellow alarm
- LOF (Loss of frame sync)
- LOS (Loss of signal)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

setE1SignalingHandler

Specifies the event handler to call after detecting a change in receive E1 signaling state.

This function "hooks" the user's event handler and enables signaling event.

Syntax

```
void setE1SignalingHandler(
    E1SignalingHandler userE1SignalingHandler
)
```

Parameters

userE1SignalingHandler

You can select one of these:

Event handler name
: A user-specified signaling event handler. For more information about this event handler, see *E1SignalingHandler* on page 167.

NULL
: Disables status event generation.

Outputs

Return values

SUCCESS Successful completion.

FAILURE Unsuccessful completion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Structures

Use this table to identify the E1/T1 driver structures. Use the structure description later in this appendix to obtain detailed information, including syntax and parameter values.

Table 9-3. E1/T1 structures

Type	Call	Description
Common (E1 and T1) definitions		
	t_T1E1_framer_id	Specifies a framer (port) for functions which can operate on any port.
	t_T1E1_card_type	Identifies the type of card installed.
	t_T1E1_led_state	Controls the LEDs for each port/span.
	t_T1E1_user_signaling_data	
	t_T1E1_BoardConfig	Indicates the installed card type.
	t_T1E1_clock_mode	Sets option card (PMC) in Master or Slave mode.
	t_T1E1_rclock_ref	Selects the line/port to extract receive clock in Slave mode.
	t_T1E1_rclock_rclk	Selects type of receive clock; dejittered (suggested) or un-dejittered.
	t_T1E1_int_rcv_sys_clk	Selects source for internal receive system interface working clock.
	t_T1E1_xclock_DCOX	Selects transmit clock source.
	t_T1E1_int_xmit_sys_clk	Selects source for internal transmit system interface working clock.
	t_T1E1_ClockConfig	Specifies the Receive/Transmit clocks and various other clock configuration options available in the QuadFalc on the option card.
T1 definitions		
	t_T1_line_coding	Specifies the line coding for a framer (port).
	t_T1_framing_mode	Specifies the line framing for a framer (port).
	t_T1_line_buildout	Specifies the line buildout for a framer (port).
	t_T1_user_config_struct	Specifies the complete configuration for a framer (port).
	t_T1_user_signaling_data, t_T1_signaling_data	Reads and writes signaling data.
	t_T1_user_command_data	Sends alarm and loop up or down codes over the T1 line.
	t_T1_user_status_struct	Provides line status.
	t_T1_user_clear_channel_data	Indicates which channels are clear channels (64Kbps) and which are not (56Kbps).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 9-3. E1/T1 structures

Type	Call	Description
	t_T1_user_idle_struct	Transmits an idle code on a selected DS0.
	t_T1_user_channel_config	Controls the configuration of a specified DS0 on a specified framer/port.
	t_T1_user_raw_signaling_struct	Reads the signaling bit (supervision bit) values for each channel on the specified T1 port.
E1 definitions		
	t_E1_line_coding	Enumerate the line coding for a framer (port).
	t_E1_signaling_mode	Enumerates the signaling mode for a framer (port).
	t_E1_line_buildout	Enumerates the line buildout for a framer (port).
	t_E1_user_config_struct	Specifies the complete configuration for a framer (port).
	t_E1_user_signaling_data	Reads and writes signaling data.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1E1_framer_id

Specifies a framer (port) for functions which can operate on any port.

Syntax

```
typedef enum {
    framer_1 = 0,
    framer_2 = 1,
    framer_3 = 2,
    framer_4 = 3
} t_T1E1_framer_id;
```

Elements

framer_1 The first port on the framer.

framer_2 The second port on the framer.

framer_3 The third port on the framer.

framer_4 The fourth port on the framer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1E1_card_type

Identifies the type of card installed.

Syntax

```
typedef enum {
    E1,
    T1
} t_T1E1_card_type;
```

Elements

E1	The E1 card.
T1	The T1 card.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1E1_led_state

Controls the LEDs for each port/span.

Each port has two associated LEDs. Each LED is bi-colored, with one LED either yellow or green, and the other either red or green.

Syntax

```
typedef enum {
    leds_off,
    leds_normal,
    leds_yellow_alarm,
    leds_red_alarm,
    leds_loopback,
    leds_line_fault,
    leds_gr_yellow,
    leds_red_gr,
    leds_off_gr,
} t_T1E1_led_state;
```

Elements

leds_off Sets both LEDs to off.

leds_normal
Sets one LED to off and sets the other to green.

leds_yellow_alarm
Sets one LED to off and sets the other to yellow.

leds_red_alarm
Sets one LED to off and sets the other to red.

leds_loopback
Sets both LEDs to green.

leds_line_fault
Sets one LED to red and sets the other to yellow.

leds_gr_yellow
Sets one LED to green and sets the other to yellow.

leds_red_gr
Sets one LED to red and sets the other to green.

leds_off_gr
Sets one LED to off and sets the other to green.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1E1_user_signaling_data

A type definition for a function pointer that points to a user handler that will receive a t_T1E1_user_signaling_data structure pointer as a parameter.

Syntax

```
typedef void
   (*T1E1SignalingHandler)
   ( t_T1E1_user_signaling_data *signaling_struct);
```

Elements

T1E1SignalingHandler
: A pointer to the T1/E1 signaling handler.

signaling_struct
: A pointer to a t_T1E1_user_signaling_data structure.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1E1_BoardConfig

Indicates the installed card type.

Calls to T1E1initCard and T1E1getBoardConfig fill in this structure.

Syntax

```
typedef struct {
   bool                    init_passed;
   t_T1E1_card_type        card_type;
   uchar                   number_of_ports;
} t_T1E1_BoardConfig;
```

Elements

init_passed

Determines whether to send AIS. You can select one of these:

TRUE Initialization was successful.

FALSE The board was not intialized.

card_type

Identifies the type of card installed. For more information, see *t_T1E1_card_type* on page 135.

number_of_ports

The number of T1 or E1 ports on the board.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1E1_clock_mode

Sets option card (PMC) in Master or Slave mode.

In Master mode PMC generates its own clock while in Slave mode it extracts clock from one of the receive lines.

Syntax

```
typedef    enum{
    MASTER = 1,
    SLAVE = 2
} t_T1E1_clock_mode;
```

Elements

MASTER PMC set in Master mode.

SLAVE PMC set in slave mode.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1E1_rclock_ref

Selects the line/port to extract receive clock in Slave mode.

Syntax

```
typedef enum{
    LINE_1 = 0,
    LINE_2 = 1,
    LINE_3 = 2,
    LINE_4 = 3
} t_T1E1_rclock_ref;
```

Elements

LINE_1 The first port on the framer. Use LINE_1 as default value in MASTER mode.

LINE_2 The second port on the framer.

LINE_3 The third port on the framer.

LINE_4 The fourth port on the framer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1E1_rclock_rclk

Selects type of receive clock; dejittered (suggested) or un-dejittered.

Syntax

```
typedef enum{
    DPLL = 0,
    DCOR = 1
} t_T1E1_rclock_rclk;
```

Elements

DPLL	Clock recovered from line directly drives the receive clock.
DCOR	Clock recovered is dejittered to drive the receive clock. This is the suggested value.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1E1_int_rcv_sys_clk

Selects source for internal receive system interface working clock.

This can be extracted from the baseboard TDM switch (suggested) or can be directly accessed from the extracted receive clock from one of the ports.

Syntax

```
typedef enum{
    SCLKR = 0,
    DCR = 1
} t_T1E1_int_rcv_sys_clk;
```

Elements

SCLKR — Internal receive clock extracted from baseboard TDM Switch. This is suggested for avoiding possible sync problems with TDM clock.

DCR — Internal receive clock extracted from one of the receive ports.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1E1_xclock_DCOX

Selects transmit clock source.

This can either be dejittered from one of the many sources selected earlier for the receive clocks (strongly suggested) or can be sourced from an external reference clock.

Syntax

```
typedef enum{
    IRF = 0,
    TCLK = 1
} t_T1E1_xclock_DCOX;
```

Elements

IRF	Transmit clock dejittered from one of the internal reference clocks, selected earlier for the receive clock. Suggested value.
TCLK	Transmit clock sourced from external reference clock.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1E1_int_xmit_sys_clk

Selects source for internal transmit system interface working clock.

Can either be sourced from the baseboard TDM Switch supplied clock (suggested) or can be directly routed from the receive clock extracted from one of the framer ports.

Syntax

```
typedef enum{
    SCLKX = 0,
    RCLK = 1
} t_T1E1_int_xmit_sys_clk;
```

Elements

SCLKX — Internal transmit clock recovered from baseboard TDM Switch supplied clock.

RCLK — The receive clock from one of the framer ports is routed to provide the internal transmit clock.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1E1_ClockConfig

Specifies the Receive/Transmit clocks and various other clock configuration options available in the QuadFalc on the option card.

Call to T1E1clockConfig uses this structure.

Syntax

```
typedef struct  {
    t_T1E1_clock_mode          clock_mode;
    t_T1E1_rclock_ref          rclock_ref;
    t_T1E1_rclock_rclk         rclock_rclk;
    t_T1E1_int_rcv_sys_clk     int_rcv_sys_clk;
    t_T1E1_xclock_DCOX         xclock_DCOX;
    t_T1E1_int_xmit_sys_clk    int_xmit_sys_clk;
} t_T1E1_ClockConfig;
```

Elements

clock_mode
: Selects MASTER or SLAVE mode for option card. For more information, see *t_T1E1_clock_mode* on page 139.

rclock_ref
: Selects the reference line/port for the receive clock. For more information, see *t_T1E1_rclock_ref* on page 140.

rclock_rclk
: Selects type of receive clock; dejittered(suggested) or un-dejittered. For more information, see *t_T1E1_rclock_rclk* on page 141.

int_rcv_sys_clk
: Selects source for internal receive system interface working clock. For more information, see *t_T1E1_int_rcv_sys_clk* on page 142.

xclock_DCOX
: Selects transmit clock source. For more information, see *t_T1E1_xclock_DCOX* on page 143.

int_xmit_sys_clk
: Selects source for internal transmit system interface working clock. For more information, see *t_T1E1_int_xmit_sys_clk* on page 144.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1_line_coding

Specifies the line coding for a framer (port).

Syntax

```
typedef enum {
    AMI,
    B8ZS
} t_T1_line_coding;
```

Elements

AMI	AMI (alternate mark inversion) line coding.
B8ZS	B8ZS (binary 8 zeros substitution) line coding.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1_framing_mode

Specifies the line framing for a framer (port).

Syntax

```
typedef enum {
    D4_SF,
    ESF
} t_T1_framing_mode;
```

Elements

D4_SF	D4 super frame.
ESF	Extended super frame.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1_line_buildout

Specifies the line buildout for a framer (port).

Syntax

```
typedef enum {
    DSX1_0_to_133_ft,
    DSX1_133_to_266_ft,
    DSX1_266_to_399_ft,
    DSX1_399_to_533_ft,
    DSX1_533_to_655_ft,
    CSU_minus7p5_db,
    CSU_minus15_db,
    CSU_minus22p5_db
} t_T1_line_build_out;
```

Elements

DSX1	Sets line buildout to short-haul mode.
CSU	Sets line buildout to long-haul mode.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1_user_config_struct

Specifies the complete configuration for a framer (port).

Syntax

```
typedef struct {
    t_T1E1_framer_id        framer_id;
    t_T1_line_coding        line_coding;
    t_T1_framing_mode       framing_mode;
    t_T1_line_build_out     line_build_out;
    uchar                   idle_code;
    ulong                   idle_channels;
    BOOL32                  payload_loopback_enable;
    BOOL32                  framer_loopback_enable;
    BOOL32                  local_loopback_enable;
    BOOL32                  remote_loopback_enable;
    BOOL32                  robbed_bit_signaling_enable;
} t_T1_user_config_struct;
```

Elements

framer_id Specifies a framer (port) for functions which can operate on any port. For more information, see *t_T1E1_framer_id* on page 134.

line_coding
Specifies the line coding for a framer (port). For more information, see *t_T1_line_coding* on page 146.

framing_mode
Specifies the line framing for a framer (port). For more information, see *t_T1_framing_mode* on page 147.

line_build_out
Specifies the line buildout for a framer (port). For more information, see *t_T1_line_buildout* on page 148.

idle_code Specifies the pattern to inject into transmitted data.

idle_channels
The lower 24 bits specify the channels to inject the pattern specified in *idle_code* into.

A 1 in any bit indicates that the pattern is to be injected. Bit 0 is associated with the first DS0 of the DS1 line while bit 23 is associated with the last DS0.

payload_loopback_enable
Instructs the framer to perform a line-side loopback. Setting any field to TRUE enables the loopback.

framer_loopback_enable
Instructs the framer to perform a system-side loopback. Setting any field to TRUE enables the loopback.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

local_loopback_enable
: Instructs the framer to perform a system-side loopback. Setting any field to TRUE enables the loopback.

remote_loopback_enable
: Instructs the framer to perform a line-side loopback. Setting any field to TRUE enables the loopback.

robbed_bit_signaling_enable
: Turns on and off the robbed bit signaling. You can select one of these:

  Disabled — The T1 line operates in clear channel mode.

  Enabled — Robbed bit signaling bits are inserted into and extracted from the T1 data stream.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1_user_signaling_data, t_T1_signaling_data

Reads and writes signaling data.

The first five fields of t_T1_signaling_data are bit masks where the 24 low-order bits represent the 24 DS0s of a DS1 line. These bits either provide status of (setT1Signaling or user callback) or allow control over the signaling bits (setT1Signaling). Bit 0 is used for timeslot 1, Bit 1 for timeslot 2 and so on to bit 23 which is used for timeslot 24.

Syntax

```
typedef struct {
   t_T1E1_framer_id        framer_id;
   t_T1_signaling_data     signaling_data;
} t_T1_user_signaling_data;

typedef struct {
   ulong                   changed_bits;
   ulong                   a_bits;
   ulong                   b_bits;
   ulong                   c_bits;
   ulong                   d_bits;
   ulong                   timestamp;
} t_T1_signaling_data;
```

Elements

framer_id Specifies a framer (port) for functions which can operate on any port. For more information, see *t_T1E1_framer_id* on page 134.

signaling_data
Reads and writes signaling data. For more information, see *t_T1_user_signaling_data, t_T1_signaling_data* on page 151.

changed_bits
Indicates which bits have changed or need to change state. When setting signaling bits, this mask indicates which bits should be updated. When reading signaling bits, this mask indicates which bits have changed state since the previous change. Any bit set to 1 indicates that a change has taken (or needs to take) place.

a_bits Represents the 24 A signaling bits of the DS1 line.

b_bits Represents the 24 B signaling bits of the DS1 line.

c_bits Represents the 24 C signaling bits of the DS1 line. If the line is configured for D4/SF framing, these bits are not used.

d_bits Represents the 24 D signaling bits of the DS1 line. If the line is configured for D4/SF framing, these bits are not used.

timestamp A 1µS granularity (LSB=1 µS) timestamp that indicates when the signaling change took place. This field is useful when forwarding signaling information to other systems which may be interpreting signaling bit changes. The rollover value for this field is 0x07C1F080. This field is not used when setting signaling bits.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1_user_command_data

Sends alarm and loop up or down codes over the T1 line.

Setting any field to TRUE sends the alarm or code. These fields are mutually exclusive.

Syntax

```
typedef struct {
    t_T1E1_framer_id        framer_id;
    BOOL32                  send_AIS;
    BOOL32                  send_yellow_alarm;
    BOOL32                  send_loop_up;
    BOOL32                  send_loop_down;
} t_T1_user_command_data;
```

Elements

framer_id Specifies a framer (port) for functions which can operate on any port. For more information, see *t_T1E1_framer_id* on page 134.

send_AIS Determines whether to send AIS. You can select one of these:

TRUE Sends an AIS (unframed all 1s signal) alarm.

FALSE Does not send and AIS alarm.

send_yellow_alarm

Determines whether to send yellow alarms. You can select one of these:

TRUE Sends yellow alarms.

FALSE Does not send yellow alarms.

send_loop_up

Determines whether to send loop up. You can select one of these:

TRUE Sends loop up.

FALSE Does not send loop up.

send_loop_down

Determines whether to send loop down. You can select one of these:

TRUE Sends loop down.

FALSE Does not send loop down.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1_user_status_struct

Provides line status.

Status is provided via one of these methods:

- Call getT1Status: The *framer_id* is specified by the caller and the *timestamp* and *change_word* fields are invalid.
- Use a callback function specified through setT1StatusHandler: A change in receive line status results in a callback to the user specified handler. In this case all four fields of t_T1_user_status_struct are valid.

Syntax

```
typedef struct {
   t_T1E1_framer_id        framer_id;
   ulong                   state_word;
   ulong                   change_word;
   ulong                   timestamp;
} t_T1_user_status_data;
```

Elements

framer_id Identifies the framer that changed state.

state_word
A bit mask which indicates the current transmit commands and receive line status. It indicates the current line state (same as in the "get" case). The bit definitions for this field are contained in iop_to_t1.h.

change_word
A bit mask which indicates which bit (or bits) in *state_word* caused the event to occur.

timestamp A 1 µS granularity time value indicating when the state change took place.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1_user_clear_channel_data

Indicates which channels are clear channels (64Kbps) and which are not (56Kbps).

Syntax

```
typedef struct {
   t_T1E1_framer_id        framer_id;
   ulong                   clear_channels;
} t_T1_user_clear_channel_data;
```

Elements

framer_id Specifies the framer or port to configure.

clear_channels

A bit mask with the 24 lower bits representing the 24 DS0s.

1 (in any bit) configures that DS0 for clear channel operation.

0 (in any bit) configure that DS0 for non-clear channel.

Bit 0 is associated with the first DS0 of the DS1 line while bit 23 is associated with the last DS0.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1_user_idle_struct

Transmits an idle code on a selected DS0.

Syntax

```
typedef struct {
    t_T1E1_framer_id        framer_id;
    uchar                   idle_code;
    ulong                   idle_channels;
} t_T1_user_idle_struct;
```

Elements

framer_id Specifies the framer or port to configure.

idle_code An 8-bit value which represents the idle code to transmit.

idle_channels

A bit mask where the 24 lower bits represent the 24 DS0s.

1 (in any bit) configures that DS0 to transmit the *idle_code*.

0 (in any bit) configures that DS0 to transmit data received from the on-board TDM switch.

Bit 0 is associated with the first DS0 of the DS1 line while bit 23 is associated with the last DS0.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1_user_channel_config

Controls the configuration of a specified DS0 on a specified framer/port.

Syntax

```
typedef struct {
    t_T1E1_framer_id        framer_id;
    ulong                   channel;
    BOOL32                  enable_idle_code;
    BOOL32                  disable_idle_code;
    BOOL32                  enable_clear_channel;
    BOOL32                  disable_clear_channel;
    BOOL32                  update_supervision;
    BOOL32                  a_signaling_bit;
    BOOL32                  b_signaling_bit;
    BOOL32                  c_signaling_bit;
    BOOL32                  d_signaling_bit;
} t_T1_user_channel_config;
```

Elements

framer_id Specifies the framer/port to configure.

channel Specifies the DS0 to configure (zero based).

enable_idle_code

You can select one of these:

TRUE Enables transmission of the idle code on the selected channel.



Do not set both this field and *disable_idle_code* to this value.

FALSE If both this field and *disable_idle_code* contain this value, transmission remains at its previous state.

disable_idle_code

You can select one of these:

TRUE Disables transmission of the idle code on the selected channel.



Do not set both this field and *enable_idle_code* to this value.

FALSE If both this field and *enable_idle_code* contain this value, transmission remains at its previous state.

enable_clear_channel

You can select one of these:

TRUE Configures the channel for eith 64Kbps (clear channel) or 56Kbps (non clear channel).



Do not set both this field and *disable_clear_channel* to this value.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

FALSE — If both this field and *disable_clear_channel* contain this value, the channel's clear/not-clear configuration remains at its previous state.

disable_clear_channel

You can select one of these:

TRUE — Configures the channel for eith 64Kbps (clear channel) or 56Kbps (non clear channel).



Do not set both this field and *enable_clear_channel* to this value.

FALSE — If both this field and *disable_clear_channel* contain this value, the channel's clear/not-clear configuration remains at its previous state.

update_supervision

You can select one of these:

TRUE — Specifies the transmitted robbed bit signaling bits (supervision bits) by changing the ABCD bit values specified in *x_signaling_bit*.



Do not set both this field and *enable_clear_channel* to this value.

FALSE — Supervision for the selected channel remains unchanged.

x_signaling_bit

You can select one of these:

TRUE — Sets the signaling bit to 1.



Do not set both this field and *enable_clear_channel* to this value.

FALSE — Sets the signaling bit to 0.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_T1_user_raw_signaling_struct

Reads the signaling bit (supervision bit) values for each channel on the specified T1 port.

Use in conjunction with the getT1SignalingRaw function.

Syntax

```
typedef struct {
   t_T1E1_framer_id        framer_id;
   uchar                   signaling_bits[12];
} t_T1_user_raw_signaling_struct;
```

Elements

framer_id
: Specifies which framer or port to read.

signaling_bits
: An array of 12 bytes, each of which contain the signaling bits for two DS0s.

Example

The high-order nibble of each byte contains the signaling bits for the lower numbered channel. For example:

```
signaling_bits[0] bit 7 = DS0-0 A bit
signaling_bits[0] bit 6 = DS0-0 B bit
signaling_bits[0] bit 5 = DS0-0 C bit
signaling_bits[0] bit 4 = DS0-0 D bit
signaling_bits[0] bit 3 = DS0-1 A bit
signaling_bits[0] bit 2 = DS0-1 B bit
signaling_bits[0] bit 1 = DS0-1 C bit
signaling_bits[0] bit 0 = DS0-1 D bit
signaling_bits[11] bit 7 = DS0-22 A bit
signaling_bits[11] bit 6 = DS0-22 B bit
signaling_bits[11] bit 5 = DS0-22 C bit
signaling_bits[11] bit 4 = DS0-22 D bit
signaling_bits[11] bit 3 = DS0-23 A bit
signaling_bits[11] bit 2 = DS0-23 B bit
signaling_bits[11] bit 1 = DS0-23 C bit
signaling_bits[11] bit 0 = DS0-23 D bit
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

T1StatusHandler

A type definition for a function pointer that points to a user handler that will receive a t_T1_user_status_data structure pointer as a parameter.

Syntax

```
typedef void
    (*T1StatusHandler)
    (t_T1_user_status_data *status_struct);
```

Elements

status_struct

Provides line status. For more information, see *t_T1_user_status_struct* on page 153.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

T1SignalingHandler

A type definition for a function pointer that points to a user handler that will receive a t_T1_user_signaling_data structure pointer as a parameter.

Syntax

```
typedef void
    (*T1SignalingHandler)
    (t_T1_user_signaling_data *signaling_struct);
```

Elements

signaling_struct

Reads and writes signaling data. For more information, see *t_T1_user_signaling_data, t_T1_signaling_data* on page 151.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_E1_line_coding

Enumerate the line coding for a framer (port).

Syntax

```
typedef enum {
    NON_HDB3
    HDB3
} t_E1_line_coding;
```

Elements

NON_HDB3	The framer uses non-HDB3 line coding.
HDB3	The framer uses HDB3 line coding.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_E1_signaling_mode

Enumerates the signaling mode for a framer (port).

Syntax

```
typedef enum {
    CAS,
    CCS
} t_E1_signaling_mode;
```

Elements

CAS	Channel Associated Signaling.
CCS	Common Channel Signaling.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_E1_line_buildout

Enumerates the line buildout for a framer (port).

Syntax

```
typedef enum {
    BUILDOUT_75_OHM,
    BUILDOUT_120_OHM
} t_E1_line_build_out;
```

Elements

BUILDOUT_75_OHM
Sets the framer (specifically the LIU portion) to 75OHM operation.

BUILDOUT_120_OHM
Sets the framer (specifically the LIU portion) to 120OHM operation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_E1_user_config_struct

Specifies the complete configuration for a framer (port).

Syntax

```
typedef struct {
    t_T1E1_framer_id        framer_id;
    t_E1_line_coding        line_coding;
    t_E1_signaling_mode     signaling_mode;
    BOOL32                  crc4_enable;
    t_E1_line_build_out     line_build_out;
    uchar                   idle_code;
    ulong                   idle_channels;
    BOOL32                  framer_loopback_enable;
    BOOL32                  local_loopback_enable;
    BOOL32                  remote_loopback_enable;
} t_E1_user_config_struct;
```

Elements

The first three fields and the line *line_build_out* field are described above in the type definitions.

framer_id Specifies a framer (port) for functions which can operate on any port. For more information, see *t_T1E1_framer_id* on page 134.

line_coding
Specifies the line coding for a framer (port). For more information, see *t_E1_line_coding* on page 161.

signaling_mode
Enumerates the signaling mode for a framer (port). For more information, see *t_E1_signaling_mode* on page 162.

crc4_enable
Sets the CRC4 multiframe mode. A value of TRUE sets the mode.

line_build_out
Specifies the line buildout for a framer (port). For more information, see *t_E1_line_buildout* on page 163.

idle_code Specifies the pattern to inject into transmitted data.

idle_channels
The lower 32 bits specify the channels to inject the pattern specified in *idle_code* into.

A 1 in any bit indicates that the pattern is to be injected. Bit 0 is associated with the first DS0 of the E1 line while bit 31 is associated with the last DS0.

framer_loopback_enable
Instructs the framer to perform a system-side loopback. Setting any field to TRUE enables the loopback.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

local_loopback_enable
: Instructs the framer to perform a system-side loopback. Setting any field to TRUE enables the loopback.

remote_loopback_enable
: Instructs the framer to perform a line-side loopback. Setting any field to TRUE enables the loopback.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

t_E1_user_signaling_data

Reads and writes signaling data.

This structure is identical to *t_T1_user_signaling_data* since the *t_E1_signaling_data* field is typedef'ed to *t_T1_signaling_data*. See description of the corresponding T1 structures. The only difference is that the E1 structure makes use of all 32 bits where as the T1 structure uses only 24, 1 per DS0.

Syntax

```
typedef struct {
    t_T1E1_framer_id        framer_id;
    t_E1_signaling_data     signaling_data;
} t_E1_user_signaling_data;
```

Elements

framer_id Specifies a framer (port) for functions which can operate on any port. For more information, see *t_T1E1_framer_id* on page 134.

signaling_data
Reads and writes signaling data. For more information, see *t_T1_user_signaling_data, t_T1_signaling_data* on page 151.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E1SignalingHandler

Type definition for user-specified signaling event handler.

Syntax

```
typedef void
    (*E1SignalingHandler)
    ( t_E1_user_signaling_data *signaling_struct);
```

Elements

signaling_struct

Reads and writes signaling data. For more information, see *t_E1_user_signaling_data* on page 166.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

10 Interrupt library

This document is the detailed design specification for the strongARM interrupt-handling library offered as part of ENP SDK. It describes the features and functionality provided by the library, its software architecture and the application programming interface (API) to the library. Present implementation of the library is in VxWorks real-time operating system.

Introduction

StrongARM core in IXP1200 has the following exception types listed in the order of their priority. Each causes program execution to switch to the address defined below. Out of these seven exceptions, IRQ is the only one addressed by this library. FIQ (Fast Interrupt) allows fast handling of a single interrupt. But, considering the large number of devices and the lack of support from VxWorks, this library doesn't use FIQ.

Exception type	Vector
Reset	0000 0000h
Undefined instructions	0000 0004h
Software interrupt	0000 0008h
Prefetch abort (instruction prefetch memory abort)	0000 000Ch
IRQ (interrupt)	0000 0010h
FIQ (fast interrupt)	0000 001Ch

The IXP1200 allows a number of sources to generate either an IRQ or FIQ. Each device can be programmed to generate either IRQ or FIQ. Since this library doesn't support FIQ, all devices are programmed to generate IRQ whenever an exception occurs. On ENP series boards the following MAC devices interrupt the strongARM core using the CINT pin generating an IRQ.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Figure 10-1. ENP interrupt sources

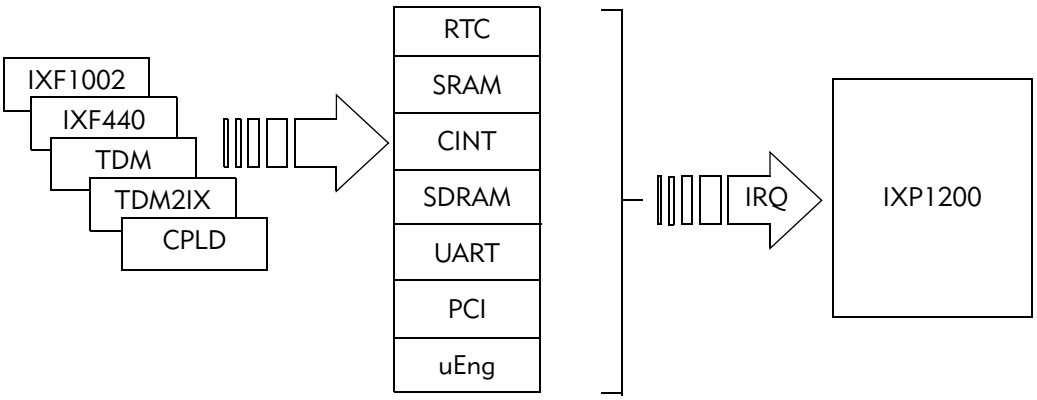


Table 10-1. IXP1200 interrupt vectors

Vector	Name	Comments
0	Reserved	
1	SI	Soft Interrupt (PCI)
2	Reserved	
3	Reserved	
4	T1	Timer1 (PCI)
5	T2	Timer2 (PCI)
6	T3	Timer3 (PCI)
7	T4	Timer4 (PCI)
8	Reserved	
9	CINT[1]	Where all the MAC devices are connected
10	uEng[1]	Uengine interrupts
11	SRAM	SRAM unit
12	RTC[1]	
13	SDRAM	
14	UART	
15	DFH	Doorbell from Host {PCI}
16	DMA1	DMA channel 1 {PCI}
17	DMA2	DMA channel 2 {PCI}
18	PIL	PCI_IRQ {PCI}
19	Reserved	
20	DMA1NB	DMA1 not busy {PCI}
21	DMA2NB	DMA1 not busy {PCI}
22	SB	Start BIST {PCI}
23	RSERR	Received SERR {PCI}
24	SDPAR	SDRAM Parity {PCI}
25	IIP	I20 inbound post_list

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 10-1. IXP1200 interrupt vectors

Vector	Name	Comments
26	Reserved	
27	DTE	Discard Timer Expired
28	DPED	Data Parity Error Detected
29	RMA	Received Master Abort
30	RTA	Received Target Abort
31	DPE	Detect Parity Error

[1] Interrupts handled by ENP-SDK.

Interrupt connection, enabling, and disabling are performed using the standard intArchLib routines.

Interrupts 9 through 14 (inclusive) cannot be enabled or disabled using the standard routines. These must be enabled or disabled by writing to the appropriate control register directly.

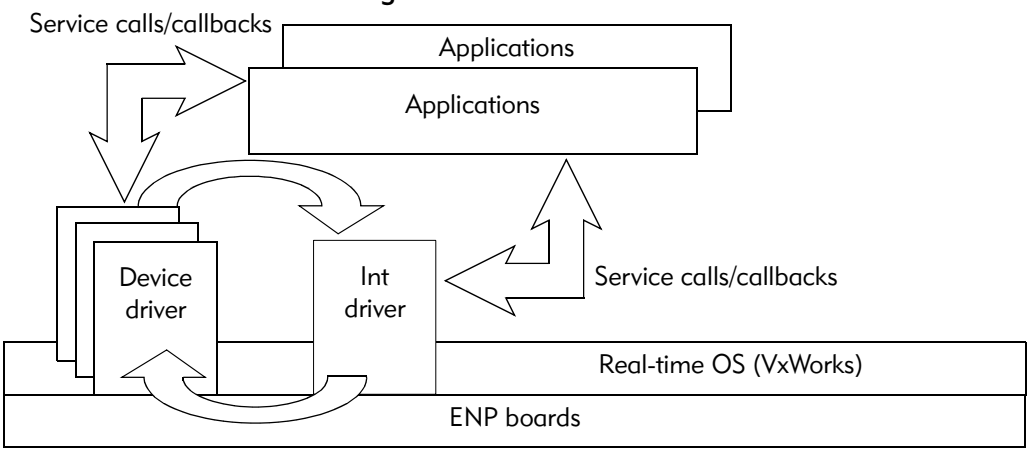
ENP-SDK Interrupt library complement the standard VxWorks intArchLib by providing routines for interrupt connection, enabling, or disabling of vectors 9, 10, and 12.

Software architecture

This library complements the intArchlib from VxWorks by providing additional routines to manage interrupts from various board specific sources in an ENP board. Interrupt from these sources are not handled by the VxWorks intArchLib in a device/thread specific manner. Since IXP1200 does not have a true interrupt controller. All interrupt processing must be in non-preemptive mode. Effectively there is only one interrupt level. This library uses a non-nested interrupt handling mechanism and relies entirely on VxWorks intArchLib routines for low-level support.

Each ENP board has its own unique set of devices which may include IXF440, IXF1002, TDM, TDM2ix, CPLD etc. It is not necessary that all the ENP boards have all these devices. Device drivers for these devices use the interrupt library to manage various events from respective devices. Some of the interrupt library routines (those deal with CPLD, uEng, etc.) can be directly accessible from applications.

Figure 10-2. Architecture



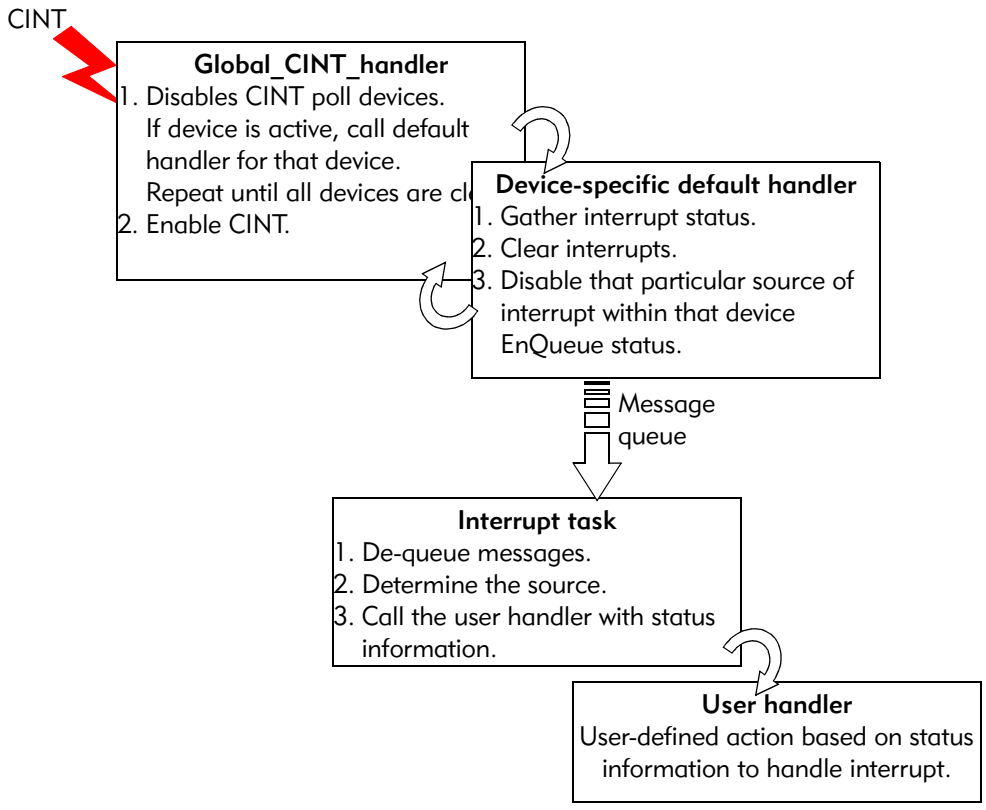
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CINT (MAC devices): interrupt handling

Multiple MAC devices connect to CINT pin in an ENP board. ENP SDK interrupt library services these device interrupts using an interrupt service routine (global_CINT_ISR) attached to the CINT_VECT(Vector 9). When an IRQ interrupt from CINT occurs, this routine is invoked. This routines polls all the MAC devices registered with it to find the interrupt source and calls the default handler for that particular device.

Device-specific default handler clears the interrupt, gets the necessary status information, and disables any further interrupts from that particular source within that device. The default handler then sends the status information to a message queue and returns. An interrupt Task waits for a messages from message queue. When it receives any message, it examines the message to determine the interrupt source, and calls its associated user handler, if any, with the status information.

Figure 10-3. CINT/Mac device interrupt handling



The following devices generate CINT on ENP boards:

Interrupt Source	ENP board
IXF440	ENP-3511, ENP-2505
TDM2IX	ENP-3511
TDM(T8105)	ENP-3511
CPLD	ENP-3511
IXF1002	ENP-2506

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MicroEngine interrupt handling

Twenty-four microengine threads (THD0 to THD23) and three debug interrupts (DBINT0 to DBINT2) can generate an IRQ through the UENG interrupt. Intel's debug1200.a library installs a global handler for UENG interrupts and uses DBINT0 to provide debugging support in the developer workbench. (for more information, see SA1_CoreLibs\ueng.cpp).

The Radisys ENP SDK provides these implementations for handling microengine interrupts:

- **Intel's global handler for microengine interrupts**: The ENP SDK provides wrapper functions that install interrupt handlers for individual microengine threads. This implementation supports debug interrupts from Intel developer workbench.

  To provide debug support, Intel's debug library (debug_1200.a) connects and installs a global handler for microengine IRQ (uengIsr). When a microengine IRQ occurs, this uengIsr is invoked. This routine checks IREG status to determine which microengine thread/threads or debug interrupt caused the interrupt. If a handler is installed for that particular thread or debug interrupt, uengIsr invokes that handler. To keep the debugging support, the ENP SDK interrupt library uses this mechanism and provides routines to install user handlers for each microengine thread.

  VxWorks_ENP3511.o/ VxWorks_ENP2505.o / VxWorks_ENP2506.o distributed as part of enp-sdk are customized versions of VxWorks_NetApps.o with the necessary ueng interrupt support so its important to use these instead of VxWorks_NetApps.o

- **RadiSys implementation**: This implementation does not support debug interrupts from the Developer Workbench, so you cannot use it during the debugging phase.

  The next figure describes the implementation. Whenever a microengine interrupt occurs, global_uengHandler is invoked. This handler reads the IREG status, clears the interrupts, then sends the interrupt status to a message queue. the uEngInterrupt task dequeues these messages and invokes user-installed thread-specific functions.

```
uEng_SetInterruptHandler(
    int threadID,
    void (*uEngUserHandler)(int),
int parm, BOOL use_enp_uint
)
```

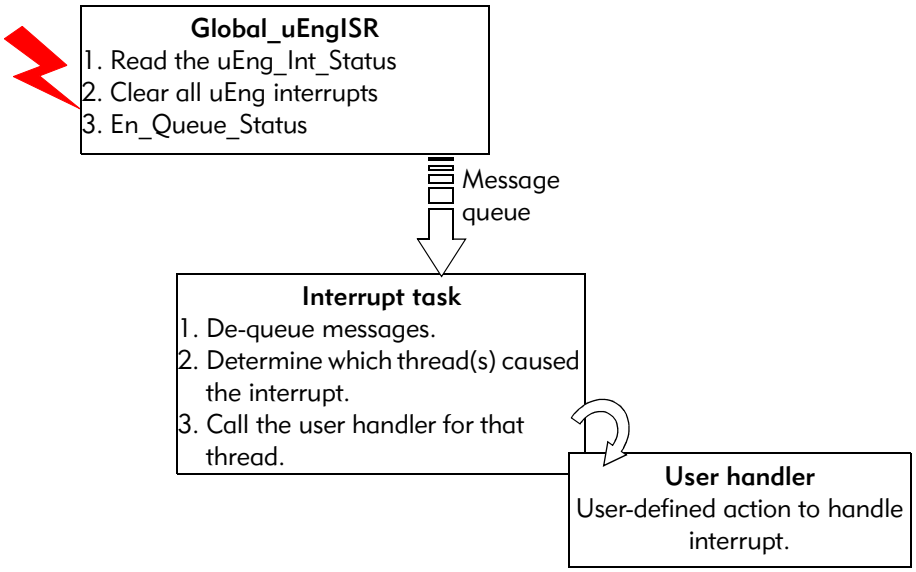
  The last parameter in uEng_SetInterruptHandler determines which version of the above mentioned implementation handles microengine interrupts:

`use_enp_uint = 1, TRUE`	Uses the RadiSys implementation.
`use_enp_uint = 0, FALSE`	Uses Intel's implementation with debugging support.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Be sure to use VxWorks_ENP2505.o/VxWorks_ENP2506.o instead of VxWorks_ENP2505.o/VxWorks_NetApp.o.

Figure 10-4. Microengine interrupt handling



InterruptStatus

```
{
   int status ;
   int device ;
   int portStatus[8];
}InterruptStatus
```

When a device generates an interrupt default handler for that device,this gathers all the interrupt-related information and fills the above structure. When the user handler is invoked, this structure is passed to the user handler for further processing of the interrupt.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Device-specific interrupt function list

Because the interrupt functions are device-specific, they are described in the chapter that documents that device's function library.

Use these tables to identify the interrupt functions you want to use. Use the function description to obtain detailed information, including syntax and parameter values.

Table 10-2. TDM interrupt functions (Chapter 8)

Function	Description	Page
TDMSetInterruptHandler	Installs a user-defined interrupt handler for TDM interrupts.	95

Table 10-3. ENP interrupt functions (Chapter 11)

Function	Description	Page
CPLD_DisableAll_Interrupt	Disables all CPLD interrupts.	181
CPLD_Disable_Interrupt	Disables a particular CPLD interrupt.	182
CPLD_EnableAll_Interrupt	Enables all CPLD interrupts.	183
CPLD_Enable_Interrupt	Enables a particular CPLD interrupt.	184
CPLD_SetInterruptHandler	Installs an interrupt handler for CPLD interrupts.	185
RTC_Disable_Interrupt	Disables real-time clock interrupts.	193
RTC_Enable_Interrupt	Enables real-time clock interrupts.	194
RTC_InitClock	Initializes the real-time clock.	195
RTC_Set_Int_frequency	Sets the real time clock interrupt frequency.	196
RTC_SetInterruptHandler	Installs a user-defined interrupt handler for real-time clock interrupts.	197
uEng_Enable_Interrupt	Enables microengine interrupts.	198
uEng_Disable_Interrupt	Disables microengine interrupts.	199
uEng_SetInterruptHandler	Installs an interrupt handler for the specified microengine thread.	200

Table 10-4. IXF440 microcode interrupt functions (Chapter 13)

Function	Description	Page
IXF440DisableAllInterrupt	Disables all IXF440 interrupt events.	225
IXF440DisableInterrupt	Disables a specific IXF440 interrupt event.	226
IXF440EnableAllInterrupt	Enables all IXF440 interrupt events.	227
IXF440EnableInterrupt	Enables a specific IXF440 interrupt event.	228
IXF440SetInterruptHandler	Installs an interrupt handler for IXF440 events.	240

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 10-5. IXF1002 microcode interrupt functions (Chapter 14)

Function	Description	Page
IXF1002DisableAllInterrupt	Disables all IXF1002 interrupt events.	270
IXF1002DisableInterrupt	Disables the specified IXF1002 interrupt event.	271
IXF1002EnableAllInterrupt	Enables all IXF1002 interrupt events.	272
IXF1002EnableInterrupt	Enables the specified IXF1002 interrupt event.	273
IXF1002SetInterruptHandler	Installs an interrupt handler for IXF1002 events.	286

Table 10-6. TDM2IX microcode interrupt functions (Chapter 15)

Function	Description	Page
TDM2IXDisableAllInterrupt	Disables all TDM2IX interrupt events.	302
TDM2IXDisableInterrupt	Disables a specific TDM2IX interrupt event.	303
TDM2IXEnableAllInterrupt	Enables all TDM2IX interrupt events.	304
TDM2IXEnableInterrupt	Enables a specific TDM2IX interrupt event.	305
TDM2IXSetInterruptHandler	Installs a user defined interrupt handler for TDM2IX interrupts.	313

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

11 ENP microcode resource manager

This document describes the ENP microcode resource manager functions.

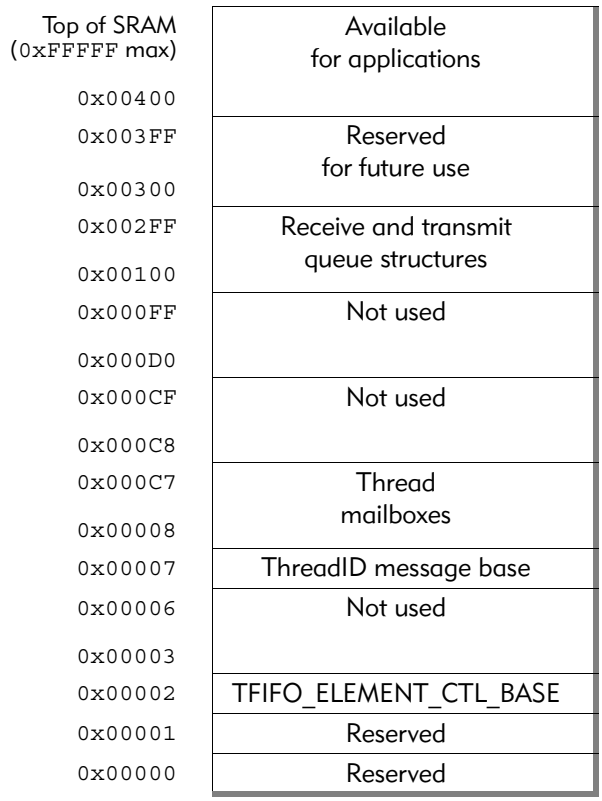
You can find a function in this chapter either by locating it alphabetically within its group, or by using Table 11-4.

Overview

You must initialize the ENP microcode resource manager before using any microcode device driver AAPI functions. For example, you initialize the ENP-3511 microcode resource manager using ENP3511_InitApp.

Memory maps

Figure 11-1. Memory map: SRAM (32-bit addressing)



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 11-1. Memory map: scratch (32-bit addressing)

0x3FF 0x090	Not used
0x08F 0x078	Transmit thread Epkt counters
0x077 0x060	Transmit thread Mpkt counters
0x05F 0x048	Receive thread Epkt counters
0x047 0x030	Receive thread Mpkt counters
0x02F 0x018	Transmit thread error message mailbox locations
0x017 0x000	Receive thread error message mailbox locations

Table 11-2. Memory map: SDRAM (64-bit addressing)

Top of SDRAM (0x1FFFFFF)	VxWorks (cached)
Top of SDRAM–16MB 0x0000000	Available for use (non-cached)

Interrupt routines

CPLD interrupt routines

Include File enpintrctl.h
Library Enp.a

CPLD on ENP boards generate CINT IRQ under various conditions. These interrupts are managed using the following routines. When a CPLD interrupt

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

occurs, CPLDUserHandler is called and a structure of type InterruptStatus is passed to that function. Various fields and their values for InterruptStatus include:

Table 11-3. InterruptStatus structure

Field	Value
InterruptStatus.device	INTERRUPT_FROM_CPLD.
InterruptStatus.status	CPLD Interrupt Pending/Clear register.
InterruptStatus.portStatus[i]	Not used.

Microengine interrupt routines

Any of the 24 microengine threads can interrupt the strongARM core. The interrupt library provides a way to install an interrupt handler of any specific thread. These routines make use of Intel's microengine IRQ handler to support debug interrupts in developer workbench.

Include File enpintrctl.h
Library Enp.a

RTC interrupt routines

Include File enpintrctl.h
Library Enp.a

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function list

Use this table to identify the ENP functions you want to use. Use the function description later in this chapter to obtain detailed information, including syntax and parameter values.

Table 11-4. ENP functions

Function	Description	Page
CPLD_DisableAll_Interrupt	Disables all CPLD interrupts.	184
CPLD_Disable_Interrupt	Disables a particular CPLD interrupt.	184
CPLD_EnableAll_Interrupt	Enables all CPLD interrupts.	184
CPLD_Enable_Interrupt	Enables a particular CPLD interrupt.	184
CPLD_SetInterruptHandler	Installs an interrupt handler for CPLD interrupts.	184
ENP2505_InitApp	Initializes hardware and data structures for ENP-2505 operation using the RadiSys supplied microcode drivers.	186
ENP2506_InitApp	Initializes hardware and data structures for ENP-2506 operation using the RadiSys supplied microcode drivers.	187
ENP3510_InitApp	This function is obsolete; it is included only for legacy purposes.	188
ENP3511_InitApp	Initializes hardware and data structures for ENP-3511 operation using the RadiSys supplied microcode drivers.	189
ENPCreateEpktStack	Creates a free list of Ethernet-compatible buffer descriptors and packet buffers using the IXP1200'S SRAM push/pop stack features.	190
ENPDestroyEpktStack	Destroys a free list of Epkt buffer descriptors created with ENPCreateEpktStack.	191
ENPInitThreadID	Sends a message to a specified thread to enable the thread to determine its thread ID.	192
RTC_Disable_Interrupt	Disables real-time clock interrupts.	193
RTC_Enable_Interrupt	Enables real-time clock interrupts.	194
RTC_InitClock	Initializes the real-time clock.	195
RTC_Set_Int_frequency	Sets the real time clock interrupt frequency.	196
RTC_SetInterruptHandler	Installs a user-defined interrupt handler for real-time clock interrupts.	197
uEng_Enable_Interrupt	Enables microengine interrupts.	198
uEng_Disable_Interrupt	Disables microengine interrupts.	199
uEng_SetInterruptHandler	Installs an interrupt handler for the specified microengine thread.	200

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CPLD_DisableAll_Interrupt

Disables all CPLD interrupts.

Syntax

```
t_ENPStatus CPLD_DisableAll_Interrupt(void);
```

Parameters

None.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CPLD_Disable_Interrupt

Disables a particular CPLD interrupt.

Syntax

```
t_ENPStatus CPLD_Disable_Interrupt (
   t_CPLDInterruptEvent Event
);
```

Parameters

Event Of type t_CPLDInterruptEvent:

t_CPLDInterruptEvent

CPLD_RTM0	Interrupt from RTM0
CPLD_RTM1	Interrupt from RTM1
CPLD_RTM2	Interrupt from RTM2
CPLD_RTM3	Interrupt from RTM3
CPLD_THERM	Thermal Alarm from IPMI
CPLD_IPMI	IPMI interrupt

Default handler clears the interrupts and disables any further interrupts from that specific source within the device, which caused the interrupt and passes the above information to the user handler. You must re-enable any interrupt, if needed.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CPLD_EnableAll_Interrupt

Enables all CPLD interrupts.

Syntax

```
t_ENPStatus CPLD_EnableAll_Interrupt(void);
```

Parameters

None.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CPLD_Enable_Interrupt

Enables a particular CPLD interrupt.

Syntax

```
t_ENPStatus CPLD_Enable_Interrupt(
   t_CPLDInterruptEvent Event
);
```

Parameters

Event Of type t_CPLDInterruptEvent:

t_CPLDInterruptEvent

CPLD_RTM0	Interrupt from RTM0
CPLD_RTM1	Interrupt from RTM1
CPLD_RTM2	Interrupt from RTM2
CPLD_RTM3	Interrupt from RTM3
CPLD_THERM	Thermal Alarm from IPMI
CPLD_IPMI	IPMI interrupt

Default handler clears the interrupts and disables any further interrupts from that specific source within the device, which caused the interrupt and passes the above information to the user handler. You must re-enable any interrupt, if needed.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CPLD_SetInterruptHandler

Installs an interrupt handler for CPLD interrupts.

Syntax

```
t_ENPStatus CPLD_SetInterruptHandler(
   void (*CPLDUserHandler)(InterruptStatus)
);
```

Parameters

Pointer to the user-defined interrupt-handling function.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENP2505_InitApp

Initializes hardware and data structures for ENP-2505 operation using the RadiSys supplied microcode drivers.

Syntax

```
t_ENPStatus ENP2505_InitApp( void );
```

Parameters

None.

Return values



For detailed information about this return value, see *Appendix A, Status codes*.

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENP2506_InitApp

Initializes hardware and data structures for ENP-2506 operation using the RadiSys supplied microcode drivers.

Syntax

```
t_ENPStatus ENP2506_InitApp( void );
```

Parameters

None.

Return values



For detailed information about this return value, see *Appendix A, Status codes*.

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENP3510_InitApp

This function is obsolete; it is included only for legacy purposes.

Initializes hardware and data structures for ENP-3510 operation using the RadiSys supplied microcode drivers.

Syntax

```
t_ENPStatus ENP3510_InitApp( void );
```

Parameters

None.

Return values



For detailed information about this return value, see *Appendix A, Status codes*.

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENP3511_InitApp

Initializes hardware and data structures for ENP-3511 operation using the RadiSys supplied microcode drivers.

Syntax

```
t_ENPStatus ENP3511_InitApp( void );
```

Parameters

None.

Return values



For detailed information about this return value, see *Appendix A, Status codes*.

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENPCreateEpktStack

Creates a free list of Ethernet-compatible buffer descriptors and packet buffers using the IXP1200'S SRAM push/pop stack features.

Each buffer descriptor uses four 32-bit SRAM words and each packet buffer uses 2048 SDRAM bytes.

Syntax

```
t_ENPStatus ENPCreateEpktStack(
    t_StackID StackID,
    int BufQty,
    Bit32 SramBase,
    Bit32 DramBase
);
```

Parameters

StackID Specifies the SRAM push/pop stack to use. Valid values include IXP_STACK0 to IXP_STACK7.

BufferQty Number of buffers to create.

SramBase Microengine base address (32bit addressing) of memory area in SRAM reserved for the descriptors.

DramBase Microengine base address (64bit addressing) of memory area in SDRAM reserved for the data buffers.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_STACKID
ENP_STACK_IN_USE
ENP_INVALID_SRAM_ADDRESS
ENP_INVALID_DRAM_ADDRESS

Comments

Buffer storage locations must not reside within the VxWorks memory space (upper 16MB of SDRAM on the ENP-3511). Application prorams are responsible for allocating non-VxWorks memory space and should not use malloc calls within VxWorks. For more information, see *Memory maps* on page 177.

See the *IXP1200 Hardware Reference Manual* for details on using SRAM push/pop stacks.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENPDestroyEpktStack

Destroys a free list of Epkt buffer descriptors created with ENPCreateEpktStack.

Syntax

```
t_ENPStatus ENPDestroyEpktStack(
    t_IXPStackID StackID
);
```

Parameters

StackID Specifies the SRAM push/pop stack to destroy.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_STACKID
ENP_INVALID_OBJECT
Comments

The stack is created with ENPCreateEpktStack.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ENPInitThreadID

Sends a message to a specified thread to enable the thread to determine its thread ID.

The specified thread must be running.

Syntax

```
t_ENPStatus ENPInitThreadID(
    Bit16 ThreadID
);
```

Parameters

ThreadID Specifies the thread to initialize. Valid values include 0 to 24.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_OBJECT
ENP_THREAD_NOT_READY

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RTC_Disable_Interrupt

Disables real-time clock interrupts.

Syntax

```
t_ENPStatus RTC_Disable_Interrupt (void);
```

Parameters

None.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RTC_Enable_Interrupt

Enables real-time clock interrupts.

Syntax

```
t_ENPStatus RTC_Enable_Interrupt(void);
```

Parameters

None.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RTC_InitClock

Initializes the real-time clock.

Initialization sets the interrupt frequency to one second.

Syntax

```
t_ENPStatus RTC_InitClock(void);
```

Parameters

None.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RTC_Set_Int_frequency

Sets the real time clock interrupt frequency.

Syntax

```
t_ENPStatus RTC_Set_Int_frequency(
   int frequency
);
```

Parameters

frequency

A non-zero integer value for real-time clock interrupt interval. The default is one second.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RTC_SetInterruptHandler

Installs a user-defined interrupt handler for real-time clock interrupts.

Syntax

```
t_ENPStatus RTC_SetInterruptHandler(
   void (*RTCUserHandler)(InterruptStatus iStatus)
);
```

Parameters

Pointer to a user-defined function.

Return values

ENP_SUCCESS
ENP_INT_HANDLER_INSTALL_FAILURE

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

uEng_Enable_Interrupt

Enables microengine interrupts.

Syntax

```
t_ENPStatus uEng_Enable_Interrupt(void);
```

Parameters

None.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

uEng_Disable_Interrupt

Disables microengine interrupts.

Syntax

```
t_ENPStatus uEng_Disable_Interrupt(void);
```

Parameters

None.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

uEng_SetInterruptHandler

Installs an interrupt handler for the specified microengine thread.

Syntax

```
t_ENPStatus uEng_SetInterruptHandler(
   int threadID,
   void (*uEngUserHandler)(int),
   int parm,
   BOOL use_enp_uint
);
```

Parameters

threadID MicroEngine thread number (0 to 23).

uEngUserHandler
Pointer to the interrupt handler function.

*parm*v Parameter passed to the interrupt handler when an interrupt occurs.

use_enp_uint
You can use one of these values:

TRUE Uses the RadiSys microengine debug features.

FALSE Uses the Intel microengine debug features.

Return value

ENP_SUCCESS



There is no default handler for microengine interrupts. You are responsible for processing interrupts. Be sure to clear the interrupt inside the handler.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

12 IXP1200 library

This document describes the IXP1200 library functions. This library provides access to commonly used IXP1200 hardware features.

You can find a function in this chapter either by locating it alphabetically within its group, or by using Table 12-1, which groups like functions together.

Function list

Use this table to identify the IXP1200 microcode functions you want to use. Use the function description later in this chapter to obtain detailed information, including syntax and parameter values.

Table 12-1. IXP1200 microcode functions

Function	Description	Page
IXPGetReadLockRetryStatus	Checks the read lock retry achieved bit in the SRAM_CSR register. If set it clears the bit and returns TRUE, else returns FALSE.	202
IXPPushSramStack	Pushes an element onto a specified SRAM push/pop stack	203
IXPPopSramStack	Pops an element off of a specified SRAM push/pop stack.	204
IXPReadLockSramA32	Locks the SRAM address and then reads it. Returns a lock failure if the addres was previously locked.	205
IXPReadReg	Reads an IXP register.	206
IXPReadScratchA32	Reads a word from Scratch memory.	207
IXPReadSramA32	Reads a word from SRAM.	208
IXPWriteReg	Writes an IXP register.	209
IXPWriteScratchA32	Writes a word to Scratch memory.	210
IXPWriteSramA32	Writes a word to SRAM.	211
IXPWriteUnlockSramA32	Writes a word to SRAM and then unlocks the address.	212

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPGetReadLockRetryStatus

Checks the read lock retry achieved bit in the SRAM_CSR register. If set it clears the bit and returns TRUE, else returns FALSE.

Syntax

t_ENPStatus IXPGetReadLockRetryStatus(void);

Parameters

None.

Return values

TRUE	A read lock retry was achieved.
FALSE	The read lock retry was not achieved.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPPushSramStack

Pushes an element onto a specified SRAM push/pop stack

Syntax

```
t_ENPStatus IXPPushSramStack(
    t_IXPStackID StackID,
    Bit32 Addr32
);
```

Parameters

StackID Specifies the SRAM push/pop stack to use. Valid values include IXP_STACK0 to IXP_STACK7.

Addr32 32-bit SRAM address of element to push starting at offset zero.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCES
ENP_INVALID_SRAM_ADDRESS
ENP_INVALID_STACKID

Comments

See the IXP1200 Hardware Reference Manual for details on using SRAM push/pop stacks.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPPopSramStack

Pops an element off of a specified SRAM push/pop stack.

Syntax

```
t_ENPStatus IXPPopSramStack(
    t_IXPStackID StackID,
    Bit32 *Addr32Ptr
)
```

Parameters

StackID Specifies the SRAM push/pop stack to use. Valid values include IXP_STACK0 to IXP_STACK7

Addr32Ptr Pointer to location to place the 32bit SRAM address of element popped.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_STACKID

Comments

See the IXP1200 Hardware Reference Manual for details on using SRAM push/pop stacks.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPReadLockSramA32

Locks the SRAM address and then reads it. Returns a lock failure if the addres was previously locked.

Syntax

```
t_ENPStatus IXPReadLockSramA32(
    Bit32 Addr32,
    Bit32 *DataPtr
);
```

Parameters

Addr32 32-bit address of SRAM data word to read starting at offset zero.

DataPtr Pointer to location to place data word read.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_SRAM_ADDRESS
ENP_LOCK_FAILURE

Comments

If the function returns a LOCK_FAILURE, the calling application must wait until the read lock retry achieved bit is set in the SRAM_CSR register. An interrupt is generated when this bit is set. You can also use IXPGetReadLockRetryStatus to poll this bit.

When a retry achieve is detected, the data must be re-read using the non-locking function. Only one process at a time should try to poll the read lock retry status.

Will block other threads attempting to read-lock memory while locked.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPReadReg

Reads an IXP register.

Syntax

```
t_ENPStatus IXPReadReg(
    Bit32 IXPReg,
    Bit32 *DataPtr
);
```

Parameters

IXPReg StrongARM address of register to read. See the IXP.h header file for valid values.

DataPtr Pointer to location to place the data word read.

Return values



For detailed information about this return value, see *Appendix A, Status codes*.

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPReadScratchA32

Reads a word from Scratch memory.

Syntax

```
t_ENPStatus IXPReadScratchA32(
    Bit32     Addr32,
    Bit32     *DataPtr
);
```

Parameters

Addr32 32-bit address of scratch data word to read starting at offset zero.

DataPtr Pointer to location to place data word read.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPReadSramA32

Reads a word from SRAM.

Syntax

```
t_ENPStatus IXPReadSramA32(
    Bit32 Addr32,
    Bit32 *DataPtr
);
```

Parameters

Addr32 32bit address of SRAM data word to read starting at offset zero.

DataPtr Pointer to location to place data word read starting at offset zero.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_SRAM_ADDRESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPWriteReg

Writes an IXP register.

Syntax

```
t_ENPStatus IXPWriteReg(
    Bit32 IXPReg,
    Bit32 Data
);
```

Parameters

IXPReg StrongARM address of register to read. See the IXP.h header file for valid values.

Data Data word to write.

Return values



For detailed information about this return value, see *Appendix A, Status codes*.

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPWriteScratchA32

Writes a word to Scratch memory.

Syntax

```
t_ENPStatus IXPWriteSramA32(
    Bit32 Addr32,
    Bit32 Data
)
```

Parameters

Addr32 32-bit address of Scratch data word to read starting at offset zero.

Data 32-bit data to write.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPWriteSramA32

Writes a word to SRAM.

Syntax

```
t_ENPStatus IXPWriteSramA32(
    Bit32 Addr32,
    Bit32 Data
)
```

Parameters

Addr32 32-bit address of SRAM data word to write starting at offset zero.

Data 32-bit data to write.

Comments



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_SRAM_ADDRESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXPWriteUnlockSramA32

Writes a word to SRAM and then unlocks the address.

Syntax

```
t_ENPStatus IXPWriteUnlockSramA32(
    Bit32 Addr32,
    Bit32 Data
);
```

Parameters

Addr32 32-bit address of SRAM data word to read starting at offset zero.

Data 32-bit data to write.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_SRAM_ADDRESS

Comments

Memory must be locked with IXPReadLockSramA32.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

13 IXF440 microcode library

This document describes the microcode modules and programming interface for the IXF440 and IX bus device.

You can find a function in this chapter either by locating it alphabetically within its group, or by using Table 13-4.

Overview

Microcode libraries

The programming interface consists of a library of C functions that run on the StrongARM core (AAPI) and a defined interface for microcode-to-microcode communications (µPI). The AAPI and microcode module work together with a microcode resource manager to simplify application development using the IXF440 devices on ENP platforms and future generations of embedded network processor boards.

Programming model

The ENP board's microcode software provides transmit and receive functionality for moving packets to and from the IXF440 bus devices on an ENP board. Microcode binary files transmit and receive on the IXP1200 microengines and provide the necessary infrastructure. A library of C function calls allows applications to configure the microcode programs for specific usage.

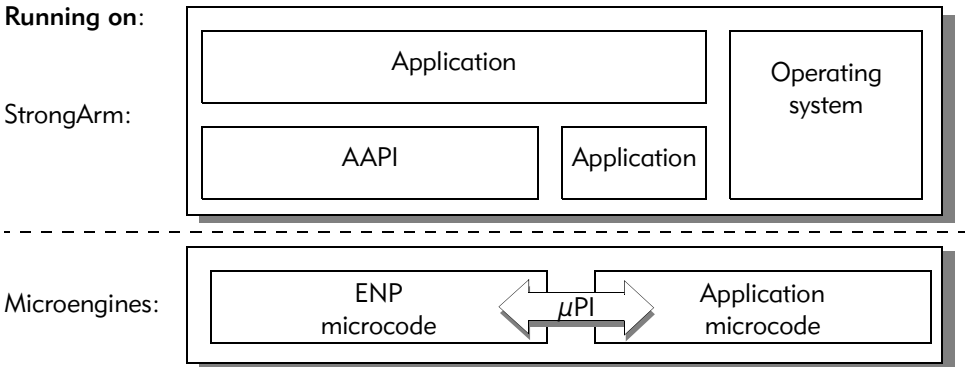


Figure 13-1. ENP board: IXF440 microcode programming model

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440 receive module

The IXF440 receive module contains four independent receive threads that run on a single microengine. The module can load on any microengine. Multiple microengines can load the same binary module to support multiple IXF440 devices.

Each thread within a module may be independently configured to receive data from a single IXF440 MacPort and to send an inter-thread signal to an application thread upon completed reception of a single Ethernet packet.

Packet data is stored in SDRAM buffers allocated by the StrongARM core prior to the start of any packet reception. Buffer descriptors contain details of the packet size, SDRAM buffer location, and transmit and receive specific details. Buffer descriptors are stored in SRAM and use the IXP1200 SRAM's push/pop features to maintain a free list of available buffers for storing packet data.

Received packets are placed into a single element receive queue. The thread or process that consumes the Ethernet packets must remove the received packet prior to the completed reception of a subsequent packet to avoid an overrun error. (The minimum packet size of 64 bytes takes approximately 6.8 microseconds to be received over 100Mb Ethernet. This is the average time between packets. The minimum time between packets received on the IX bus dithes, depending on when the ready bus is sampled, the time for the receive thread to issue a receive request, and the number of requests queued ahead of the current request.) Packets move from process to process by passing a pointer to a Buffer Descriptor. IXF440 receive threads idle if no data arrives.

The single element receive queue is a dedicated data word in SRAM. A pointer to a buffer descriptor is placed into this location upon completed reception of a packet. The location of the receive queue data word (RxBase) is reserved at compile-time. It can be accessed through AAPI C function calls.

Receive threads report several error conditions. The next table shows the possible error conditions. Upon detection of any error condition, threads generate an FIQ StrongARM interrupt. An application or interrupt handler can then use an AAPI function to read the error message generated by the thread. Note that the OS or application running on the StrongARM must enable FIQ interrupts.

Table 13-1. IXF440 receiver error conditions

Error	Description
Receive fail	A packet was received with errors causing an IXBus transfer in process to abort and the IXF440 MAC to discard the packet.
Receive error	Indicates a packet was not received OK by the MAC.
Rx overrun	Indicates that the prior packet was not removed from the receive queue element prior to reception of the current packet.
Out of buffers	Indicates that a receiver thread has requested a new buffer descriptor, but none are available.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The following statistics are reported by the receive threads for debug purposes:

- Rx Ethernet Packet Count
- Rx Mpkt Count

You use AAPI function calls to read the data. All counters are unsigned and roll over at 32bits.

IXF440 transmit module

The IXF440 Transmit Module contains four independent transmit threads that run on a single microengine. The module may be loaded on any microengine. Multiple microengines can be loaded with the same binary module to support multiple IXF440 devices.

Each thread within a module may be independently configured to transmit data to a single IXF440 MacPort.

Transmit packets are placed into a variable length transmit queue implemented as a linked list of pointers to Buffer Descriptors. Actual packet data is stored in SDRAM buffers allocated by the StrongARM core prior to the start of any packet reception or transmission. Buffer Descriptors contain details of the Packet size, the SDRAM buffer location and transmit and receive specific details. Buffer Descriptors are stored in SRAM and use the push/pop features of the IXP1200 SRAM unit to maintain a free list of available buffers for storing packet data. Packets move from process to process by passing a pointer to a Buffer Descriptor. The transmit thread automatically pushes a Buffer Descriptor back onto the SRAM stack after it is completely transmitted.

The transmit queue may be accessed from the StrongARM core via AAPI functions. The location of the transmit queue Head and Tail pointers and current size is reserved at compile-time of the AAPI library.

Transmit threads report several error conditions. The next table shows the possible error conditions. On detection of any error condition, threads generate an FIQ StrongARM interrupt. An application or interrupt handler can then use an AAPI function to read the error message generated by the thread (see IXF440 functions for further details). Note that the OS or application running on the StrongARM must enable FIQ interrupts.

Table 13-2. IXF440 transmitter error conditions

Error	Description
TxQueueHighWaterMark	The transmit queue contains a number of packets equal to or greater than the high water mark. This is configurable by the application.

The following statistics are reported by the transmit threads for debug purposes:

- Tx Ethernet Packet Count
- Tx Mpkt Count

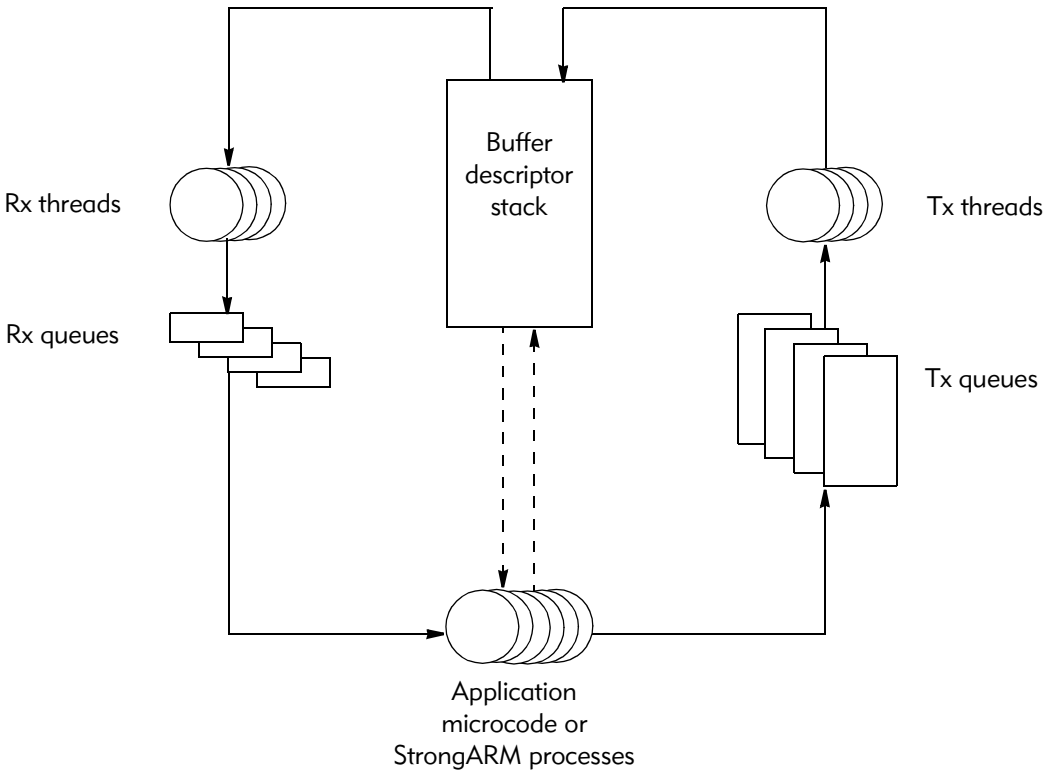
AAPI function calls read the data. All counters roll over at 32bits.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440 data flow

This figure shows the data flow for IXF440 Ethernet packets:

Figure 13-2. IXF440 packet data flow



A receive thread pops a free buffer descriptor from an SRAM stack when it detects the start of a new packet. When the packet is fully received, a pointer to the buffer descriptor is placed into the single element receive queue. Application processes remove the pointer from the receive queue, perform any application processing and then pass the packet onto a transmit thread or may discard it directly by pushing the descriptor back onto the buffer descriptor stack.

Multiple threads may access a single descriptor stack, or use independent stacks.

Using the IFX440 microcode drivers

Start-up and initialization

To start the IFX440 transmit and receive modules the following actions are required.

1. Initialize the ENP SDK microcode run-time environment by calling ENP3511_InitApp(). This should be completed before any microcode is allowed to begin normal execution. Note that all ENP SDK supplied microcode modules do not begin normal execution until signaled to do so via AAPI function calls.
2. Create a free list of Epkt buffers descriptors to be used for each port.
3. Load and start the microcode modules to be used.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

4. Open a handle to each IFX440 port to be used.
5. Initialize any desired hardware settings for each IFX440 port.
6. Hook each port to the desired receive thread, transmit thread, Epkt stack, and receive signal thread.
7. Enable receive and/or transmit operation for each port.



Ensure that no inter-thread communications occur until after a port is hooked.

Example code

The following microcode source listing demonstrates how to interface to the IFX440 receive and transmit modules.

```
.while(1)

       // Wait for inter thread signal
       ctx_arb[inter_thread]

       //Get the epkt from the rx queue and then write back zero so that
the rx thread
       //can check for an overrun condition
       sram[read_lock, $xfer1, RxQBase, 0, 1], ctx_swap
       immed[$xfer0, 0]
       sram[write_unlock, $xfer0, RxQBase, 0, 1]; no swap
       alu[Epkt, --, b, $xfer1]

       //pretend to do something
       readEpktDesc( Epkt, EPKT_DRAMBASE, $xfer0, 1 )
       sdram[read, $$xfer0, $xfer0, 0, 4], ctx_swap

       //now put the descriptor into the tx queue
       putQueue( TxQBase, Epkt, $xfer2, $xfer3, $xfer4 )
       SignalThread( TargetThread )

.endw // (FOREVER)
```

The thread above wakes up when it sees an inter-thread signal from the IFX440 receive thread. It then removes the Epkt pointer from the designated receive queue, writes back NULL, and pretends to do something. The packet is then placed into the transmit queue for the target process. This application simply reads a packet from a receive queue and forwards it to a designated transmit queue.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

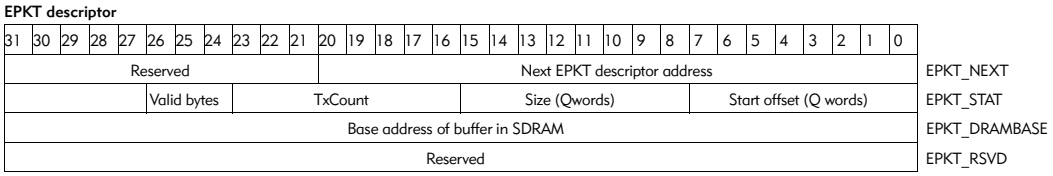
Managing packets

Epkt descriptor

Data structures located in SRAM used to describe and manage the Ethernet data packets.

Each Epkt descriptor occupies four words. There are a total of six fields currently used within the four words.

Figure 13-3. Epkt descriptor format



Next Epkt Descriptor Address
: Maintains a linked list structures of Epkt descriptors. An SRAM push/pop stack uses this filed to point to the next item in the stack. IXF440 transmit queues use the field to point to the next item in the queue.

Start Offset
: Informs the transmit thread where the first word to transmit is located within the 2K DRAM buffer. Informs the receive thread where to place the first word received within the 2K DRAM buffer. This field deletes or inserts information into the packet without physically moving data in DRAM. To insert information, the receive thread indicates a start offset of some number of quad words. After insertion, the start offset is modified to indicate the new start offset. All packet data must be contiguous within DRAM.

Size
: The total number of full and partial quadwords of packet data. Only the last quad word of data can be partial.

TxCount
: Not used at this time

Valid Bytes
: The number of valid bytes in the last quad word of data starting a base 0 (i.e a value of 0 indicates 1 valid byte and a value of seven indicates all eight bytes are valid)

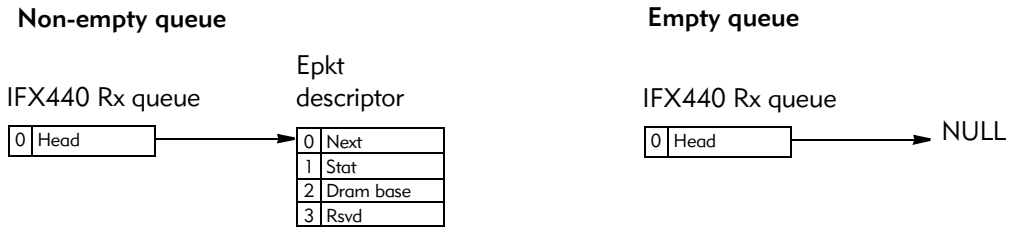
Base Address of Buffer in SDRAM
: The base address of the contiguous 2048 byte SDRAM buffer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440 receive queue

A list of Epkt descriptors. Each queue uses three SRAM words to describe the current Head, Tail, and Size state

Figure 13-4. IXF440 receive queue format

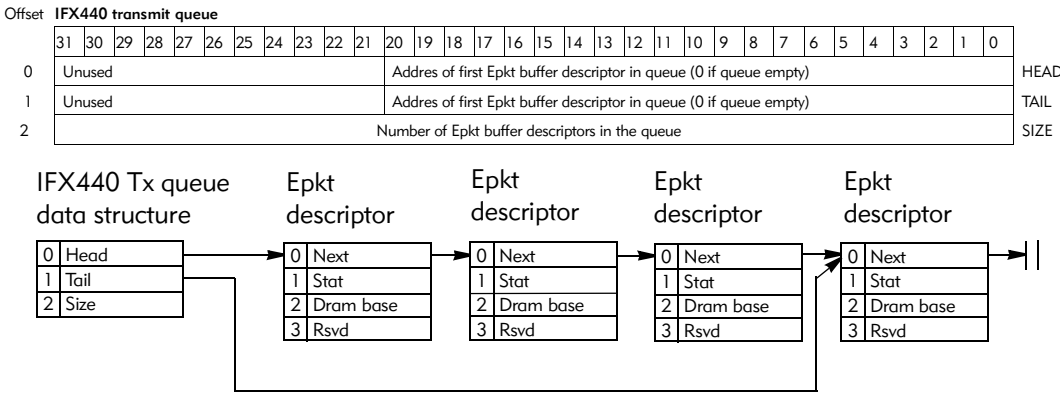


The semantics of the IXF440 receive queue require that the process removing the buffer descriptor pointer write back NULL to the queue. This allows the receive thread to determine if there is an overrun condition.

IXF440 transmit queue

Variable length linked lists of Epkt descriptors. Each queue uses three SRAM words to describe the current Head, Tail, and Size states.

Figure 13-5. IXF440 transmit queue format



Items are inserted into a queue at the tail using the IXF440WritePacket function call on the StrongARM and the IXF440PutTxQueue macro on the microengines.

IXF440 uPI

The microcode programming interface to the IXF440 receive and transmit modules is extremely simple. On the receive side, a target thread (that will process the receive packets) waits for a signal from the IXF440 receive thread and then accesses the receive queue. On the transmit side, a source thread (that has prepared a packet to send) will access the transmit queue and then signal the transmit thread. Each of these operations is detailed further.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440 Receive thread uPI

There are two components to the IXF440 receive thread uPI:

- Receiving an inter-thread signal
- Accessing the receive queue

The IXF440 receive thread sends an inter-thread signal to a specified thread when it has completely received a packet into its receive queue. It is possible to specify that no thread is signaled by the IXF440 receive thread, however this will require the target process to poll the receive queue to determine when a packet is available which will reduce system performance.

The IXF440 receive queue is accessed via SRAM read-lock and write-unlock commands from the target process. When the target process has removed the Epkt pointer from the IXF440 receive queue it must write back a zero to the queue. The receive thread will check the contents of the receive queue just prior to placing a new Epkt pointer into the queue and if the contents are non-zero it will signal an Rx Buffer Overrun error condition. Locking is required to prevent a NULL write back from the target process from overwriting a newly received Epkt pointer.

The target process may be a micro engine or the StrongARM core. If the StrongARM core is the target process, there is no mechanism to signal the core (could be a future implementation). The AAPI IXF440ReadPacket() function call should be used to poll the queue. It will return a value indicating whether or not the queue was empty.

IXF440 transmit thread uPI

There are two components to the IXF440 transmit thread uPI:

- Accessing the transmit queue
- Sending an inter-thread signal

The IXF440 transmit queue is accessed from the micro engines via the putQueue macro included in the enp_macro.uc file. The putQueue macro uses SRAM locking to ensure that simultaneous accesses to the queue do not destroy the integrity of the linked list structure. Transmit queues may also be accessed by the StrongARM via the IXF440WritePackt() AAPI function call.

Once an element is placed into an IXF440 transmit queue, an inter-thread signal must be sent to the transmit thread. The inter-thread signal is used to wake up the IXF440 transmit thread if it has become idle due to lack of data to send. There is no risk of over-writing a previous inter-thread signal and losing synchronization, because the transmitter will continue to check for data in the transmit queue unit it becomes empty. Once the queue does become empty, the thread will stop polling and go to sleep. This reduces unnecessary load on the SRAM unit that would occur if we continuously polled an empty queue.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440 interrupt routines

Include File Ixf440.h

Library Ixf440.a

Each MAC port in IXF440 can generate various types of interrupts. For more information, see the IXF440 Data sheet.

When an IXF440 interrupt occurs, IXF440UserHandler is called and a structure of type InterruptStatus is passed to that function.

Table 13-3. InterruptStatus structure

Field	Value
InterruptStatus.device	INTERRUPT_FROM_IXF440
InterruptStatus.status	IXF440_CHIP_INT_REG
InterruptStatus.portStatus[i] First byte (bits 0–7) Second byte (bits 8–15) Third byte (bits 16–23)	status information of ith port IXF440_INT_STT_REG IXF440_RECV_STT_REG IXF440_TXMT_STT_REG

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function list

Use this table to identify the IXF440 functions and structures you want to use. Use the function or structure description later in this chapter to obtain detailed information, including syntax and parameter or element values.

Table 13-4. IXF440 microcode functions

Function	Description	Page
IXF440ClearNetStats	Clears the network statistics counters from an IXF440 port.	223
IXF440ClosePort	Closes an IXF440 port.	224
IXF440DisableAllInterrupt	Disables all IXF440 interrupt events.	225
IXF440DisableInterrupt	Disables a specific IXF440 interrupt event.	226
IXF440EnableAllInterrupt	Enables all IXF440 interrupt events.	227
IXF440EnableInterrupt	Enables a specific IXF440 interrupt event.	228
IXF440GetDuplexMode	Gets the current duplex mode of an IXF440 port.	229
IXF440GetRxThreadError	Gets an error message from an IXF440 receive thread.	230
IXF440GetRxTxMode	Gets the current receive and transmit operation of an IXF440 port.	231
IXF440GetTxThreadError	Gets an error message from an IXF440 transmit thread.	232
IXF440HookPort	Connects a MacPort to a receive and/or transmit thread for IX Bus operation.	233
IXF440OpenPort	Opens an IXF440 port, resets the hardware, and returns a handle to the port	234
IXF440QueryNetStats	Reads the non-resetting network statistics from an IXF440 port.	235
IXF440ReadPacket	Reads a packet from an IXF440 receive queue.	236
IXF440ReadReg	Reads an IXF440 register.	237
IXF440ResetPort	Resets an IXF440 port.	238
IXF440SetDuplexMode	Sets full or half duplex operation on an IXF440 port.	239
IXF440SetInterruptHandler	Installs an interrupt handler for IXF440 events.	240
IXF440StartPort	Enables the receive and/or transmit operation of an IXF440 port.	241
IXF440StopPort	Disables receive and transmit operation of an IXF440 port.	242
IXF440UnhookPort	Disconnects the MacPort from the previously bound receive (Rx) and/or transmit (Tx) threads.	243
IXF440WritePacket	Writes a packet into an IXF440 transmit queue.	244
IXF440WriteReg	Writes an IXF440 register.	245

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440ClearNetStats

Clears the network statistics counters from an IXF440 port.

Syntax

```
t_ENPStatus IXF440ClearNetStats(
    t_IXF440PortHandle IXF440PortHandle
)
```

Parameters

IXF440PortHandle
Specifies an IXF440 port.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440ClosePort

Closes an IXF440 port.

Syntax

```
t_ENPStatus IXF440ClosePort (
    t_ENPPortHandle  PortHandle
)
```

Parameters

PortHandle

Specifies an IXF440 port.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440DisableAllInterrupt

Disables all IXF440 interrupt events.

Syntax

```
IXF440DisableAllInterrupt(
   t_ENPPortHandle PortHandle
);
```

Parameters

PortHandle

Handle to the IXF440 port.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440DisableInterrupt

Disables a specific IXF440 interrupt event.

Syntax

```
IXF440DisableInterrupt(
   t_ENPPortHandle PortHandle,
   t_IXF440InterruptEvent Event
);
```

Parameters

PortHandle

Handle to the IXF440 port.

Event IXF440 event to disable:

t_IXF440InterruptEvent

IXF440_RXOKE	Receive OK
IXF440_RXEVE	Receive Event
IXF440_TXOKE	Transmit OK
IXF440_TXERE	Transmit Error
IXF440_LNCHE	Link Change
IXF440_OVFCE	Counter overflow
IXF440_STOPE	Port Stop

The default handler clears the interrupts and disables any further interrupts from that specific source within the device, which caused the interrupt and passes the above information to the user handler. You must re enable any interrupt, if needed.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440EnableAllInterrupt

Enables all IXF440 interrupt events.

Syntax

```
IXF440EnableAllInterrupt(
   t_ENPPortHandle PortHandle
);
```

Parameters

PortHandle

Handle to the IXF440 port.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440EnableInterrupt

Enables a specific IXF440 interrupt event.

Syntax

```
IXF440EnableInterrupt(
   t_ENPPortHandle PortHandle,
   t_IXF440InterruptEvent Event
);
```

Parameters

PortHandle

Handle to the IXF440 port.

Event IXF440 event to enable:

t_IXF440InterruptEvent

IXF440_RXOKE	Receive OK
IXF440_RXEVE	Receive Event
IXF440_TXOKE	Transmit OK
IXF440_TXERE	Transmit Error
IXF440_LNCHE	Link Change
IXF440_OVFCE	Counter overflow
IXF440_STOPE	Port Stop

The default handler clears the interrupts and disables any further interrupts from that specific source within the device, which caused the interrupt and passes the above information to the user handler. You must re enable any interrupt, if needed.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440GetDuplexMode

Gets the current duplex mode of an IXF440 port.

Syntax

```
t_ENPStatus IXF440GetDuplexMode(
    t_ENPPortHandle  PortHandle,
    t_IXF440DuplexMode *DuplexMode
)
```

Parameters

PortHandle Specifies an IXF440 port.

DuplexMode

Pointer to location to place the current duplex mode of an IFX440 port. Valid values include:

IXF440_FULL_DUPLEX
IXF440_HALF_DUPLEX

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port must be opened with IXF440OpenPort.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440GetRxThreadError

Gets an error message from an IXF440 receive thread.

Syntax

```
t_ENPStatus IXF440GetRxError(
    BIT16            ThreadID,
    t_ENPPortHandle  *PortHandle,
    t_IXF440RxError  *RxError
)
```

Parameters

ThreadID Specifies the thread to get the error message from.

PortHandle Pointer to location to store the port handle associated with the error.

RxError Pointer to location to store the error message. Valid values include:

IXF440_RX_FAIL
IXF440_RX_ERR
IXF440_RX_OVERRUN
IXF440_RX_OUT_OF_BUFFERS

Return values



For detailed information about this return value, see *Appendix A, Status codes*.

ENP_SUCCESS

Comments

Thread interrupt was received by StrongARM.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440GetRxTxMode

Gets the current receive and transmit operation of an IXF440 port.

Syntax

```
t_ENPStatus IXF440GetRxTxMode(
    t_ENPPortHandle  PortHandle,
    t_IXF440RxTxMode *RxTxMode
)
```

Parameters

PortHandle Specifies an IXF440 port.

RxTxMode Pointer to location to place the current RxTx mode of an IXF440 port. Valid values include:

IXF440_RXONLY
IXF440_TXONLY
IXF440_RXANDTX
IXF440_STOP

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port must be opened with IXF440OpenPort.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440GetTxThreadError

Gets an error message from an IXF440 transmit thread.

Syntax

```
t_ENPStatus IXF440GetTxError(
    BIT16            ThreadID,
    t_ENPPortHandle  *PortHandle,
    t_IXF440TxError  *TxError
)
```

Parameters

ThreadID Specifies the thread to get the error message from.

PortHandle Pointer to location to store the port handle associated with the error.

TxError Pointer to the location to store the error message. Valid values include IXF440_TX_QUEUE_HIWATER.

Return values



For detailed information about this return value, see *Appendix A, Status codes*.

ENP_SUCCESS

Comments

Thread interrupt was received by StrongARM.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440HookPort

Connects a MacPort to a receive and/or transmit thread for IX Bus operation.

Assigns a thread to signal upon completed receive of a packet. Assigns an SRAM stack to use as a free list of buffer descriptors for receive and transmit threads.

Syntax

```
t_ENPStatus IXF440HookPort(
    t_ENPPortHandle  PortHandle,
    Bit16            RxThreadID,
    Bit16            TxThreadID,
    t_IXPStackID     StackID,
    Bit16            RxSigThreadID
)
```

Parameters

PortHandle Specifies an IXF440 port.

RxThreadID
ThreadID of the receiver thread to use. Valid values include 0 to 24 or IXP_THREADNONE.

TxThreadID
ThreadID of the transmitter thread to use. Valid values include 0 to 24 or IXP_THREADNONE

StackID Specifies the SRAM push/pop stack to use. Valid values include IXP_STACK0 to IXP_STACK7.

RxSigThreadID
ThreadID of thread to signal upon completed packet reception. Valid values include 0 to 24 or IXP_THREADNONE.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_INVALID_OBJECT
ENP_INSUFFICIENT_RESOURCES
ENP_THREAD_NOT_READY

Comments

IXF440 receiver and/or transmitter microcode must be running on the microengines. An EpktStack must exist for use by the receiver and transmitter.

A port opened in read only or write only must have existing threads to source or sink the new/used buffer descriptors.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440OpenPort

Opens an IXF440 port, resets the hardware, and returns a handle to the port

Syntax

```
t_ENPStatus IXF440OpenPort (
    Bit16            MacNum,
    Bit16            PortNum,
    t_ENPPortHandle  *PortHandle
)
```

Parameters

MacNum Mac number of the IXF440 device. Valid values include:

0 or IXP_MAC0
(this is the only IXF440 device on the ENP3511).

PortNum Port number on the Mac device. Valid values include 0 to 7.

PortHandle Pointer to location where the ENPPortHandle will be returned.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_PORT_ALREADY_OPEN
ENP_INVALID_MACPORT
ENP_INVALID_PORT_TYPE

Comments

You must call an initializing function (ENP2505_InitApp, ENP2506_InitApp, or ENP3511_InitApp) before any ports are opened.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440QueryNetStats

Reads the non-resetting network statistics from an IXF440 port.

Syntax

```
t_ENPStatus IXF440QueryNetStats(
    t_IXF440PortHandle IXF440PortHandle,
    t_IXF440NetStat  *IXF440NetStat
)
```

Parameters

IXF440PortHandle
Specifies an IXF440 port.

IXF440NetStat
Pointer to the t_IXF440NetStat structure where the network statistics will be placed. See the IXF440.h header file for additional details.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440ReadPacket

Reads a packet from an IXF440 receive queue.

Syntax

```
t_ENPStatus IXF440ReadPacket(
    t_ENPPortHandle  PortHandle,
    Bit32            *ReadPacket
)
```

Parameters

PortHandle
Specifies an IXF440 port.

ReadPacket Pointer to location where the address of the buffer descriptor read from the receive queue will be placed.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_INVALID_SRAM_ADDRESS
ENP_RXQUEUE_EMPTY

Comments

Port is open.

This function is used for debug. Typically, the target thread for the receive would handle queuing packes to the core.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440ReadReg

Reads an IXF440 register.

Syntax

```
t_ENPStatus IXF440ReadReg(
    t_ENPPortHandle  PortHandle,
    Bit16            RegOffset,
    Bit8             *ReadValue
)
```

Parameters

PortHandle
Specifies an IXF440 port.

RegOffset Specifies an IXF440 register. See the IXF440.h header file for valid values.

ReadValue Pointer to location to place read data.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440ResetPort

Resets an IXF440 port.

Syntax

```
t_ENPStatus IXF440ResetPort(
    t_IXF440PortHandle IXF440PortHandle
)
```

Parameters

IXF440PortHandle
Specifies an IXF440 port.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440SetDuplexMode

Sets full or half duplex operation on an IXF440 port.

Syntax

```
t_ENPStatus IXF440SetDuplexMode(
    t_ENPPortHandle  PortHandle,
    t_IXF440DuplexMode DuplexMode
)
```

Parameters

PortHandle Specifies an IXF440 port.

DuplexMode Specifies full or half-duplex operation. Valid values are:

IXF440_FULL_DUPLEX
IXF440_HALF_DUPLEX

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_INVALID_OBJECT
ENP_INVALID_OPERATION

Comments

The port must be stopped before calling this function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440SetInterruptHandler

Installs an interrupt handler for IXF440 events.

Syntax

```
IXF440SetInterruptHandler(
   void (*IXF440UserHandler)(InterruptStatus)
);
```

Parameters

Pointer to the user-defined interrupt-handling function.

Return value

ENP_SUCCESS
ENP_INT_HANDLER_INSTALL_FAILURE

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440StartPort

Enables the receive and/or transmit operation of an IXF440 port.

Syntax

```
t_ENPStatus IXF440SetRxTxMode(
    t_ENPPortHandle PortHandle,
   t_IXF440RxTxMode RxTxMode
)
```

Parameters

PortHandle
Specifies an IXF440 port.

RxTxMode Specifies Rx enable, Tx enable, or both. Valid values include:

IXF440_RXONLY
IXF440_TXONLY
IXF440_RXANDTX

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_INVALID_OBJECT

Comments

Port must be opened with IXF440OpenPort.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440StopPort

Disables receive and transmit operation of an IXF440 port.

Syntax

```
t_ENPStatus IXF440SetRxTxMode(
    t_ENPPortHandle PortHandle
)
```

Parameters

PortHandle Specifies an IXF440 port.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

You must open the port with IXF440OpenPort.

Stop state is reported in the Interrupt Status register. You must either verify that this bit is set after the port stops, or add enough delay to allow the current packet to complete reception or transmission.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440UnhookPort

Disconnects the MacPort from the previously bound receive (Rx) and/or transmit (Tx) threads.

Syntax

```
t_ENPStatus IXF440UnhookPort(
    t_ENPPortHandle  PortHandle
)
```

Parameters

PortHandle Specifies the IXF440 port to unhook.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_THREAD_NOT_READY

Comments

Port was hooked.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440WritePacket

Writes a packet into an IXF440 transmit queue.

Syntax

```
t_ENPStatus IXF440WritePacket(
    t_ENPPortHandle  PortHandle,
    Bit32            WritePacket
)
```

Parameters

PortHandle Specifies an IXF440 port.

*WritePacket*Address of buffer descriptor to write to the transmit queue.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_INVALID_SRAM_ADDRESS

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440WriteReg

Writes an IXF440 register.

Syntax

```
t_ENPStatus IXF440WriteReg(
    t_ENPPortHandle  PortHandle,
    Bit16            RegOffset,
    Bit8             WriteValue
)
```

Parameters

PortHandle
Specifies an IXF440 port.

RegOffset Specifies an IXF440 register. See the IXF440.h header file for valid values.

WriteValue
8-bit data to write.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

14 IXF1002 microcode library

This chapter describes the IFX1002 microcode device driver modules and programming interface provided with the ENP SDK.

The programming interface consists of:

- A library of C functions that run on the StrongARM core (AAPI)
- A defined interface to the microcode modules (uPI).

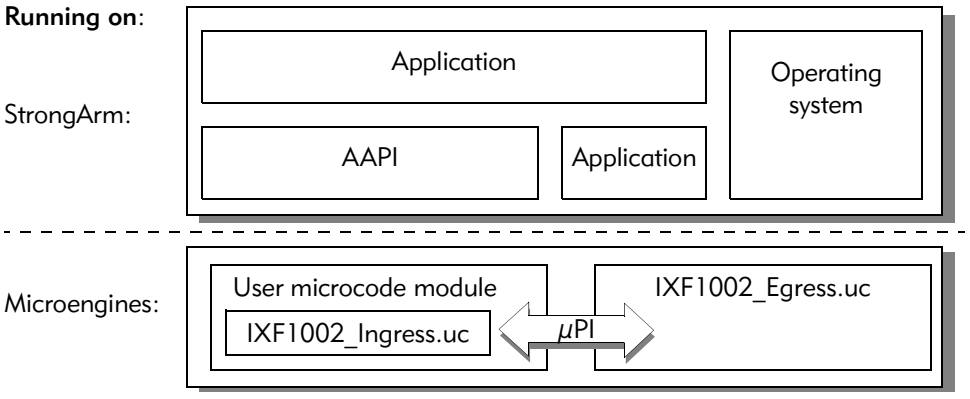
The AAPI and uPI work in conjunction with a microcode resource manager and are designed to simplify application development using IFX1002 devices on the ENP-2506 and future generations of RadiSys ENP boards.

Programming model

The IXF1002 microcode programming model differs from the IXF440 and TDM2IX microcode device drivers. The IXF1002 microcode ingress device driver is released as a macro to allow packet processing within the same thread as the packet reception. To execute, the ingress macro must be incorporated into a user-defined microcode source file. The IXF1002 egress driver remains a stand-alone microcode module.

The next figure shows the programming model for the IFX1002 microcode device drivers.

Figure 14-1. ENP board: IFX1002 microcode programming model



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Functional details

Overview

The IFX1002 microcode drivers provide transmit and receive functionality for moving packets to and from IFX1002 IX bus devices on an IXP1200-based network processor board. Microcode source code handles transmitting and receiving on the IXP1200 microengines as well as the necessary infrastructure. A library of C function calls allows applications to configure the microcode modules for specific usage at run time.

IFX1002 ingress module

Overview

The IXF1002 ingress module is designed for packet reception and application processing within the same thread. User-defined source modules call the IFX1002 ingress module to receive the next *Mpkt* of data from the port on the IX bus. The source module then branches to the correct processing algorithm based on the return status of the ingress macro.

A total of eight threads are reserved for ingress and application processing of a single IXF1002 port. Any two microengines may be utilized with the same ingress macro function called in all threads. Multiple IXF1002 devices are supported, up to the limitations of the IXP1200 chip (typically a maximum of two IXF1002 ports).

Thread sequencing

The eight threads in the IXF1002 ingress process execute in a pre-defined round-robin order that alternates Receive requests between each microengine. The algorithm is similar to that described in the Intel Gigabit Example Design. A semaphore throttles the number of pending Receive requests from a microengine at any time to only a single request. The Receive Request is alternated between microengines to allow for greater parallelism. Enforcing a pre-defined ordering of thread execution also simplifies re-assembly of a packet received by multiple threads.

Initialization

An IXF1002 ingress initialization macro works in conjunction with AAPI function calls to configure each IXF1002 ingress process at run time. The initialization macro must be called prior to executing the ingress function.

You can configure the following thread parameters:

Parameter	Description
MacPort	Specifies the MAC number and port number of the IXF1002 port.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameter	Description
StackID	Specifies the SRAM push/pop stack used to maintain a free list of available buffer descriptors.
StartOffset	Specifies the number of quad words to leave blank at the beginning of the SDRAM buffer (to allow for header or other user data insertion at the beginning of the packet data).

These parameters are automatically assigned to the thread, based on information maintained by the ENP SDK microcode resource manager:

Parameter	Description
Rfifo Element	Statically assigned RFIFO element utilized by each thread.
NextThread	The next thread to execute in the round-robin thread execution sequence.
FPSeqNo	The Fast Port number of the IXF1002 port. This is defined by the hardware platform.
FirstThread	The first thread to execute in the round-robin daisy chain.
StackSramBase	The base address of the contiguous region of SRAM where buffer descriptors are located. Used to calculate the SDRAM buffer address from the SRAM buffer descriptor address.
StackDramBase	The base address of the contiguous region of SDRAM where data buffers are located. Used to calculate the SDRAM buffer address from the SRAM buffer descriptor address.

You can re-configure the ingress process during run-time by re-starting the individual microengines running the ingress process.

This process diverges from the IXF440 method where you can re-configure a thread without a microengine re-start by re-sending an initialization message. Using a re-start process for re-configuration improves performance by eliminating the processing steps required to check for message updates.

To change a port configuration:

1. Stop the port with IXF1002StopPort().
2. Unhook the port with IXF1002UnhookPort().
3. Re-Hook the port by calling IXF1002HookPort() with the new parameters.
4. Re-initialize any application microcode within the ingress threads (this would be a user defined process).
5. Re-start the port.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Alternatively, you can:

1. Stop all ports
2. Reset, reload, and restart all of the microengines
3. Re-initialize the ENP SDK microcode resource manager

Re-run the function to configure and start the microcode drivers

Receive requests

The IXF1002 ingress process utilizes blind speculative Receive requests on its assigned fast port. These are fast port Receive requests issued without any knowledge of the port's ready status. The FBI unit checks the fast ready flag as part of the Receive request and, if invalid, cancels the Receive request. The ingress macro returns this result in the return status parameter. One effect of speculative Receive requests is that an inactive port may actually cause a greater load on the FBI unit than an active one because less loop delay exists between cancelled requests than between non-cancelled requests. The ingress macro issues blind speculative Receive requests just as fast as the loop delay and thread chaining allows. For this reason, it *may* be beneficial to add some delay to the path of a cancelled speculative Receive request in the application microcode program that calls the ingress macro.



If no incoming data exists, the IFX1002 ingress macro issues blind speculative receive requests indefinitely.

Fast port thread-to-thread communication

When receiving an `Mpkt`, a thread must determine where to store the data received if it is not the start of a new packet. This information is provided in a buffer descriptor address and the current `Mpkt` count passed from thread to thread in the daisy chain via a set of mailbox locations in Scratch memory. The SOP Sequence Number is also passed along so that the EOP thread may return this parameter in the return status of the ingress macro.

An SOP thread does not need to check the mailbox before starting its processing. It first passes the information to the downstream thread and then waits to consume the information from the upstream thread. All other situations require the upstream information to be consumed before passing on the information to the downstream thread. As a consequence of this optimization, it is possible for an EOP thread to finish its ingress processing and start packet processing before the SOP thread completes its ingress processing. This can result in a situation where the header information may not physically be in SDRAM when the EOP process begins looking at it.

To address this, ensure that the SOP ingress process reaches a known state before an EOP process operates on the packet.

The IXF1002IngressEgress and ENP2506FwdDemo microcode projects show how this may be handled.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Buffer descriptors

Packet data is stored in SDRAM buffers allocated by the StrongARM core prior to the start of any packet reception.

Buffer descriptors, used to define SDRAM buffer locations, are stored in SRAM and use the push/pop features of the IXP1200 SRAM unit to maintain a free list of available buffers for storing packet data.

The IXF1002 ingress and egress macros assume that SDRAM buffers are each 2048 bytes and that both the SDRAM buffers and SRAM buffer descriptors are contiguous in memory.

Based on the above criteria, one can calculate the SDRAM buffer address from the SRAM buffer descriptor address instead of performing a slower secondary read from a buffer descriptor data structure. This improves overall performance.

Return conditions

The ingress macro exits with one of these exception codes:

Code	Description
IXF1002_RX_NO_EXCEPTION	The macro executed normally.
IXF1002_RX_CANCELLED	The FBI cancelled the Receive request.
IXF1002_RX_FAIL	A receive fail was encountered (the packet is automatically discarded by the macro).
IXF1002_RX_OUT_OF_BUFFERS	There were no free buffer descriptors available and the *Mpkt* was left in the RFIFO.

The state of the SOP and EOP flags are also returned to the calling program so that early header processing may be initiated before the entire packet is received.

See section 5.5.2 for additional details on the macro return parameters.

IXF1002 egress module

Overview

The IXF1002 egress module is a stand-alone microcode source file that handles transmission of packets to an IXF1002 IX bus device. The microcode source file may load on any microengine and multiple modules may load up to the limit of the IXP1200 hardware (typically a maximum of two IXF1002 ports).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Initialization

The IXF1002 egress module works in conjunction with AAPI function calls to configure each IXF1002 egress process at microengines start-up. The following thread parameters are configurable by the user:

Parameter	Description
MacPort	Specifies the MAC number and port number of the IXF1002 port.
StackID	Specifies the SRAM push/pop stack to return freed buffer descriptors.

The parameters below are assigned automatically to the thread, based on information maintained by the ENP SDK microcode resource manager:

Parameter	Description
`StackSramBase`	The base address of the contiguous region of SRAM where buffer descriptors are located. Used to calculate the SDRAM buffer address from the SRAM buffer descriptor address.
`StackDramBase`	The base address of the contiguous region of SDRAM where data buffers are located. Used to calculate the SDRAM buffer address from the SRAM buffer descriptor address.

You can re-configure the egress process during run-time by re-starting the individual microengines running the egress process. This process diverges from the IXF440 method where a thread could be re-configured without a microengine re-start by re-sending an initialization message. Using a re-start process for re-configuration improves performance by eliminating the processing steps required to check for message updates.

To change a port configuration:

1. Stop the port with IXF1002StopPort().
2. Unhook the port with IXF1002UnhookPort().
3. Re-Hook the port by calling IXF1002HookPort() with the new parameters.
4. Re-initialize any application microcode within the ingress threads (this would be a user defined process).
5. Re-start the port.

Alternatively, you can:

1. Stop all ports
2. Reset, reload, and restart all of the microengines
3. Re-initialize the ENP SDK microcode resource manager
4. Re-run the function to configure and start the microcode drivers

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Transmit queues

The queues associated with each IXF1002 egress are defined at compile time in the microcode source files. An AAPI function call allows these definitions to be incorporated into the microcode resource manager for access to the queues. The following parameters define the IXF1002 transmit queues:

Parameter	Description
QscratchBase	Specifies the Scratch base address of the enpFqueue associated with the egress. (See enpFqueue.uc for additional details.)
QSramBase	Specifies the SRAM base address of the enpFqueue associated with the egress. (See enpFqueue.uc for additional details.)

Each IXF1002 egress process is associated with a single transmit queue that allows multiple sources to insert packets. Only the associated egress process can remove packets from the queues. The IXF1002 egress process polls the transmit queue to determine when an element is available for transmission. If the port is idle for a period of time, the polling rate decreases to avoid excess load on the memory subsystems. Transmit queues may be accessed from the StrongARM core via IXF1002WritePacket AAPI functions.

For more information on enpFqueue, see *enpFqueue* on page 260.

Thread sequencing

The four threads in the IXF1002 egress process execute in a round-robin order to transmit packets on the IX bus. Context 0 initially checks the transmit queue, dequeues a packet, then sends the first *Mpkt* of data over the IX bus and passes on the packet information to the next thread in the daisy chain (Context 1). The next thread reads the information passed on from the upstream context and either continues the already-started packet transmission or checks the queue for a new packet to transmit.

By sequencing threads in a defined order, a context always knows which thread accesses the transmit data next, and to pass information to a “downstream” thread, which lets that thread start before the current thread completes its processing.

Global GPR mailboxes

Information is communicated from one IXF1002 egress thread to another via a set of global GPRs within the microengine running the IXF1002 egress module. Keeping these “mailbox” registers inside the microengine significantly increases processing speed.

TFIFO element control

The ENP SDK microcode drivers all dynamically allocate TFIFO elements. Dynamic allocation allows adding or removing egress devices in a project without modifying source code, and allows slow devices such as the TDM2IX to interleave with faster Ethernet devices without permanently consuming TFIFO elements.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A data structure is maintained in SRAM to allow multiple processes to atomatically capture to the next available TFIFO element. The current state of the XMIT_PTR prevents a wrap condition in which the next element exceeds 16 elements beyond the currently transmitting element.

Packet counters

The egress process maintains two counters that track of the number of transmitted `Mpkts` and Ethernet packets. These counters, accessible via AAPI function calls, roll over at 32 bits.

Buffer descriptors

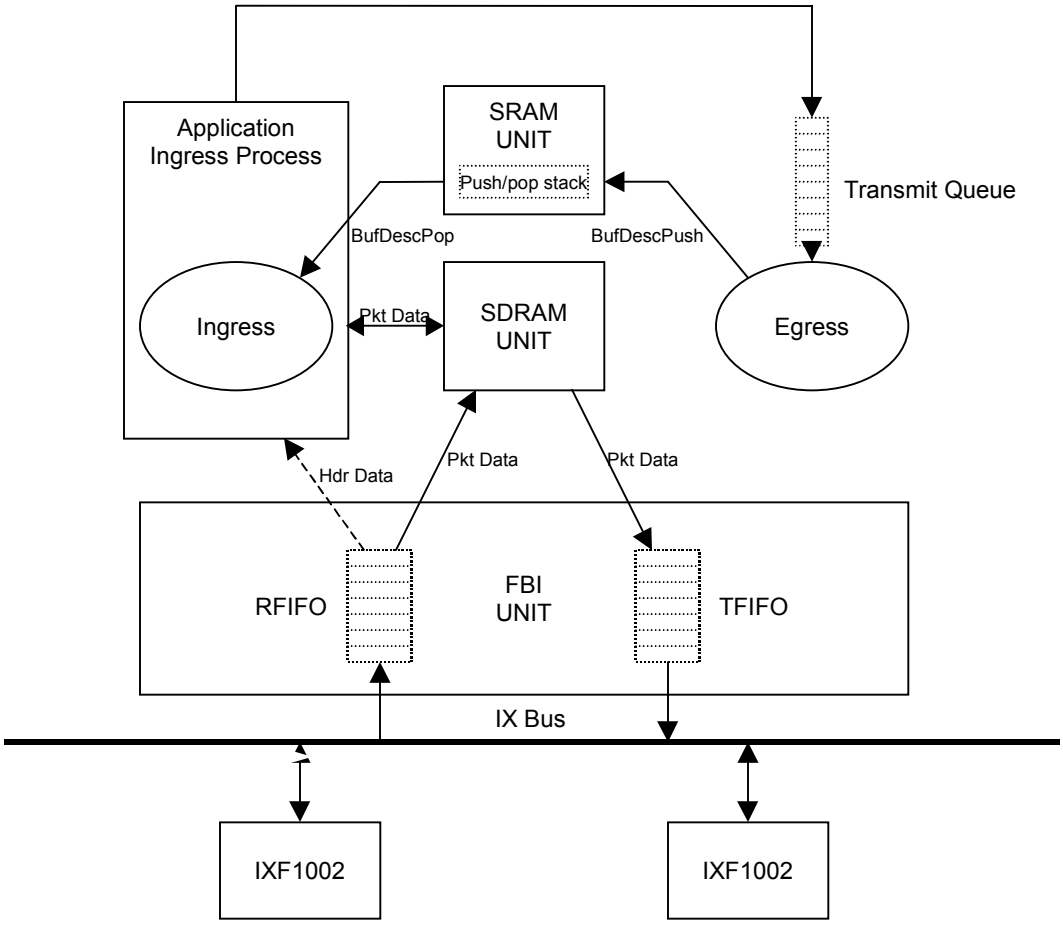
Packet data inserted into a transmit queue must be stored in SDRAM data buffers associated with SRAM buffer descriptors. The buffer descriptors are automatically returned to the specified SRAM push/pop stack after the packet data is transmitted to the MACs.

For more information about buffer descriptors, see *Buffer descriptors* on page 259.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Data flow

Figure 14-2. IFX1002 Ethernet packet data flow



Packet reception is initiated by speculative Receive request issued to the FBI unit by the ingress process. When a non-cancelled request is returned by the FBI, the ingress process determines if it is the start of a new packet and, if so, automatically pops a buffer descriptor from the specified push/pop queue. Packet data then moves from the RFIFO into the SDRAM buffer associated with the buffer descriptor. The application thread calling the ingress macro may also access the RFIFO directly to process header data early. Alternatively, the application thread accesses and modifies header data from the SDRAM unit after the packet is completely received.

Once a packet is processed by the application calling the ingress macro, the buffer descriptor associated with the packet is typically inserted into the transmit queue of an egress process (shown above) or sent to the core for further work (not shown). IXF1002 egress processes remove buffer descriptors from a transmit queue, move the associated packet data from SDRAM to the TFIFO, and initiate IX bus transfers of the TFIFO data.

When all data is sent from a packet, the egress process returns the buffer descriptor back onto a specified push/pop stack to make it available for re-use.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Ingress macro definitions

IXF1002_Ingress_Init

IXF1002_Ingress_Init initializes thread variables and data structures required by the IFX1002 ingress module. It works in conjunction with IXF1002HookPort to parameterize a thread for receiving data from a specified MacPort.

Syntax

```
#macro IXF1002_Ingress_Init(
    ThreadIDIn,          \
    RecReqBusyIn,        \
    Reg0Out,             \
    RcvReqOut,           \
    StackSramBaseOut,    \
    StackDramBaseOut
)
```

Parameters (input)

ThreadIDIn

The thread ID of the currently executing context.

RecReqBusyIn

A microengine global GPR used as a semaphore to limit the number of Receive request pending from a microengine to only one.

Parameters (output)

Reg0Out A configuration registers packed with the following fields:

[31:24] *FPSeqNo*: Fast port sequence number of the IXF1002 port in use.

[23:16] *StartOff*: AAPI configuration parameter that indicates the amount of free space to reserve in the SDRAM buffer before the start of the Rx packet data.

[15:08] *NextThrd*: The next thread to issue a Receive request in the round-robin execution chain.

[07:00] *StackID*: Stack ID of the SRAM push/pop stack to use for obtaining free buffer descriptors.

RcvReqOut A pre-formatted Receive request formatted for the current context.

StackSramBaseOut

32-bit base address of the contiguous region of buffer descriptors in SRAM.

StackDramBaseOut

64-bit base address of the contiguous region of 2K buffers in SDRAM.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002_Ingress

This macro receives the next Mpacket of data from an assigned IXF1002 IX bus device.

Syntax

```
#macro IXF1002_Ingress(
    ThreadIDIn,      \
    RecReqBusyIn,    \
    RcvReqIn,        \
    CfgReg0In,       \
    StackSramBaseIn, \
    StackDramBaseIn, \
    EpktPtrOut,      \
    EpktDramBaseOut, \
    EBufStatOut,     \
    RxStatusOut
)
```

Parameters (input)

ThreadIDIn

Thread ID of the currently executing context.

RecReqBusyIn

Global GPR used as a semaphore to determine if a thread within a microengine is currently executing a Receive request. I

To prevent saturating the FBI unit on the IXP1200, a microengine is limited to only one pending Receive request at a time.

RcvReqIn A pre-formatted Receive request for the currently executing context.

CfgReg0In A packed set of configuration parameters.

[31:24] *FPSeqNo*: Fast port sequence number of the IXF1002 port in use.

[23:16] *StartOff*: An AAPI Configuration parameter that indicates the amount of free space to reserve in the SDRAM buffer before the start of the Rx packet data.

[15:08] *NextThrd*: The next thread to issue a Receive request in the round robin execution chain.

[07:00] *StackID*: Stack ID of the SRAM push/pop stack to use for obtaining free buffer descriptors.

StackSramBaseIn

The 32-bit base address of the contiguous region of buffer descriptors in SRAM.

StackDramBaseIn

The 64-bit base address of the contiguous region of 2K buffers in SDRAM.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parameters (output)

BufDescPtr

A pointer to the buffer descriptor in SRAM (microengine addressing).

BufDescDramBase

A pointer to the SDRAM data buffer associated with the buffer descriptor (microengine addressing).

BufDescStat

Status information associated with the packet received in the data buffer. See field definitions below:

```
// 31    28 27      24 23      6 15  12 11    8 7     4 3     0
//+-------------------+---------+-------------+-------+------+
//|StackID | VldByts  | MacPort |PktSize       |StartOffset   |
//+-------------------+---------+-------------+-------+------+
```

StackID Push/pop SRAM stack that the buffer descriptor came from.

VldByts Valid bytes in the last quadword of the packet data.

MacPort MacPort that the packet was received on.

PktSize Size in quadwords of the packet.

StartOffset
Offset in quadwords of the first byte of packet data in the buffer.

RxStatus Return status of the macro. See field definitions below.

```
// 31                     16 15    12 11   8 7   4 3         0
//+----------------+--------+-------------+-----+----------+
//|MacStatus (optional)      |RSVD   |SopSeq | | |E|S|Excptn |
//+----------------+--------+-------------+-----+----------+
```

MacStatus A copy of the status word optionally returned by the IXF1002.

SopSeq SOP Sequence Number of the first Mpacket received in the current packet.

E A flag that indicates EOP if was signaled on the Receive request.

S A flag that indicates SOP if was signaled on the Receive request.

Excptn A 4-bit exception code for the macro. These exceptions include:

IXF1002_RX_NO_EXCEPTION
IXF1002_RX_CANCELLED
IXF1002_RX_FAIL
IXF1002_RX_OUT_OF_BUFFERS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Data structures

The speed requirements of gigabit ports preclude the use of IXF440 style linked-list transmit queues and DRAM buffer pointers stored in buffer descriptors for memory management. To address this issue, two new objects are introduced for the gigabit threads:

- FastEpkt buffer descriptors
- enpFqueue

Ssubsequent sections describe each object.

Buffer descriptors

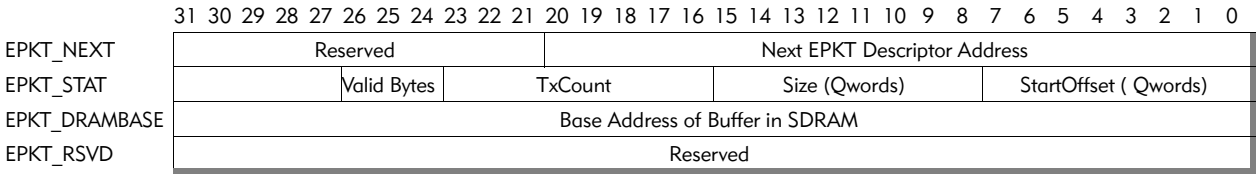
Buffer descriptors specify a data buffer location in SDRAM and are maintained on SRAM push/pop stacks to allow dynamic allocation of the data buffers by multiple processes. A process allocates a buffer descriptor by popping a buffer descriptor pointer from an SRAM push/pop stack and de-allocates it by pushing the pointer back onto the stack. Ownership of the descriptor may be passed from one process to another by passing a pointer variable.

Buffer descriptors may occupy multiple words in SRAM. The definition of these words varies depending on the source or target of the buffer descriptor. Currently there are two variations of the buffer descriptor fields:

- Epkt
- FastEpkt

Epkt buffer descriptors

Epkt buffer descriptors are data structures located in SRAM that describe and manage the Ethernet data packets for IXF440 ingress and egress modules. Each Epkt buffer descriptor occupies four words. There are a total of six fields currently used within the four words:



Next Epkt Descriptor Address

Maintains linked list structures of Epkt descriptors. An SRAM push/pop stack uses this field to point to the next item in the stack. IFX440 transmit queues use the field to point to the next item in the queue.

Start Offset

Informs the transmit thread where the first word to transmit is located within the 2K DRAM buffer. Informs the receive thread where to place the first word received within the 2K DRAM buffer. Deletes or inserts information into the packet without physically moving data in DRAM. For insertion, the receive thread indicates a start offset of some number

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

of quad words. After insertion, the start offset is modified to indicate the new start offset. All packet data must be contiguous within DRAM.

Size The total number of full and partial quad words of packet data. Only the last quad word of data can be partial.

TxCount Not used at this time.

Valid Bytes
The number of valid bytes in the last quad word of data starting at base 0 (i.e a value of 0 indicates 1 valid byte and a value of seven indicates all eight bytes are valid).

Base Address of Buffer in SDRAM
The base address of the contiguous 2048 byte SDRAM buffer.

FastEpkt buffer descriptors

FastEpkt buffer descriptors, used with the IXF1002 ingress and egress processes, are conceptually similar to Epkt buffer descriptors except they use the defined relationship between the SRAM address of the buffer descriptor and the SDRAM address of the physical buffer. This allows a fast ALU calculation to determine the DRAM address instead of a slower memory access.

A FastEpkt buffer descriptor does not need to contain any information within the descriptor fields. However, for backwards compatibility to Epkt buffer descriptors, the locations should be reserved and initialized when the stack is created.

Two GPRs that contain the stack SRAM and SDRAM bases must be reserved in any thread that needs to perform the FastEpkt descriptor address to SDRAM buffer address conversion.

ENPCreateEpkt creates a stack compatible with the IXF1002 microcode drivers.

When working with FastEpkt buffer descriptors, the IXF1002 ingress and egress functions do not look inside the descriptor for the status information associated with the packet received in the buffer descriptor. This information is instead incorporated into the enpFqueue element data structure which allows the transmit process to retrieve the buffer descriptor and the packet status information with a single memory reference.

The BufDescStat data word returned from the IXF1002 ingress process is directly compatible with the enpFqueue `PktStat` field definitions. It is also compatible with the EpktStat field definitions inside the Epkt buffer descriptor.

enpFqueue

An enpFqueue is a ring structure with the enqueue and dequeue pointers located in different memory spaces.

A global enqueue index is maintained in SRAM memory and is shared by multiple processes. Locating this index in SRAM allows use of the read-lock instructions for atomic access on insertions.

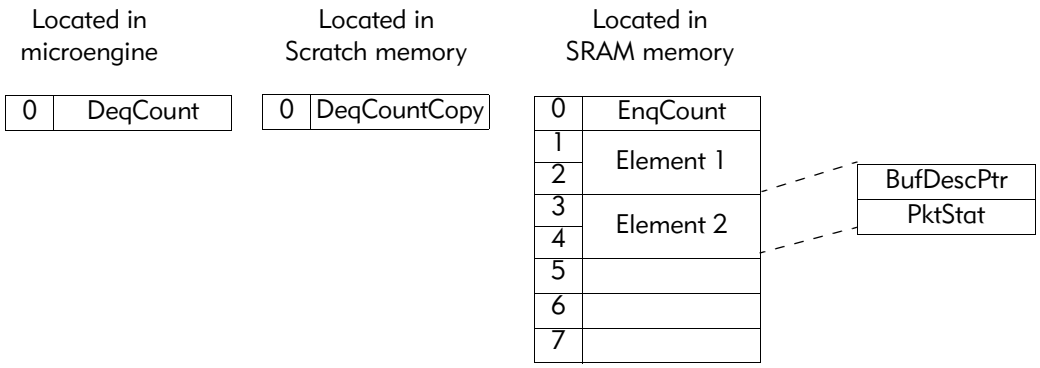
The dequeue index is maintained inside the microengine or process associated with the queue. Maintaining the dequeue index inside a microengine speeds up the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

dequeue process by eliminating a memory reference to determine the location of the next element to remove and by allowing the insertion and removal processes to occur in parallel.

Packets are inserted into an enpFqueue with the enpFqueuInsert macro and removed with the enpFqueueWaitRemove macro. The source for these macros is provided in the enpFqueue.uc file in the ENP SDK release 2.0 or higher.

Figure 14-3. enpFqueue data structure



To determine whether a ring buffer is full, the insertion process must have visibility of both the enqueue and dequeue index. For this reason, a copy of the dequeue index is maintined in Scratch memory. Scratch memory takes advantage of the native increment operations which allow a thread to increment a scratch memory location in a single opcode without context swapping. An issue arises with regard to stale data in the copy of the dequeue index, however this will not cause any problem since the stale data could only be used to indicate a false queue full indication. (Effectively this might cause the queue to act as if there were 1 or 2 element less than the total available in the ring.)

The enqueue index may also be stale from the perspective of the removal process because a locked access to the enqueue index is not required for removal operations. The risk here is that the removal process will read the enqueue index while it being updated by an enqueue process. Again, this is not a problem because the stale data could only be used to indicate a false empty condition. The dequeue process will continue to poll the enqueue index without affect until the updated value is eventually received.

The variables in an enpFqueue are defined below.

DeqCount A global GPR maintained inside the microengine or process associated with the queue. It is a pointer to the next element to remove from the queue.

DeqCountCopy
Copy of the *DeqCount* in Scratch memory and available to the processes inserting elements into the queue. It determines if the queue is full.

EnqCount Pointer to the location to insert the next element into the queue. The insertion process must request a lock on this location before proceeding.

Element A queue element that consists of a buffer descriptor pointer and a *PktStat* variable.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

`BufDescPtr`
A pointer to an SRAM buffer descriptor.

`PktStat` A packed data word containing the following fields:

[31:28] SRAM push/pop stack that the buffer descriptor originated from.

[27:24] The number of valid bytes contained in the last quad word of packet data. A value of 0 indicates 1 valid byte and a value of 7 indicates 8 valid bytes.

[23:16] MacPort on which the packet was received.

[15:08] Size of the packet in quad words (including last partial quad word).

[07:00] Offset in quad words of first byte of packet data in the data buffer.

IFX1002 µPI

The microcode programming interface to the IFX1002 ingress and egress modules is simple. On the receive side, a user microcode source file calls the IXF1002_Ingress() macro to receive the next Mpkt of data from the assigned port and then branches to a post processing function based on the return status of the macro. On the transmit side, a source process (that has prepared a packet to send) inserts the buffer descriptor and packet status word into the transmit queue of an egress process and the packet is automatically transmitted and the buffer descriptor returned to the specified SRAM push/pop stack.

Example applications are included in the ENP SDK to show how the microcode modules are incorporated into a functional application.

IXF1002 ingress µPI

For details on the IXF1002 ingress uPI, see section 5.5 which details the ingress macro functions.

Applications that call the ingress process must implement a function to move packets up to the StrongARM, if that is a requirement of the design.

IFX1002 egress µPI

The IFX1002 transmit queue is accessed from the microengines via the enpFqueueInsert macro included in the enpFqueue.uc source file. The enpFqueueInsert macro uses SRAM locking to ensure atomic accesses. Transmit queues may also be accessed by the StrongARM via IXF1002WritePacket.

For additional details on the IXF1002 egress uPI, see section 5.6.2 which describes the enpFqueue data structure and section 5.3 describing the egress functionality.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Using the IFX1002 microcode drivers

Ingress cycle time

To prevent buffer overrun inside the IXF1002 MACs, the process that calls the ingress macro must ensure that the average cycle time of the ingress and packet processing loop does not exceed the average minimum `Mpkt` arrival rate of the packet data.

RadiSys recommends that you simulate the final microcode design on the IXP1200 Developers Workbench to determine if line rate processing can be achieved.

Start-up and initialization

To start the IFX1002 transmit and receive modules:

1. Load and start the microcode modules you want to use.
2. Initialize the ENP SDK microcode run-time environment by calling ENP2506_InitApp.

   Complete this before allowing any microcode to begin normal execution. Note that all ENP SDK supplied microcode modules do not begin normal execution until signaled to do so via AAPI function calls.

3. Create a free list of Epkt buffers descriptors for each port to use.
4. Open a handle to each IFX440 port you plan to use.
5. Initialize any desired hardware settings for each IFX440 port.
6. Hook each port to the desired receive thread, transmit thread, Epkt stack, and receive signal thread.
7. Enable receive and/or transmit operation for each port.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002 interrupt routines

Include File Ixf1002.h

Library Ixf1002.a

Each IXF1002 MAC port can generate various interrupts types. For more information, see the IXF1002 datasheet.

When an IXF1002 interrupt occurs, the IXF1002UserHandler is called and a structure of type InterruptStatus is passed to that function:

Table 14-1. InterruptStatus structure

Field	Value
InterruptStatus.device	INTERRUPT_FROM_IXF1002
InterruptStatus.status	IXF1002_INT_STT1|IXF1002_INT_STT0 (last 16 bits port1 and first 16 bits port0)
InterruptStatus.portStatus	
[0]	IXF1002_TXRX_STT0|IXF1002_INT_STT0
[1]	IXF1002_TXOV_STT0
[2]	IXF1002_TXOV_STT0
[3]	IXF1002_TXRX_STT1|IXF1002_INT_STT1
[4]	IXF1002_TXOV_STT1
[5]	IXF1002_TXOV_STT1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function list

Use this table to identify the IX1002 functions you want to use. Use the function description later in this chapter to obtain detailed information, including syntax and parameter or element values.

Table 14-2. IXF1002 microcode functions

Function	Description	Page
IXF1002ClearNetStats	Clears the network statistics counters from an IXF1002 port.	267
IXF1002ClosePort	Closes an IXF1002 port.	268
IXF1002DefineTxQueue	Defines the location of the enpFqueue associated with an IXF1002 transmit process to the microcode resource manager.	269
IXF1002DisableAllInterrupt	Disables all IXF1002 interrupt events.	270
IXF1002DisableInterrupt	Disables the specified IXF1002 interrupt event.	271
IXF1002EnableAllInterrupt	Enables all IXF1002 interrupt events.	272
IXF1002EnableInterrupt	Enables the specified IXF1002 interrupt event.	273
IXF1002GetRxTxMode	Gets the current receive and transmit operation status of an IXF1002 port.	274
IXF1002GetTxEpktCount	Gets the current Epkt transmit count for an IXF1002 port.	275
IXF1002GetTxMpktCount	Gets the current Mpkt transmit count for an IXF1002 port.	276
IXF1002HookPort	Connects an IXF1002 MacPort to a receive and/or transmit process for IX bus operation.	277
IXF1002OpenPort	Opens an IXF1002 port and returns a handle to the port.	278
IXF1002QueryNetStats	Reads the resetting or non-resetting network statistics from an IXF1002 port.	279
IXF1002ReadNetStatReg	Reads a network statistic register from an IXF1002 port.	280
IXF1002ReadPacket	This function is not implemented.	281
IXF1002ReadReg	Reads an IXF1002 register.	282
IXF1002ResetPort	Resets an IXF1002 port.	283
IXF1002SetGMIIMode	Sets the port to work in GMII mode.	284
IXF1002SetGPCSMode	Sets the port to work in GPCS mode.	285
IXF1002SetInterruptHandler	Installs an interrupt handler for IXF1002 events.	286
IXF1002StartPort	Enables the receive and/or transmit operation of an IXF1002 port.	287

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 14-2. IXF1002 microcode functions

Function	Description	Page
IXF1002StopPort	Disables receive and transmit operation of an IXF1002 port.	288
IXF1002UnhookPort	Disconnects the MacPort from the specified Rx and/or Tx threads	289
IXF1002WritePacket	Writes a packet into an IXF1002 transmit queue.	290
IXF1002WriteReg	Writes an IXF1002 register.	291

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002 AAPI

IXF1002ClearNetStats

Clears the network statistics counters from an IXF1002 port.

Syntax

```
t_ENPStatus IXF1002ClearNetStats(
    t_IXF1002PortHandle IXF1002PortHandle
)
```

Parameters

IXF1002PortHandle

Specifies an IXF1002 port.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002ClosePort

Closes an IXF1002 port.

Syntax

```
t_ENPStatus IXF1002ClosePort (
   t_ENPPortHandle  PortHandle
);
```

Parameters

PortHandle

Specifies an IXF1002 port.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

The port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002DefineTxQueue

Defines the location of the enpFqueue associated with an IXF1002 transmit process to the microcode resource manager.

Syntax

```
t_ENPStatus IXF1002DefineTxQueue(
    Bit8            MacNum,
    Bit8            PortNum,
    Bit32           ScratchBase,
    Bit32           SramBase,
    Bit32           MaxElements
);
```

Parameters

MacNum MAC number of the IXF1002 device. Valid values include 0 to 7.

PortNum Port number on the Mac device. Valid values include 0 to 1 (only two ports exist on the IXF1002).

ScratchBase
Specifies the scratch block of the IXF1002 Tx queue.

SramBase Specifies the SRAM block of the IXF1002 Tx queue.

MaxElements
Specifies queue size.

Return values

ENP_SUCCESS
ENP_INVALID_MACPORT
ENP_INVALID_PORT_TYPE

Comments

The locations of the enpFqueues are also defined in the microcode source files.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002DisableAllInterrupt

Disables all IXF1002 interrupt events.

Syntax

```
IXF1002DisableAllInterrupt(
   t_ENPPortHandle PortHandle
);
```

Parameters

PortHandle

Handle to the IXF1002 port.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002DisableInterrupt

Disables the specified IXF1002 interrupt event.

Syntax

```
IXF1002DisableInterrupt(
   t_ENPPortHandle           PortHandle,
   t_IXF1002InterruptEvent   Event
);
```

Parameters

PortHandle

Handle to the IXF1002 port.

Event IXF1002 event to disable.

t_IXF1002InterruptEvent

IXF1002_PGRCE Page Received interrupt
IXF1002_ANERE Auto Negotiation Error
IXF1002_RXOKE Receive OK
IXF1002_PKEVE Packet Event
IXF1002_TXOKE Receive OK
IXF1002_LNCHE Link Change
IXF1002_OVFCECounter overflow
IXF1002_STOPE Port Stop

Default handler clears the interrupts and disables any further interrupts from that specific source within the device, which caused the interrupt and passes the above information to the user handler. You must re-enable any interrupt, if needed.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002EnableAllInterrupt

Enables all IXF1002 interrupt events.

Syntax

```
IXF1002EnableAllInterrupt(
   t_ENPPortHandle PortHandle
);
```

Parameters

PortHandle

Handle to the IXF1002 port.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002EnableInterrupt

Enables the specified IXF1002 interrupt event.

Syntax

```
IXF1002EnableInterrupt(
   t_ENPPortHandle           PortHandle,
   t_IXF1002InterruptEvent   Event
);
```

Parameters

PortHandle

Handle to the IXF1002 port.

Event IXF1002 event to enable.

t_IXF1002InterruptEvent

IXF1002_PGRCE	Page Received interrupt
IXF1002_ANERE	Auto Negotiation Error
IXF1002_RXOKE	Receive OK
IXF1002_PKEVE	Packet Event
IXF1002_TXOKE	Receive OK
IXF1002_LNCHE	Link Change
IXF1002_OVFCE	Counter overflow
IXF1002_STOPE	Port Stop

Default handler clears the interrupts and disables any further interrupts from that specific source within the device, which caused the interrupt and passes the above information to the user handler. You must re-enable any interrupt, if needed.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002GetRxTxMode

Gets the current receive and transmit operation status of an IXF1002 port.

Syntax

```
t_ENPStatus IXF1002GetRxTxMode(
    t_ENPPortHandle      PortHandle,
    t_IXF1002RxTxMode    *RxTxMode
);
```

Parameters

PortHandle
Specifies an IXF1002 port.

RxTxMode Pointer to location to place the current RxTx mode of an IXF1002 port. Valid values include:

IXF1002_RXONLY
IXF1002_TXONLY
IXF1002_RXANDTX
IXF1002_STOP

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port must be opened with IXF1002OpenPort().

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002GetTxEpktCount

Gets the current Epkt transmit count for an IXF1002 port.

Syntax

```
t_ENPStatus

IXF1002GetTxEpktCount (
    t_ENPPortHandle      PortHandle,
    Bit32                *DataPtr
);
```

Parameters

PortHandle
Specifies an IXF1002 port.

DataPtr Pointer to location to place the count data

Return values

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_INVALID_OBJECT

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002GetTxMpktCount

Gets the current Mpkt transmit count for an IXF1002 port.

Syntax

```
t_ENPStatus

IXF1002GetTxMpktCount(
    t_ENPPortHandle      PortHandle,
    Bit32                *DataPtr
);
```

Parameters

PortHandle
Specifies an IXF1002 port.

DataPtr Pointer to location to place the count data.

Return values

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_INVALID_OBJECT

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002HookPort

Connects an IXF1002 MacPort to a receive and/or transmit process for IX bus operation.

Specifies an SRAM push/pop stack to use for buffer allocation and de-allocation. Specifies a start offset to leave free at the beginning of the packet data buffer.

Syntax

```
t_ENPStatus IXF1002HookPort(
    t_ENPPortHandle  PortHandle,
    Bit16            RxUeng0ID,
    Bit16            RxUeng1ID,
    Bit16            TxUengID,
    t_IXPStackID     StackID,
    Bit8             StartOffset
);
```

Parameters

PortHandle
Specifies an IXF1002 port.

RxUeng0ID One of two Microengine allocated to the IXF1002 ingress process. Valid values include 0 to 5.

RxUeng1ID One of two Microengine allocated to the IXF1002 ingress process. Valid values include 0 to 5.

TxUengID Microengine allocated to the IXF1002 egress process. Valid values include 0 to 5.

StackID Specifies the SRAM push/pop stack to use. Valid values include IXP_STACK0 to IXP_STACK7

StartOffset
Number of quad words for the ingress process to leave blank in the SDRAM data buffer before inserting the packet data.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_INVALID_OBJECT
ENP_INSUFFICIENT_RESOURCES
ENP_THREAD_NOT_READY

Comments

IXF1002 ingress and/or egress Microcode must be running on the Microengines. An EpktStack must exist for use by the receiver and transmitter.

A port opened in read only or write only must have existing threads to source or sink the new/used buffer descriptors.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002OpenPort

Opens an IXF1002 port and returns a handle to the port.

Syntax

```
t_ENPStatus IXF1002OpenPort (
   Bit16            MacNum,
   Bit16            PortNum,
   t_ENPPortHandle  *PortHandle
);
```

Parameters

MacNum Mac number of the IXF1002 device. Valid values include 0 to 7.

PortNum Port number on the Mac device. Valid values include 0 to 1 (only two ports exist on the IXF1002).

PortHandle
Pointer to location where the ENPPortHandle returns.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_PORT_ALREADY_OPEN
ENP_INVALID_MACPORT
ENP_INVALID_PORT_TYPE

Comments

The ENP Microcode Resource manager must be initialized with ENPXXXX_InitApp before opening any ports.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002QueryNetStats

Reads the resetting or non-resetting network statistics from an IXF1002 port.

Syntax

```
t_ENPStatus IXF1002QueryNetStats(
    t_IXF1002PortHandle     IXF1002PortHandle,
    t_IXF1002NetStatType    NetStatType,
    t_IXF1002NetStat        *IXF1002NetStat
)
```

Parameters

IXF1002PortHandle

Specifies an IXF1002 port.

NetStatType

Specifies the resetting or non-resetting network statistic registers. alid values include:

IXF1002_NETSTAT_RESETTING,

IXF1002_NETSTAT_NON_RESETTING

IXF1002NetStat

Pointer to t_IXF1002NetStat structure where the network statistics will be placed. See IXF1002.h header file for additional details.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_INVALID_OBJECT

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002ReadNetStatReg

Reads a network statistic register from an IXF1002 port.

Syntax

```
t_ENPStatus IXF1002ReadNetStatReg(
    t_IXF1002PortHandle     IXF1002PortHandle,
    t_IXF1002NetStatType    NetStatType,
    Bit16                   RegOffset,
    Bit16                   *RegData
)
```

Parameters

IXF1002PortHandle
Specifies an IXF1002 port.

RegOffset Specifies an IXF1002 network statistic register (see IXF1002.h).

NetStatType
Specifies the resetting or non-resetting network statistic registers. Valid values include:

IXF1002_NETSTAT_RESETTING

IXF1002_NETSTAT_NON_RESETTING

RegData Pointer to location to place the data read.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_INVALID_OBJECT

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002ReadPacket

This function is not implemented.

You must set up a method to handle this function as part of the application Microcode that calls the ingress process.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002ReadReg

Reads an IXF1002 register.

Syntax

```
t_ENPStatus IXF1002ReadReg(
    t_ENPPortHandle  PortHandle,
    Bit16            IXF1002RegOffset,
    Bit16            *ReadData
)
```

Parameters

PortHandle
Specifies an IXF1002 port.

IXF1002RegOffset
Specifies an IXF1002 register. See IXF1002.h header file for valid values.

ReadData Pointer to location to place read data

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002ResetPort

Resets an IXF1002 port.

Syntax

```
t_ENPStatus IXF1002ResetPort(
    t_IXF1002PortHandle IXF1002PortHandle
)
```

Parameters

IXF1002PortHandle
Specifies an IXF1002 port.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002SetGMIIMode

Sets the port to work in GMII mode.

Syntax

```
t_ENPStatus

IXF1002SetGMIIMode(
    t_IXF1002PortHandle IXF1002PortHandle
)
```

Parameters

IXF1002PortHandle
Specifies an IXF1002 port.

Return values

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open. The port must be in stopped prior to calling this function.

Both ports must be set to the same mode.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002SetGPCSMode

Sets the port to work in GPCS mode.

Syntax

```
t_ENPStatus IXF1002SetGPCSMode(
    t_IXF1002PortHandle IXF1002PortHandle
)
```

Parameters

IXF1002PortHandle
Specifies an IXF1002 port.

Return values

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open. The port must be in stopped prior to calling this function.

Both ports must be set to the same mode.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002SetInterruptHandler

Installs an interrupt handler for IXF1002 events.

Syntax

```
IXF1002SetInterruptHandler(
   void (*IXF1002UserHandler)(InterruptStatus)
);
```

Parameters

Pointer to the user-defined interrupt-handling function.

Return value

ENP_SUCCESS
ENP_INT_HANDLER_INSTALL_FAILURE

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002StartPort

Enables the receive and/or transmit operation of an IXF1002 port.

Syntax

```
t_ENPStatus IXF1002StartPort(
   t_ENPPortHandle      PortHandle,
   t_IXF1002RxTxMode    RxTxMode
);
```

Parameters

PortHandle
Specifies an IXF1002 port.

RxTxMode Specifies Rx enable, Tx enable, or both. Valid values include:

IXF1002_RXONLY
IXF1002_TXONLY
IXF1002_RXANDTX

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_INVALID_OBJECT

Comments

The port must be opened with IXF1002OpenPort.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002StopPort

Disables receive and transmit operation of an IXF1002 port.

Syntax

```
t_ENPStatus IXF1002StopPort(
    t_ENPPortHandle PortHandle
);
```

Parameters

PortHandle

Specifies an IXF1002 port.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port must be opened with IXF1002OpenPort().

The port completes transmission and reception of current packets and then enters stop state. Stop state entry may generate an interrupt, if enabled, and is reported by the stop bit in the IXF1002 port's interrupt status register.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002UnhookPort

Disconnects the MacPort from the specified Rx and/or Tx threads

Syntax

```
IXF1002UnhookPort(
    t_ENPPortHandle  PortHandle
);
```

Parameters

PortHandle

Specifies the IXF1002 port to unhook.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_THREAD_NOT_READY

Comments

Port was hooked.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002WritePacket

Writes a packet into an IXF1002 transmit queue.

Syntax

```
t_ENPStatus IXF1002WritePacket(
    t_ENPPortHandle         PortHandle,
    Bit32                   FastEpktDescPtr,
    Bit32                   FastEpktStat
);
```

Parameters

PortHandle
Specifies an IXF1002 port.

FastEpktDescPtr
Pointer to a FastEpkt buffer descriptor to write to the transmit queue.

FastEpktStat
Status information associated with the packet received in the data buffer.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_INVALID_SRAM_ADDRESS
ENP_TXQUEUE_NOT_DEFINED
ENP_QUEUE_FULL

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF1002WriteReg

Writes an IXF1002 register.

Syntax

```
t_ENPStatus IXF1002WriteReg(
    t_ENPPortHandle         PortHandle,
    Bit16                   IXF1002RegOffset,
    Bit16                   WriteValue
)
```

Parameters

PortHandle
Specifies an IXF1002 port.

IXF1002RegOffset
Specifies an IXF1002 register. See IXF1002.h header file for valid values.

WriteData 16-bit data to write

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

15 TDM2IX microcode library

This describes the TDM2IX microcode library.

You can find a function in this chapter either by locating it alphabetically within its group, or by using Table 15-2, which groups like functions together.

Overview

Microcode libraries

The programming interface consists of a library of C functions that run on the StrongARM core (AAPI) and a defined interface for microcode-to-microcode communications (µPI). The AAPI and microcode modules work together with a microcode resource manager to simplify application development using the TDM2IX device on ENP platforms and future generations of embedded network processor boards.

Programming model

The ENP board's microcode software provides transmit and receive functionality for moving packets to and from the IXF440 and TDM2IX Bus devices on an ENP board. Microcode binary files transmit and receive on the IXP1200 microengines and provide the necessary infrastructure. A library of C function calls allows applications to configure the microcode programs for specific usage.

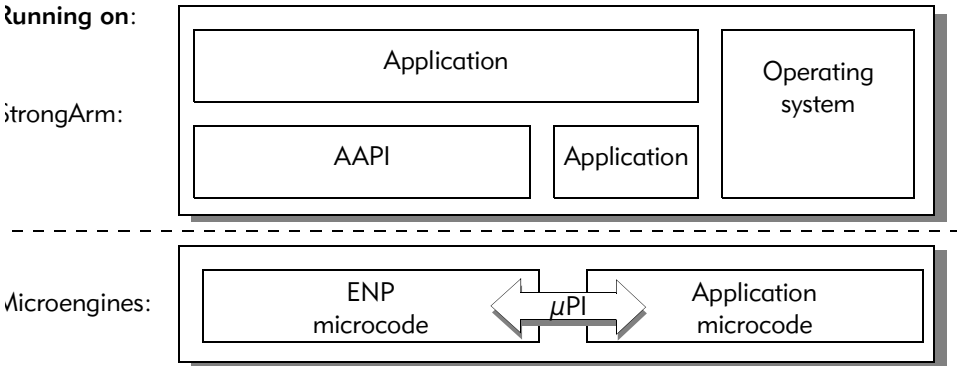


Figure 15-1. ENP board: microcode programming model

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IX Receive module

The TDM-to-IX Receive module provides receive functionality for the TDM-to-IX Bus device on the ENP board. A library of C function calls allows applications to configure the microcode programs for specific application. Developer-created buffer packing and unpacking thread(s) format and packetize TDM data. The buffer processing threads may reside on the TDM receive/transmit microengine and/or other engine(s), depending on the application. Sample microcode is supplied as an example for developers creating a TDM post-process task.

TDM to IX receive module

Receive data buffer

The TDM receive data is stored in one SDRAM buffer statically allocated by the StrongARM core at initialization, prior to the start of the microengines.



- Post processing buffer ownership is limited to a maximum 850 microseconds.
- You must ensure that the microcode receive process does not overwrite unprocessed TDM data.

The TDM buffer, allocated 2048 bytes by default, corresponds to the maximum hardware capacity of 256 receive channels. Each channel contains eight bytes, representing eight 125 microsecond frames of TDM data. The TDM data is referenced little endian, with the least significant byte being the first byte in the stream for each timeslot. The next table shows the format of buffered TDM data.

Timeslot 0	Timeslot 1	Timeslot 2	Timeslot ...	Timeslot 255
Bytes 0–7	Bytes 8–15	Bytes 16–23	Bytes ...	Bytes 2031–2047

Receive buffer queue

The receive queue is a dedicated data word in SRAM. A buffer pointer is placed into this location after a packet is received. The TDM post process writes a NULL value to this location when processing is complete. This signals the receive process that post-processing is complete. The location of the RxTDM2IXBase (receive queue) data word is reserved at compile time, and is accessed through AAPI C function calls.

Timing considerations

TDM data is periodic. One 2048 byte block is received every 1 (one) millisecond. This timing requirement is independent of the number of channels actually received. The number of channels read from TDM may be reduced in increments of 64 bytes / 8 channels to accommodate other processing and bandwidth demands when 256 channels are not required. The TDM receive module consumes approximately 3% of the IX bus bandwidth of the ENP board.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Error conditions

The TDM receiver reports two error conditions: buffer overwrite and receiver overrun. The next table details those error conditions.

TDM2IX receiver error conditions

Rx_Overwrite	The thread(s) processing the TDM data did not write a NULL to the buffer pointer queue before reception of the next one millisecond sample. This condition may be considered a fatal fault in applications that require continuous, uninterrupted reception and should transition the ENP-3511 into a fault condition.
Rx Overrun	Indicates that the receiver was unable to complete reception of the TDM packet. This fault may be caused by overloading of the memory and / or IX bus interfaces or other problems not yet identified. This condition is detected by the TDM to IX bus interface hardware.

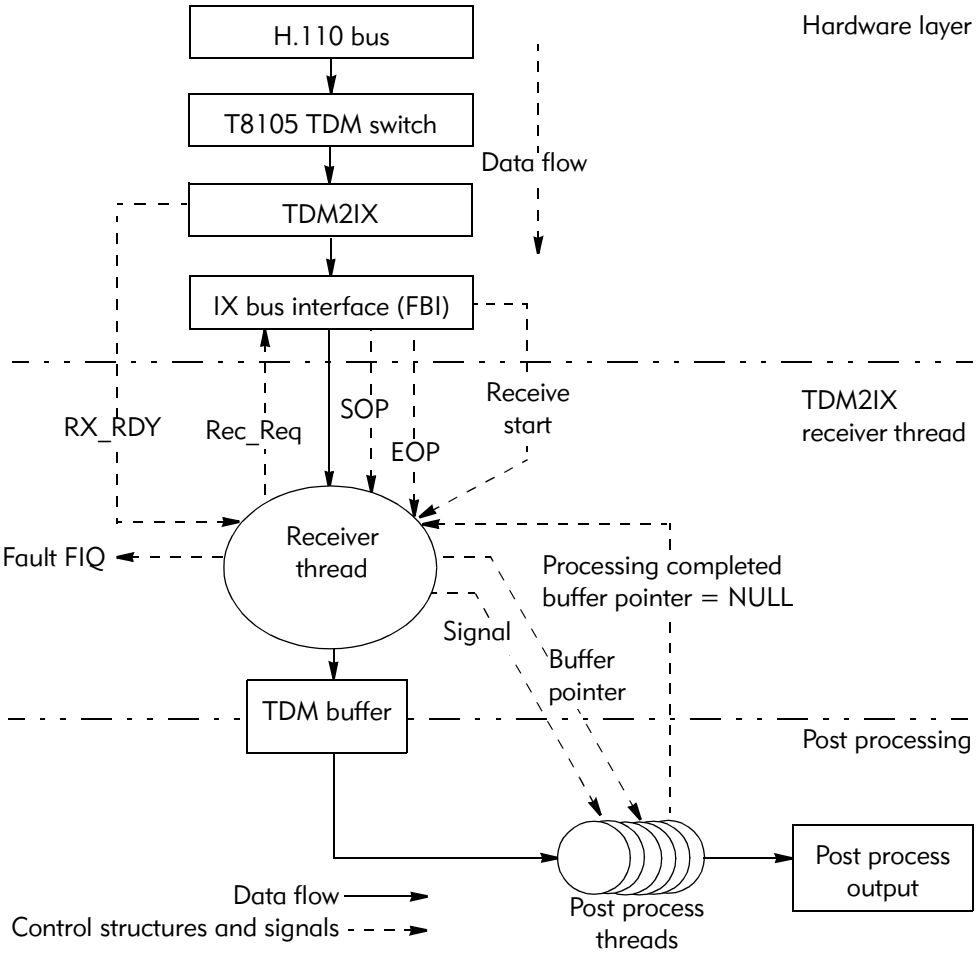
The following statistic is reported by the receive thread for debug purposes.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Rx TDM2IX packet count

AAPI function calls are provided to read the data. All counters are unsigned and roll over at 32 bits.The next figure shows a simplified signal and data flow diagram for the TDM2IX receive operation.

Figure 15-2. IXF440 packet data flow



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IX Transmit module

Transmit data buffer(s)

The TDM transmit data is stored in one-to-many SDRAM buffer(s) allocated by the StrongARM core. Each TDM buffer is allocated 2048 bytes by default. This corresponds to the maximum hardware capacity of 256 transmit channels. Each channel consists of 8 bytes, representing eight 125 microsecond frames of TDM data.

Transmit buffer queue

The transmit queue is a dedicated data word in SRAM. The location of the transmit queue data word (TxTDM2IXBase) is defined at initialization. It can be accessed through AAPI C function calls. After a packet is assembled, a buffer pointer is placed into this location. After transmission, the TDM transmitter writes a NULL to the value in this location. This signals to the transmit assembly process that the packet is transmitted. If, on the next transmit the transmitter detects a null pointer in the transmit queue, this indicates that the assembly process did not complete in the required 850 millisecond period.

Error conditions

The TDM transmitter reports two error conditions: buffer underrun and transmit underrun. The next table details those error conditions.

TDM2IX transmitter error conditions

Buffer Underwrite	The transmit driver indicated that a NULL pointer value was read from the transmit queue. This condition should be considered fatal if TDM packet continuity must be maintained. This fault is indicative of the TDM packet builder thread using more that one millesecond during the transmit packet assembly. The TDM packet for that time period must be completly assembled and the queue must contain the buffer pointer before the signal is received by the registered packet builder. This fault is detected in firmware and should be considered fatal.
Buffer Underrun	The TDM2IX device has requested data from the transmitter. The transmitter thread has not initiated transfer. This is indicative of SDRAM memory or IX bus congestion.

The transmit threads report the following statistics for debug purposes.

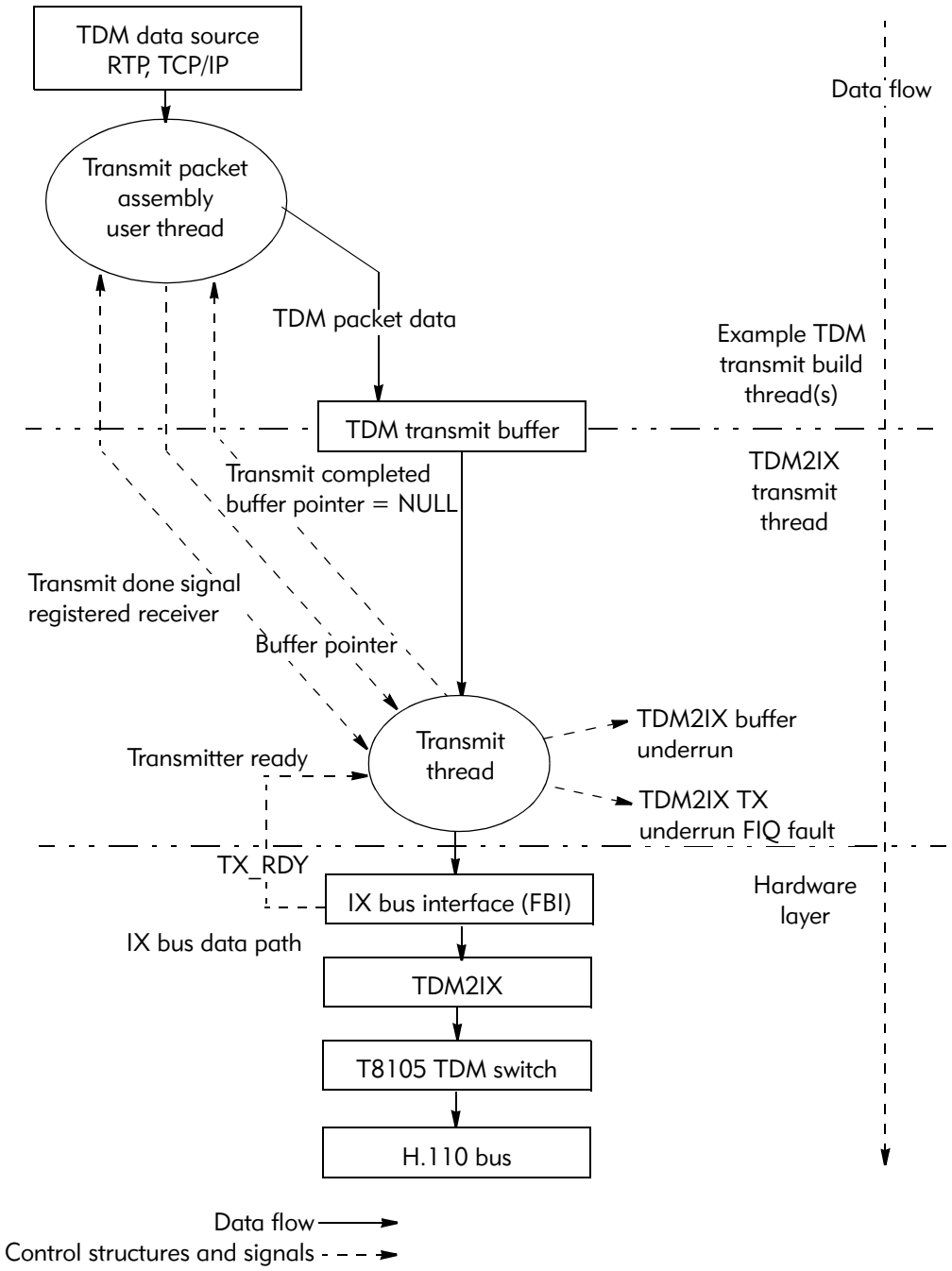
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Tx TDM2IX packet count

AAPI functions read the data. All counters roll over at 32bits.

This figure shows the signal and data flow for the TDM2IX receive operation:

Figure 15-3. IXF440 packet data flow



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IX interrupt routines

Include FileTdm2ix.h

Library Tdm2ix.a

TDM2IX can generate two interrupt types:

- Receive overflow
- Transmit underflow

For more information, see the respective hardware reference manuals.

Whenever a TDM2IX interrupt occurs, TDM2IXUserHandler is called and a structure of type InterruptStatus is passed to that function.

Table 15-1. InterruptStatus structure

Field	Value
InterruptStatus.device	INTERRUPT_FROM_TDM2IX
InterruptStatus.status	Contains TDM2IX_STATUS_REG
InterruptStatus.portStatus	Not used.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Function list

Use this table to identify the TDM2IX functions you want to use. Use the function description later in this chapter to obtain detailed information, including syntax and parameter values.

Table 15-2. TDM2IX microcode functions

Function	Description	Page
TDM2IXClosePort	Closes a TDM2IX port.	301
TDM2IXDisableAllInterrupt	Disables all TDM2IX interrupt events.	302
TDM2IXDisableInterrupt	Disables a specific TDM2IX interrupt event.	303
TDM2IXEnableAllInterrupt	Enables all TDM2IX interrupt events.	304
TDM2IXEnableInterrupt	Enables a specific TDM2IX interrupt event.	305
TDM2IXGetRxThreadError	Gets an error message from the TDM2IX receive thread.	306
TDM2IXGetTxThreadError	Gets an error message from the TDM2IX receive thread.	307
TDM2IXHookPort	Connects the TDM2IX receiver threads to a receive microcode helper thread for post receive operations. Assigns an SDRAM memory location as a pointer to the TDM2IX receive buffer base.	308
TDM2IXOpenPort	Opens an TDM2IX port and returns a handle to the port.	309
TDM2IXReadPacket	Reads a packet from the TDM2IX receive queue.	310
TDM2IXReadReg	Reads a TDM2IX hardware register.	311
TDM2IXResetPort	Resets a TDM2IX device.	312
TDM2IXSetInterruptHandler	Installs a user defined interrupt handler for TDM2IX interrupts.	313
TDM2IXSetRxLength	Sets the number of timeslots to be received from the TDM2IX.	314
TDM2IXSetTxLength	Sets the number of timeslots to be Transmitted by the TDM2IX device.	315
TDM2IXStartPort	Enables the receive and transmit operation of a TDM2IX.	316
TDM2IXStopPort	Disables receive and transmit operation of the TDM2IX.	317
TDM2IXUnhookPort	Disconnects the TDM2IX from the specified Rx thread	318
TDM2IXWritePacket	Writes a packet into the TDM2IX transmit queue.	319
TDM2IXWriteReg	Writes a TDM2IX register.	320

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXClosePort

Closes a TDM2IX port.

Syntax

```
t_ENPStatus TDM2IXClosePort (
    t_ENPPortHandle PortHandle
)
```

Parameters

PortHandle Specifies an handle to a TDM2IX port.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXDisableAllInterrupt

Disables all TDM2IX interrupt events.

Syntax

```
TDM2IXDisableAllInterrupt(
   t_ENPPortHandle PortHandle
);
```

Parameters

PortHandle

Handle to the TDM2IX port.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXDisableInterrupt

Disables a specific TDM2IX interrupt event.

Syntax

```
TDM2IXDisableInterrupt(
   t_ENPPortHandle PortHandle,
   t_TDM2IXInterruptEvent Event
);
```

Parameters

PortHandle

Handle to the TDM2IX port.

Event Specific interrupt event of type t_TDM2IXInterruptEvent.

t_TDHHM2IXInterruptEvent

TDM2IX_RCV_OVFL	Receive overflow
TDM2IX_XMT_UNFL	Transmit underflow

The default handler clears the interrupts and disables any further interrupts from that specific source within the device, which caused the interrupt and passes the above information to the user handler. You must re-enable any interrupts, if needed.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXEnableAllInterrupt

Enables all TDM2IX interrupt events.

Syntax

```
TDM2IXEnableAllInterrupt(
   t_ENPPortHandle PortHandle
);
```

Parameters

PortHandle

Handle to the TDM2IX port.

Return value

To be developed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXEnableInterrupt

Enables a specific TDM2IX interrupt event.

Syntax

```
TDM2IXEnableInterrupt(
   t_ENPPortHandle PortHandle,
   t_TDM2IXInterruptEvent Event
);
```

Parameters

PortHandle
Handle to the TDM2IX port.

Event Specific interrupt event of type t_TDM2IXInterruptEvent.

t_TDHHM2IXInterruptEvent

TDM2IX_RCV_OVFL	Receive overflow
TDM2IX_XMT_UNFL	Transmit underflow

The default handler clears the interrupts and disables any further interrupts from that specific source within the device, which caused the interrupt and passes the above information to the user handler. You must re-enable any interrupts, if needed.

Return value

ENP_SUCCESS

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXGetRxThreadError

Gets an error message from the TDM2IX receive thread.

Syntax

```
t_ENPStatus TDM2IXGetRxError(
    Bit16 ThreadID,
    t_TDM2IXRxError *RxError
)
```

Parameters

ThreadID Specifies the thread to get the error message from.

RxError Pointer to location to store the error message. Valid values include:

TDM2IX_RX_OVERRUN
TDM2IX_RX_OVERWRITE

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_OBJECT

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXGetTxThreadError

Gets an error message from the TDM2IX transmit thread.

Syntax

```
t_ENPStatus TDM2IXGetTxError(
    Bit16 ThreadID,
    t_TDM2IXTxError *TxError
)
```

Parameters

ThreadID Specifies the thread to get the error message from.

TxError The error message. Valid values include:

TDM2IX_BUFFER_UNDERRUN
TDM2IX_BUFFER_UNDERWRITE

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_OBJECT

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXHookPort

Connects the TDM2IX receiver threads to a receive microcode helper thread for post receive operations. Assigns an SDRAM memory location as a pointer to the TDM2IX receive buffer base.

Syntax

```
t_ENPStatus TDM2IXHookPort(
    t_ENPPortHandle PortHandle,
    Bit16 RxThreadID,
    Bit16 TxThreadID,
    Bit16 RxSigThreadID
    Bit32 RxDramBufferBase
)
```

Parameters

PortHandle Handle to an open TDM2IX device.

RxThreadID

ThreadID of the receiver thread to use. Valid values include:

IXP_THREAD0 to IXP_THREAD24 or
IXP_THREADNONE

TxThreadID

ThreadID of the receiver thread to use. Valid values include:

IXP_THREAD0 to IXP_THREAD24 or
IXP_THREADNONE

RxSigThreadID

ThreadID of thread to signal upon completed packet reception. Valid values include:

IXP_THREAD0 to IXP_THREAD24 or
IXP_THREADNONE

RxDramBufferBase

A pointer to an SDRAM location that contains the base address of the receive buffer.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_INSUFFICIENT_RESOURCES
ENP_THREAD_NOT_READY

Comments

TDM2IX receiver and/or transmitter microcode must be running on the microengines.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXOpenPort

Opens an TDM2IX port and returns a handle to the port.

Syntax

```
t_ENPStatus TDM2IXOpenPort (
    Bit16 MacNum,
    Bit16 PortNum,
    t_ENPPortHandle *PortHandle
)
```

Parameters

MacNum MAC number of the TDM2IX device. Valid value:

IXP_MAC1

(this is the only TDM2IX device on the ENP board).

PortNum Port number on the TDM2IX device. Valid value:

IXP_PORT0

PortHandle

Pointer to location where the ENPPortHandle will be returned.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_PORT_ALREADY_OPEN
ENP_INVALID_PORT_TYPE
ENP_INVALID_MACPORT

Comments

You must call ENP3511_InitApp before any ports are opened.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXReadPacket

Reads a packet from the TDM2IX receive queue.

Syntax

```
t_ENPStatus TDM2IXReadPacket(
    t_ENPPortHandle *PortHandle,
    Bit32 *ReadPacket
)
```

Parameters

PortHandle Handle to the opened TDM2IX device.

ReadPacket Pointer to location where the address of the buffer descriptor read from the receive queue will be placed.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_INVALID_SRAM_ADDRESS
ENP_QUEUE_EMPTY
ENP_LOCK_FAILURE

Comments

The port is open, initialized, and running.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXReadReg

Reads a TDM2IX hardware register.

Syntax

```
t_ENPStatus TDM2IXReadReg(
    t_IXFTDM2IXReg TDM2IXReg,
   Bit8 *ReadValue
)
```

Parameters

TDM2IXReg Specifies an TDM2IX register. See TDM2IX.h header file for valid values.

ReadValue Pointer to location to place read data.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXResetPort

Resets a TDM2IX device.

Syntax

```
t_ENPStatus TDM2IXResetPort(
    T_ENPPortHandle PortHandle
)
```

Parameters

PortHandle Handle to the open TDM2IX device.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXSetInterruptHandler

Installs a user defined interrupt handler for TDM2IX interrupts.

Syntax

```
TDM2IXSetInterruptHandler(
   void (*TDM2IXUserHandler)(InterruptStatus)
);
```

Parameters

Pointer to the user-defined interrupt function.

Return value

ENP_SUCCESS
ENP_INT_HANDLER_INSTALL_FAILURE

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXSetRxLength

Sets the number of timeslots to be received from the TDM2IX.

Syntax

```
t_ENPStatus TDM2IXSetRxLength(
    t_ENPPortHandle PortHandle,
    UCHAR Length
)
```

Parameters

PortHandle
: Handle to the opened TDM2IX device.

Length
: Specifies the number of timeslots to read from the TDM2IX. Number of received timeslots is as follows:

  (Length +1) * 8

  Length implies a multiple of eight timeslots; valid range is 0x00 to 0x1f (0 to 31 decimal).

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Device must be opened but stopped.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXSetTxLength

Sets the number of timeslots to be Transmitted by the TDM2IX device.

Syntax

```
t_ENPStatus TDM2IXSetTxLength(
    t_ENPPortHandle PortHandle,
    UCHAR Length
)
```

Parameters

PortHandle Handle to a open TDM2IX device.

Length Specifies the number of timeslots to transmit from the TDM2IX. Number of received timeslots is as follows:

(Length +1) * 8

Length implies a multiple of eight timeslots; valid range is 0x00 to 0x1f (0 to 31 decimal).

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

The device must be opened but stopped.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXStartPort

Enables the receive and transmit operation of a TDM2IX.

Syntax

```
t_ENPStatus TDM2IXStartPort(
    t_ENPPortHandle *PortHandle
)
```

Parameters

PortHandle Handle to an open TDM2IX device.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Device must be opened with TDM2IXOpenPort.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXStopPort

Disables receive and transmit operation of the TDM2IX.

Syntax

```
t_ENPStatus TDM2IXStopPort(
    t_ENPPortHandle PortHandle
)
```

Parameters

PortHandle Handle to the opened TDM2IX device.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Device must be opened with TDM2IXOpenPort.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXUnhookPort

Disconnects the TDM2IX from the specified Rx thread

Syntax

```
t_ENPStatus TDM2IXUnhookPort(
    t_ENPPortHandle PortHandle
)
```

Parameters

PortHandle Handle to a open TDM2IX device.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_THREAD_NOT_READY

Comments

TDM2IX was hooked.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXWritePacket

Writes a packet into the TDM2IX transmit queue.

Syntax

```
t_ENPStatus TDM2IXWritePacket(
    t_ENPPortHandle *PortHandle,
    Bit32 *WritePacket
)
```

Parameters

PortHandle Handle to the opened TDM2IX device.

WritePacket Address of buffer descriptor to write to the transmit queue.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE
ENP_INVALID_SRAM_ADDRESS

Comments

Port is open, initialized, and running.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXWriteReg

Writes a TDM2IX register.

Syntax

```
t_ENPStatus TDM2IXWriteReg(
    t_TDM2IXReg TDM2IXReg,
    Bit8 WriteValue
)
```

Parameters

TDM2IXReg Specifies a TDM2IX register. See TDM2IX.h header file for valid values.

WriteValue
Data to write.

Return values



For detailed information about these return values, see *Appendix A, Status codes*.

ENP_SUCCESS
ENP_INVALID_HANDLE

Comments

Port is open.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

III Appendices

The appendices include:

Appendix A: Status codes

Lists and describes the ENP SDK status codes.

Appendix B: Diagnostic tests

Explains how to use the diagnostic utilities provided as part of your ENP SDK software.

Appendix C: Re-programming the flash chip

Explains how to update or recover programs that reside on the flash chip. You accomplish this by re-programming all or part of an ENP board's flash chip.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A Status codes

Status codes are numeric values that represent unique conditions. Each code also has a mnemonic such as ENP_SUCCESS, which indicates successful completion.

The next table lists the ENP SDK status codes and mnemonic names:

Code	Name
0	ENP_SUCCESS
1	ENP_INVALID_HANDLE
2	INVALID_PORT_HANDLE
3	ENP_INVALID_OBJECT
4	ENP_PORT_ALREADY_OPEN
5	ENP_INVALID_MACPORT
6	ENP_INVALID_ADDRESS
7	ENP_INSUFFICIENT_RESOURCES
8	ENP_THREAD_NOT_READY
9	ENP_INVALID_OPERATION
10	ENP_STACK_IN_USE
11	ENP_RXQUEUE_EMPTY
12	ENP_INVALID_SRAM_ADDRESS
13	ENP_INVALID_DRAM_ADDRESS
14	ENP_INVALID_STACKID
15	ENP_INVALID_HOOK_HANDLE
16	ENP_INVALID_PORT_TYPE
17	ENP_TDM_REGISTER_READ_ERROR
18	ENP_TDM_MEMORY_READ_ERROR
19	ENP_TDM_FAILURE
20	ENP_LOCK_FAILURE
21	ENP_INT_FAILURE
22	ENP_INT_HANDLER_INSTALL_FAILURE
23	ENP_TXQUEUE_NOT_DEFINED
24	ENP_QUEUE_FULL

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

B Diagnostic tests

The System Monitor includes tests that verify the operational status of individual hardware components. You execute via the RS-232 port on the front panel, and the test results and execution state display on the monitor.

You can recall previous commands by pressing the up and down arrow keys. You can also shorten command and keywords, when not ambiguous.

Use this table to identify the diagnostic tests you want to use. Use the test description later in the chapter to get detailed information, including syntax and parameter values:

Table B-1. Diagnostic tests

Function	Description
test 21555	Checks the validity of the 21555 SROM data against the factory default values.
test cpld	Tests the CPLD.
test e1t1	Tests the optional 4-port E1/T1 PMC.
test ethernet	Tests the PCI Ethernet controller, if present.
test flash	Sizes the flash array by checking the Manufacture and Device ID.
test h110	Tests the H110 interface.
test ipmi	Checks the operation of the KCS IPMI interface on the ENP3511.
test ixf440	Exercises the IXF440 10/100 Ethernet controller by downloading transmit and receive microcode into the microengines, and using the scratch pad and SDRAM to communicate to the ARM core.
test ixf440	Exercises the IXF440 10/100 Ethernet controller by downloading transmit and receive microcode into the microengines, and using the scratch pad and SDRAM to communicate to the ARM core.
test macaddrs	Tests the IXF MAC addresses.
test pci	Enumerates the PCI bus devices.
test scratch	Writes a series of test patterns to scratch pad memory and confirms that the same data is read back.
test sdram	Writes a series of test patterns to the SDRAM array and confirms that the same data is read back.
test sram	Writes a series of test patterns to the SRAM array and confirms the same data is read back.
test t8105	Tests the T8105 H.110 controller.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-1. Diagnostic tests

Function	Description
test tdm2ix	Tests the TDM2IX device.
test teeprom	Checks the GPIO I2C bus for presence of serial EEPROM. It reads the serial EEPROM data and checks for valid checksum.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test 21555

Checks the validity of the 21555 SROM data against the factory default values.

Syntax

```
test 21555 loop mode display
```

Parameters

`loop` (Optional) Runs the test continuously. To exit the test, enter Q, q, or Ctrl+C.

`mode` (Optional) You can use one of these values:

- `external` Runs the test in external mode.

  In this mode a PCI host system must be present and preferably running DOS. The test opens a 1Mbyte memory window to the host system. The test then test writes a pattern to the window a byte at a time, then reads and verifies the pattern for accuracy.

  You can also run this test without any OS booted on the host machine, provided the BIOS initializes the chipset and memory.

- `extended` Generates an extended number of data transfers through the 21555 bridge. This test autmatically operates in external mode.

`display` (Optional) You can use this value:

- `verbose` Displays status information during the test. If any part of the test fails, running the test again with this option can provide additional information regarding the failure.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test cpld

Tests the CPLD.



This test runs only on the ENP-3511 board.

The test checks the following:

- **CPLD code/revision number**: Checks the Product Code and Revision and prompts you to verify. This test displays this information only if you use the `attended` option.
- **CPLD data out**: Makes all four bits input and checks that the Data Value is 0x0F due to the pull-up resistors.
- **CPLD interrupt pending**: Writes and reads back the 0x40 mask register values, masks the IPMI interrupts, then unmasks the RTM interrupts.

  If you use the `attended` option, this test asks you to cause the edge for each of the four RTM bits, waits for you to enter 'y', then checks to see that the corresponding Interrupt Pending and IREG bits are set. It then clears those bits and continues. It then saves and restores mask state.
- **CPLD watchdog enable**: Displays register contents. Enables watchdog and holds it off for about 10 seconds, then disables the watchdog and waits for about 10 seconds to ensure it is disabled.



To display command syntax, enter the command followed by "help".

Syntax

```
test cpld loop attended display
```

Parameters

`loop` (Optional) Tests the CPLD continuously in internal loop-back mode. Writes and reads back the16 direction values, Sets the direction to output (0x0F), then writes the 16 data values in a continuous loop. Use one of these values to exit the test:

y	Forces the test to pass.
n	Forces the test to fail.

q or Ctrl_c Aborts the test.

`attended` (Optional) Prompts you for input following each check, or portion of the test. The test continues after you provide input.

If you include this parameter, CPLD watchdog enable sets the bit when you enter 'c'. The system resets. If reset doesn't occur in about 10 seconds, the test resets the bit and fails.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

If you omit this parameter:

- CPLD code/revision number compares Product Code and Revision to hard-coded values.
- CPLD data out performs one pass through the data out.
- CPLD interrupt pending performs only a mask write/read.
- CPLD watchdog enable displays register contents.

display (Optional) You can use one or more of these values:

`verbose` Displays status information during the test. If any part of the test fails, running the test again with this option can provide additional information regarding the failure.

If you include this paramter, CPLD interrupt pending displays register values.

`debug` Displays extended status information during the test. Running the test with this option provides useful information only if you are famliar with the chip and its address structure.

`DEBUG` Displays all status information during the test. Run the test with this option only if the manufacturer requests it for troubleshooting.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test e1t1

Tests the optional 4-port E1/T1 PMC.

This test runs only on the ENP-3511 board.

Syntax

```
test e1t1 loop external payload interactive display
```

Parameters

`loop` (Optional) Runs the test continuously. To exit the test, enter Q, q, or Ctrl+C.

`external` (Optional) Performs an external loopback test in T1 mode using any of the E1T1 ports. Can be used as a comprehensive communication test between the ports and the baseboard. Needs an external loop connector.



This test requires the T1/E1 loopback loopback connector. If not present, the test will fail. For pinout information, see *T1/E1 loopback* on page 348.

`payload` (Optional) Performs a T1 data loopback test between an external T1 frame generator and the E1T1 PMC. Can be used to perform user specified data loopback. This test needs separate equipment that can generate T1 data. Do not run this test if such equipment is not available.

`interactive`
(Optional) Runs diagnostics in interactive mode. Use this option only if you are familiar with the E1T1 device.

`display` (Optional) You can use one or more of these values: verbose Displays status information during the test. If any part of the test fails, running the test again with this option can provide additional information regarding the failure.

`debug` Displays extended status information during the test. Running the test with this option provides useful information only if you are famliar with the various E1T1 PMC components and its address structure.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test ethernet

Tests the PCI Ethernet controller, if present.

This test checks the following:

- **Check for All 559 devices**: Ensures at least two devices exist at the proper PCI addresses. If it finds more devices, it tests them.
- **Serial EEPROM checksum**: For each device, does a checksum and excludes the invalid MAC addresses of 0x000000000000 and 0xFFFFFFFFFFFF.
- **Port selftest**: For each device, runs the built-in selftest of the part.
- **MAC internal loopback**: For each device, runs a single packet of 1500 bytes through examines error status bits and compares data. This is not a performance test.
- **External loopback**: If the `external` option is used, runs a single packet of 1500 bytes through examines error status bits and compares data for each device. This is not a performance test. This test requires a single-end loopback connector.

To display command syntax, enter the command followed by "help".

Syntax

```
test ethernet loop external pmc location display
```

Parameters

`loop` (Optional) Tests the 82559 continuously. Use one of these to exit the test:

- y — Forces the test to pass.
- n — Forces the test to fail.
- q or Ctrl_c — Aborts the test.

`pmc` (Optional) Tests the PMC connector. This test requires an 82559 PMC card connected to the ENP board.

`location` (Optional) You can use one of these values:

`external` (Optional) Tests the 82559 with external loop-back connectors.

This test requires the Ethernet loopback loopback connector. If not present, the test will fail. For pinout information, see *Ethernet loopback* on page 348.

`front` Forces Channel 0 External Loopback to the front connector. (i.e. Manual mode).

This option runs only on the ENP-3511 board.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

`rear` Forces Channel 0 External Loopback to the rear connector. (i.e. Manual mode).



This option runs only on the ENP-3511 board.

`display` (Optional) You can use one or more of these values:

`verbose` Displays status information during the test. If any part of the test fails, running the test again with this option can provide additional information regarding the failure.

`debug` Displays extended status information during the test. Running the test with this option provides useful information only if you are famliar with the chip and its address structure.

`DEBUG` Displays all status information during the test. Run the test with this option only if the manufacturer requests it for troubleshooting.

This table describes the results of using the `external`, `front`, and `rear` options. Columns Front and Rear indicate whether the loopback connector is installed.



Options presented in this table apply only to the ENP-3511 board.

Front	Rear	Command	Result
No	No	`test external`	Fail
No	No	`test front`	Fail
No	No	`test rear`	Fail
No	Yes	`test external`	Pass
No	Yes	`test front`	Fail
No	Yes	`test rear`	Pass
Yes	No	`test external`	Pass
Yes	No	`test front`	Pass
Yes	No	`test rear`	Fail
Yes	Yes	`test external`	Pass
Yes	Yes	`test front`	Pass
Yes	Yes	`test rear`	Pass

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test flash

Sizes the flash array by checking the Manufacture and Device ID.

This test does check erase or program operations, nor does it overwrite the flash's boot block.

Syntax

```
test flash
```

Parameters

None.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test h110

Tests the H110 interface.



- This test requires the CompactPCI H110 loopback connector. If not present, the test will fail. For pinout information, see *CompactPCI H110 loopback* on page 349.
- This test runs only on the ENP-3511 board.

Syntax

```
test h110 loop interactive display
```

Parameters

`loop` (Optional) Runs the test continuously. To exit the test, enter Q, q, or Ctrl+C.

`interactive` (Optional) Runs diagnostics in interactive mode. Use this option only if you are familiar with the TDM2IX device.

`display` (Optional) You can use one or more of these values:

`verbose` Displays status information during the test. If any part of the test fails, running the test again with this option can provide additional information regarding the failure.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test ipmi

Checks the operation of the KCS IPMI interface on the ENP3511.

This test first checks to see if the IPMI interface is present and available. The test then submits various commands and checks the validity of the response from the IPMI controller.

Syntax

```
test ipmi loop extended display
```

Parameters

`loop` (Optional) Runs the test continuously. To exit the test, enter Q, q, or Ctrl+C.

`extended` (Optional) Runs the test in extended mode. In this mode, the diagnostic submits additional IPMI commands and checks for a valid response.

`display` (Optional) You can use one of these values:

- `verbose` Displays status information during the test. If any part of the test fails, running the test again with this option can provide additional information regarding the failure.
- `debug` Displays extended status information during the test. Running the test with this option provides useful information only if you are famliar with the chip and its address structure.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test ixf440

Exercises the IXF440 10/100 Ethernet controller by downloading transmit and receive microcode into the microengines, and using the scratch pad and SDRAM to communicate to the ARM core.

Syntax

```
test ixf440 loop mode display
```

Parameters

`loop` (Optional) Runs the test continuously. To exit the test, enter Q, q, or Ctrl+C.

`mode` (Optional) You can use one of these values:

`external` Tests the IXF440 with an external loop-back connector.



This test requires four Ethernet crossover loopback connectors, which are installed between the eight ports. If not present, the test will fail. For pinout information, see *Ethernet crossover loopback* on page 348.

`extended` In this mode the test generates an extended number of packets and requires external loopbacks.

`display` (Optional) You can use one or more of these values:

`verbose` Displays status information during the test. If any part of the test fails, running the test again with this option can provide additional information regarding the failure.

`debug` Displays extended status information during the test. Running the test with this option provides useful information only if you are famliar with the chip and its address structure.

Additional information

The IXbus test used for beta release is based on the diagnostics supplied in the INTEL V1.3 IXA SDK and performs the following:

- Initializes the scratch RAM used in this test.
- Scans for the IXF device. If this is not detected the test returns with failure report.
- Passes the MAC parameters needed by the microcode using the scratch pad memory.
- Generates the test packets in SDRAM memory.
- Initializes the IXF device and sets the mode to internal or external loopback.
- Resets all microengines and initializes program counters and context status.
- Loads microcode into the engines (0–3: receive) (4: receive scheduler) (5: transmit).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Programs the ready bus sequencer.
- Enables the microengines, enables the microcode contexts, and sets up the events that wake the contexts.
- The receive threads check that the received data is same as the original transmitted data and, if different, increments the RX_ERROR_COUNTER.
- The test continues until the specified number of packets are sent, then the microengines are disabled.
- All status is reported to the user using the serial port.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test ixf1002

Exercises the IXF1002 10/100 Ethernet controller by downloading transmit and receive microcode into the microengines, and using the scratch pad and SDRAM to communicate to the ARM core.

Syntax

```
test ixf1002 loop mode display
```

Parameters

`loop` (Optional) Runs the test continuously. To exit the test, enter Q, q, or Ctrl+C.

`mode` (Optional) You can use one of these values:

`external` Tests the IXF1002 with an external loop-back connector.



This test requires four Ethernet crossover loopback connectors, which are installed between the eight ports. If not present, the test will fail. For pinout information, see *Ethernet crossover loopback* on page 348.

Issue: Need to reference correct loopback connector.

`extended` In this mode the test generates an extended number of packets and requires external loopbacks.

`display` (Optional) You can use one or more of these values:

`verbose` Displays status information during the test. If any part of the test fails, running the test again with this option can provide additional information regarding the failure.

`debug` Displays extended status information during the test. Running the test with this option provides useful information only if you are famliar with the chip and its address structure.

Additional information

The IXbus test used for beta release is based on the diagnostics supplied in the INTEL V1.3 IXA SDK and performs the following:

- Initializes the scratch RAM used in this test.
- Scans for the IXF device. If this is not detected the test returns with failure report.
- Passes the MAC parameters needed by the microcode using the scratch pad memory.
- Generates the test packets in SDRAM memory.
- Initializes the IXF device and sets the mode to internal or external loopback.
- Resets all microengines and initializes program counters and context status.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- Loads microcode into the engines (0–3: receive) (4: receive scheduler) (5: transmit).
- Programs the ready bus sequencer.
- Enables the microengines, enables the microcode contexts, and sets up the events that wake the contexts.
- The receive threads check that the received data is same as the original transmitted data and, if different, increments the RX_ERROR_COUNTER.
- The test continues until the specified number of packets are sent, then the microengines are disabled.
- All status is reported to the user using the serial port.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test macaddrs

Tests the IXF MAC addresses.



To display command syntax, enter the command followed by "help".

Upon starting, the Boot Manager runs this test with the `init` option. If the Boot Manager displays the message below as it starts, run this test with the `debug` option to determine the reason for failure:

```
WARNING: IXF440 MAC Address Initialzation failed. sts = 130
```

Syntax

```
test macaddrs test init clear show display
```

Parameters

`test` (Optional; default) Reads the IXF440 registers.

`init` (Optional) Initializes the IXF PHY_ADR registers registers from the serial EEPROM. If no valid addresses exist in the serial EEPROM, this test fails.

`clear` (Optional) Clears, or zeros out, the IXF440 PHY_ADR contents.

`show` (Optional) Displays what was put in the IXF PHY_ADR registers when you ran this test using the `init` option.

`display` (Optional) You can use one or more of these values:

- `verbose` Displays status information during the test. If any part of the test fails, running the test again with this option can provide additional information regarding the failure.
- `debug` Displays extended status information during the test. Running the test with this option provides useful information only if you are famliar with the chip and its address structure.
- `DEBUG` Displays all status information during the test. Run the test with this option only if the manufacturer requests it for troubleshooting.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test pci

Enumerates the PCI bus devices.

Syntax

```
test pci
```

Parameters

None.

Additional information

The following displays when this test runs sucessfully.

```
enp> sho pci
Enumerating PCI configuration space...
Vendor ID: 8086, Device ID: b555
Vendor ID: 8086, Device ID: 1209
Vendor ID: 8086, Device ID: 1209

enp>
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test scratch

Writes a series of test patterns to scratch pad memory and confirms that the same data is read back.

Syntax

```
test scratch loop display
```

Parameters

`loop` (Optional) Runs the test continuously. To exit the test, enter Q, q, or Ctrl+C.

`display` (Optional) You can use one or more of these values:

`verbose` Displays status information during the test. If any part of the test fails, running the test again with this option can provide additional information regarding the failure.

Return values

Success: SCRATCH Test Passed.

Failure: SCRATCH Test Failed: Test *PassNumber* address = *Address*, expected = *Expected*, actual = *Actual*

Additional information

The following test patterns are written to the array:

Test pass	Data pattern	Test pass	Data pattern
1	`0x00000000`	4	`0x55555555`
2	`0xffffffff`	5	Walking ones
3	`0xaaaaaaaa`	6	Address

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test sdram

Writes a series of test patterns to the SDRAM array and confirms that the same data is read back.

Syntax

```
test sdram loop fast extended display
```

Parameters

`loop` (Optional) Runs the test continuously. To exit the test, enter Q, q, or Ctrl+C.

`fast` (Optional) Runs the test in fast mode, passing through the array once.

`extended` (Optional) Tests cacheable memory, mapped to address 0000 0000.

Portions of this memory region are used by the System Monitor and are therefore not tested by this diagnotic. These memory regions are tested before the System Monitor is called by the minimal memory test.

`display` (Optional) You can use one or more of these values:

`verbose` Displays status information during the test. If any part of the test fails, running the test again with this option can provide additional information regarding the failure.

Return values

Success: SDRAM Test Passed.

Failure: SDRAM Test Failed: Test *PassNumber* address = *Address*, expected = *Expected*, actual = *Actual*

Additional information

The following test patterns are used:

Test pass	Data pattern
1	`0x00000000`
2	`0xffffffff`
3	`0xaaaaaaaa`
4	`0x55555555`
5	Walking ones
6	Address

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test sram

Writes a series of test patterns to the SRAM array and confirms the same data is read back.

Upon completion, the monitor clears the full SRAM memory array. The cache is enabled by default.

Syntax

```
test sram loop display
```

Parameters

`loop` (Optional) Runs the test continuously. To exit the test, enter Q, q, or Ctrl+C.

`display` (Optional) You can use one or more of these values:

- `verbose` Displays status information during the test. If any part of the test fails, running the test again with this option can provide additional information regarding the failure.

Return values

Success: SRAM Test Passed.

Failure: SRAM Test Failed: Test *PassNumber* address = *Address*, expected = *Expected*, actual = *Actual*

Additional information

The following test patterns are written to the array:

Test pass	Data pattern	Test pass	Data pattern
1	`0x00000000`	4	`0x55555555`
2	`0xffffffff`	5	Walking ones
3	`0xaaaaaaaa`	6	Address

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test t8105

Tests the T8105 H.110 controller.



This test runs only on the ENP-3511 board.

Syntax

```
test t8105 loop interactive display
```

Parameters

`loop` (Optional) Runs the test continuously. To exit the test, enter Q, q, or Ctrl+C.

`interactive` (Optional) Runs diagnostics in interactive mode. Use this option only if you are familiar with the TDM2IX device.

`display` (Optional) You can use one or more of these values:

- `verbose` Displays status information during the test. If any part of the test fails, running the test again with this option can provide additional information regarding the failure.
- `debug` Displays extended status information during the test. Running the test with this option provides useful information only if you are famliar with the chip and its address structure.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test tdm2ix

Tests the TDM2IX device.



This test runs only on the ENP-3511 board.

Syntax

```
test tdm2ix loop interactive display
```

Parameters

`loop` (Optional) Runs the test continuously. To exit the test, enter Q, q, or Ctrl+C.

`interactive` (Optional) Runs diagnostics in interactive mode. Use this option only if you are familiar with the TDM2IX device.

`display` (Optional) You can use one ore more of these values:

- `verbose` Displays status information during the test. If any part of the test fails, running the test again with this option can provide additional information regarding the failure.
- `debug` Displays extended status information during the test. Running the test with this option provides useful information only if you are famliar with the chip and its address structure.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

test teeprom

Checks the GPIO I2C bus for presence of serial EEPROM. It reads the serial EEPROM data and checks for valid checksum.

Checks the ability of the Boot Manager to read Serial EEPROM and initialize the IXF440 PHY_ADDR registers. These registers are used only for the IXF440 Flow Control feature.



To display command syntax, enter the command followed by "help".

Syntax

`test teeprom` *`mode timing display`*

Parameters

`mode` (Optional) You can use one of these values:

- `show` — If there is a valid checksum, displays a decoded data structure.
- `dump` — Lists EEPROM contents in hex with an ASCII decode on the side. This command doesn't care what is in the EEPROM to start.
  - . — Non-printing characters.
  - ! — Clear fills with this character.
  - ? — Uninitialized data.

`timing` (Optional) You can use this value:

- `clock` — Runs the I2C Clock Forever utility. Rest or power-cycle to quit. Runs fake I2C SpinDelay() clock so that period can be measured with an oscilloscope.

`display` (Optional) You can use one or more of these values:

- `verbose` — Displays status information during the test. If any part of the test fails, running the test again with this option can provide additional information regarding the failure.
- `debug` — Displays extended status information during the test. Running the test with this option provides useful information only if you are famliar with the chip and its address structure.
- `DEBUG` — Displays all status information during the test. Run the test with this option only if the manufacturer requests it for troubleshooting.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Loopback connectors

T1/E1 loopback

This loopback connector is an RJ-45 connector with the following connections:

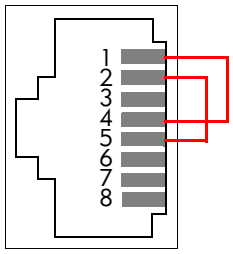


Table B-2. Ethernet loopback pin-out

Connect this pin...	To this pin...
1	4
2	5

Note: All other wires remain unconnected.

For information about using this connector, see *test e1t1* on page 330.

Ethernet loopback

This loopback connector is an RJ-45 connector with the following connections:

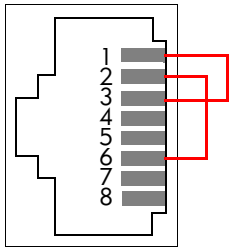


Table B-3. Ethernet loopback pin-out

Connect this pin...	To this pin...
1	3
2	6

Note: All other wires remain unconnected.

For information about using this connector, see *test ethernet* on page 331.

Ethernet crossover loopback

Issue: This connector is for the IXF440; need correct connector for IXF1002 device.

This loopback connector is an RJ-45 connector with the following connections:

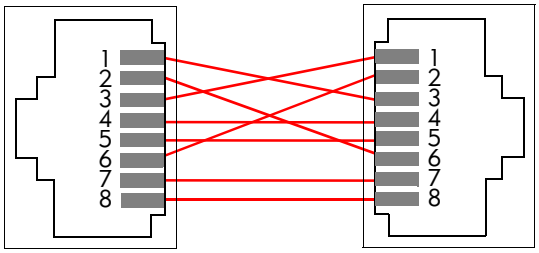


Table B-4. Ethernet crossover loopback pin-out

Connect this pin...	To this pin...	Connect this pin...	To this pin...
1	3	5	5
2	6	6	2

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table B-4. Ethernet crossover loopback pin-out

Connect this pin...	To this pin...	Connect this pin...	To this pin...
3	1	7	7
4	4	8	8

For information about using this connector, see *test ixf440* on page 336.

CompactPCI H110 loopback

This loopback connector is a CompactPCI backplane connector with the following connections:

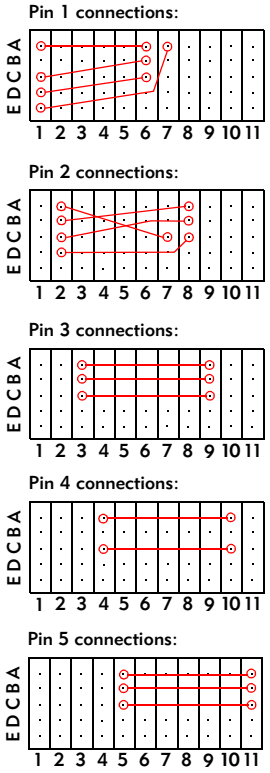


Table B-5. CompactPCI H110 loopback pin-out

Connect this pin...	To this pin...
1A	A6
C1	B6
D1	C6
E1	A7
A2	C7
B2	A8
C2	B8
D2	C8
A3	A9
B3	B9
C3	C9
A4	A10
C4	C10
A5	A11
B5	B11
C5	C11

For information about using this connector, see *test h110* on page 334.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

C Re-programming the flash chip

This appendix details how to update or recover programs that reside on the flash chip. You accomplish this by re-programming all or part of an ENP board's flash chip.

This appendix assumes that you know how to reach the Boot Manager, System monitor, or operating system (OS) of your choice. For more information, see *Configuring the ENP board for boot* on page 10.

When reading this file online, you can immediately view information about any topic by placing the mouse cursor over a task and clicking.

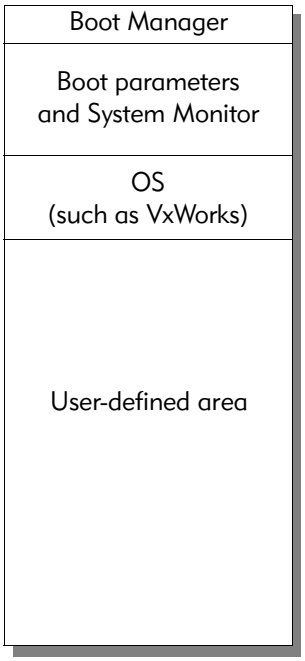
For information about...	Go to this page...
About the flash chip	352
About re-programming the flash chip	353
Before you begin	354
Re-programming the flash chip from VxWorks	356
Re-programming the flash chip from the System monitor	357
Re-programming the flash chip from the Boot Manager	358

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

About the flash chip

The ENP SDK flash chip contains these major sections:

Figure C-1. Flash chip configuration



- **Boot Manager**: A 64 KB, hardware-write-protected area that contains these programs:
  - **Boot Manager**: Selects the OS residing in the flash chip that loads on boot.
  - **Flash utility**: Works together with the Flash utility that runs on the host to update the flash chip.
- **Boot flash parameters**: Contains the parameters required to start the Boot Manager.
- **System monitor**: Provides a rudimentary execution environment and includes both monitor commands and diagnostic tests. For detailed information about this tool's commands and tests, see *Chapter 5, System Monitor software* and *Appendix B, Diagnostic tests*.
- **OS**: Contains operating systems such as VxWorks.
- **User-defined area**: The remaining reprogrammable area contains various programs that you provide.

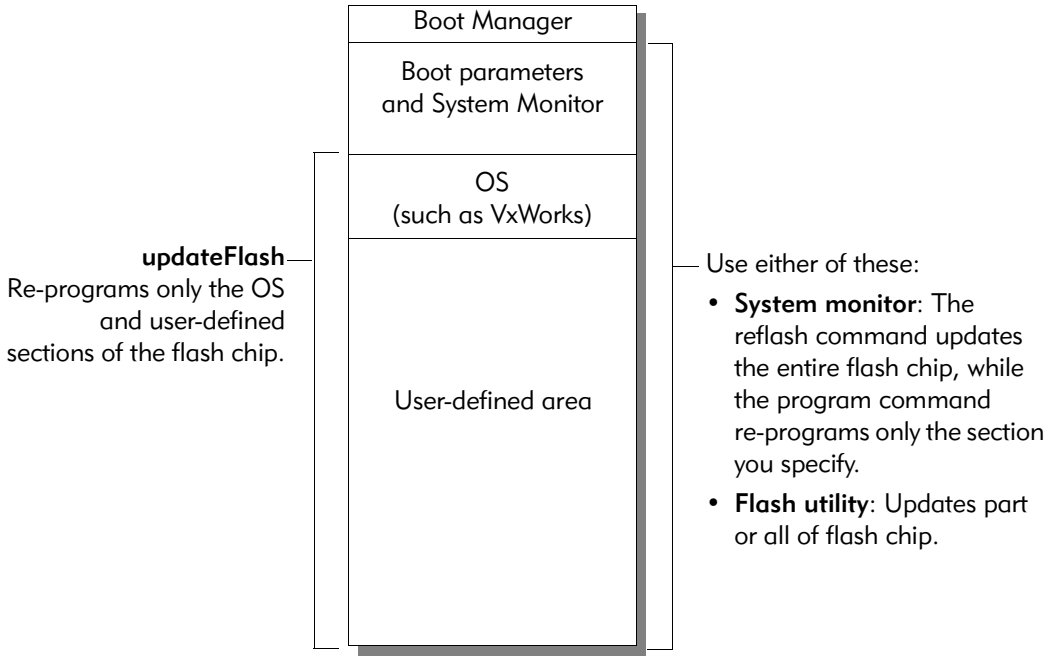
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

About re-programming the flash chip

On rare occasions, part or all of the flash chip contents may require replacement.

Use extreme caution when re-programming the flash chip. The Boot flash parameters and System Monitor blocks rarely change and should not require re-programming.

Figure C-2. Flash chip re-programming coverage



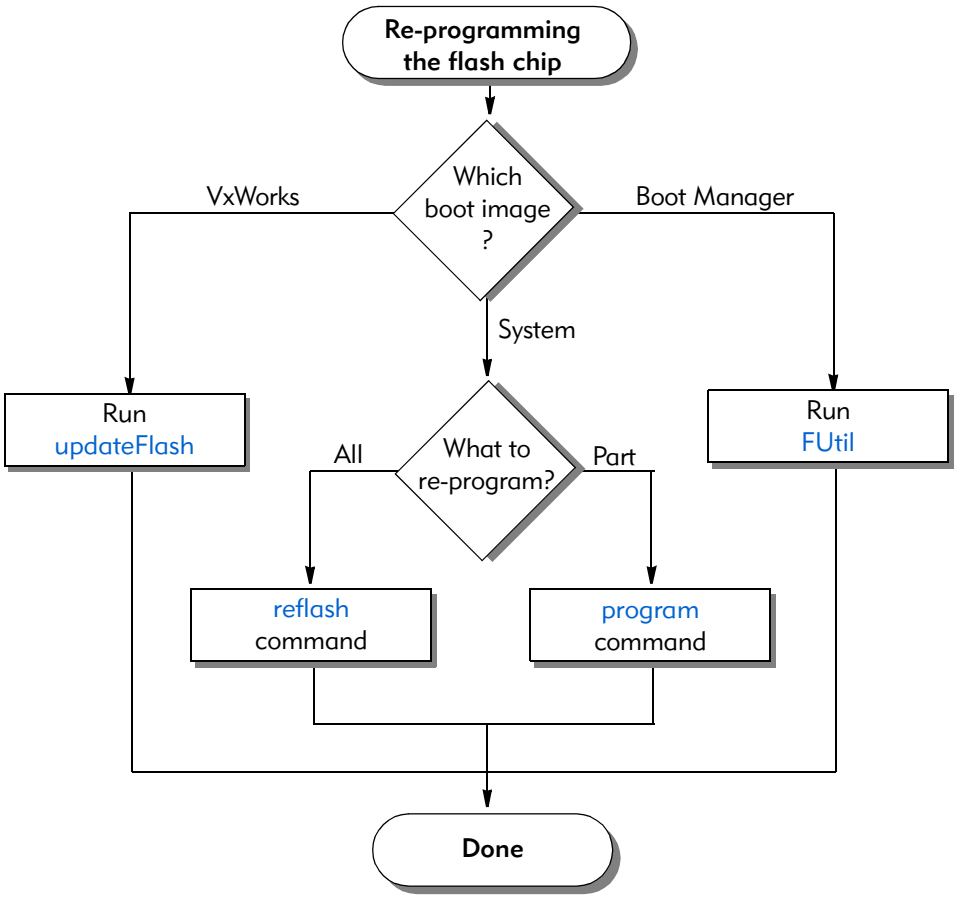
You can set options that re-program the flash chip using any of these:

- **VxWorks**: Re-programs the flash chip using the updateFlash program.
- **System monitor**: Re-programs the flash chip using the System monitor software.
- **Boot Manager**: Re-programs the flash chip using the flash utility program.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

When re-programming the flash chip, follow this process. The rest of this chapter includes detailed instructions for each task:

Figure C-3. Flash chip re-programming process flow



Before you begin

- Ensure that you have the following:
  - Minimum 2 MB of DRAM to run the re-flash program
  - The flash image. You can use the binary image on the ENP SDK CD-ROM. This image includes these flash sections: Boot Manager, Boot flash parameters and System monitor, and OS.
- Decide which portion of your flash chip to re-program:
  - **Partial**: Select this option to re-program only the portions of the flash chip you specify. Your flash chip may require this type of update if specific code or data structures require an update.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

- **Complete (all except the Boot Manager/reflash block)**: Select this option to fix bugs, add new features, and replace corrupted sections of the flash.

  If the Boot Block is corrupt and not executable, return the ENP board to the factory for repair. For information about returning items to RadiSys, see the RadiSys web site. For information about accessing the web site, see *Where to get more information* on page v.

- Decide where you want to intiate the re-programming process from:
  - **VxWorks**: Invoke the updateFlash program.

    This process cannot be performed in a non-VxWorks OS (for example: Windows NT, QNX, or LINUX).

  - **System monitor**: Invoke either the System monitor's program or reflash command.
  - **Boot Manager**: Invoke the futil.exe program.

You are now ready to re-flash your system using the directions for the re-flash method you want to use:

Method	Page
Re-programming the flash chip from VxWorks	356
Re-programming the flash chip from the System monitor	357
Re-programming the flash chip from the Boot Manager	358

Use caution when reprogramming the flash. If your system crashes halfway through, or if you load an invalid image, you must change the board jumpers and use the boot manager's reflash tool to recover your flash image.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Re-programming the flash chip from VxWorks

1. Ensure that your board can boot to VxWorks. For booting information, see *Chapter 5, System Monitor software* or *Appendix A, Configuration jumper settings*.
2. Boot the system to VxWorks.
3. Invoke the updateFlash program by entering this command:

   ```
   updateFlash
   ```

4. Enter flash update parameters as the program prompts for the following (press the Return key to accept the default value):

Prompt	Description
Filename	The directory path and name of the flash image file. When you enter this value, the program immediately opens and reads the file.
Starting offset in flash	The flash address at which the re-write begins. The default is 000000.
Ending offset in flash	The flash address at which the re-write ends. The default is 1C0000.
Starting offset in file	The address in the flash image file at which the read begins. The default is 000000.

   You can abort the update at any time by entering a period ('.').

   After entering the last value, the program updates the flash chip, displaying this message:

   ```
   Progress:
   100% ... done
   ```

   The program asks if you want to re-program other sections of the flash:

   ```
   Do you want to program other offsets (No):
   ```

   If you want to re-program other sections, enter Yes and repeat step 2. Otherwise, press the Return key to accept the default.

5. Reboot the ENP board to use the new flash image.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Re-programming the flash chip from the System monitor

1. Ensure that your board can boot to the System monitor. For booting information, see *Chapter 5, System Monitor software* or *Appendix A, Configuration jumper settings*.
2. Boot the system to the System monitor.

   The command prompt displays:

   ```
   enp>
   ```

3. Enter one of these commands:
   - To reprogram only the part of the flash chip you specify, enter:

     `program FlashAddress FileSize MemoryAddress`

     where:

     `FlashAddress` is the flash address of the block you want to program. The supplied flash address must be the starting address of a flash block and a valid flash address, otherwise the program command aborts.

     `FileSize` is the image size you want to program.

     `MemoryAddress` is the memory address of the image you want to program.
   - To reprogram the entire flash chip, except the Boot Manager section, enter:

     `reflash MemoryAddress`

     where:

     `MemoryAddress` is the memory address of the image you want to program.

   For detailed information about these commands, see *Chapter 5, System Monitor software*.
4. Reboot the ENP board to use the new flash image.

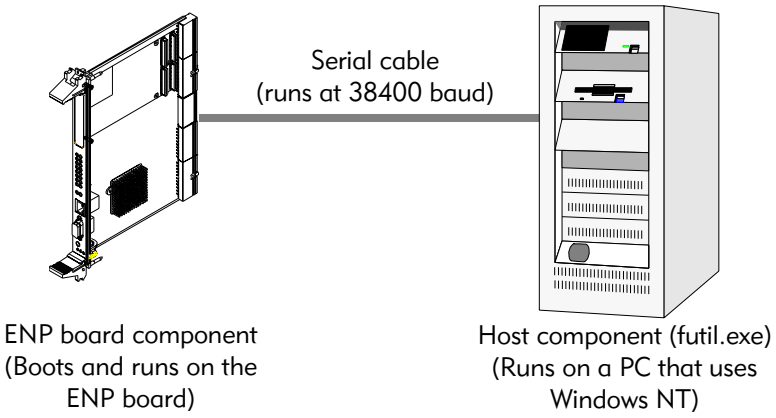
Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Re-programming the flash chip from the Boot Manager

About the Flash utility setup and enumeration

The Flash utility consists of these parts:

Figure C-4. Flash utility components



You enter commands on the Host component, which runs on the PC. The Host component sends those commands to the ENP board component, which then modifies the flash. The ENP board component returns messages that tell you the changes are complete.

Reprogramming the flash

1. Ensure that your board can boot to the Boot Manager. For booting information, see *Chapter 5, System Monitor software* or *Appendix A, Configuration jumper settings*.
2. Turn on the power or reset the ENP board.

   The Boot Manager displays this message:

   ```
   Press space bar to stop auto-boot...
   10
   ```

3. Press the space bar to remain in the Boot manager.

   The Boot Manager prompt displays:

   ```
   [BootMgr]:
   ```

4. Invoke the Flash utility by entering this command:

   ```
   b 0
   ```

   The Flash utility displays this message:

   ```
   FlashUtil, Version 1.1
   Waiting for requests from host....
   The program will then wait for the host program to run.
   ```

5. Go to the \bin directory in the ENP install directory, and run futil.exe.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The program displays this message, then prompts you to enter flash update parameters.

```
Numeric parameters (other than the comm port) should be entered as
    hexadecimal numbers with out the leading '0x' .
Enter a blank line to accept the default value.
Enter a period ('.') to abort the program.
```

6. Enter flash update parameters as the program prompts for the following (press the Return key to accept the default value):

Prompt	Description

Comm port	The port the serial cable is attached to. The default is 2.
	After entering this value, the program contacts the ENP board, checks for flash chip information, and displays information similar to this:
	`FlashUtil, Version 1.1`
	`CPU Revision: 6901C121`
	`Flash Width: 32`
	`Flash Bank Size: 4096 KBytes`
	`Flash Type Bank[0]: 28F320C3-B`
	`Flash Type Bank[1]: Unknown`
	`Flash Type Bank[2]: Unknown`
	`Flash Type Bank[3]: Unknown`

Filename	The directory path and name of the flash image file. When you enter this value, the program immediately opens and reads the file. The default is boot1200.bin.
Starting offset in flash	The flash address at which the re-write begins. The default is 000000.
Ending offset in flash	The flash address at which the re-write ends. The default is 1C0000.
Starting offset in file	The address in the flash image file at which the read begins. The default is 000000.
Do you want to re-program the Boot Parameter and System Monitor partition?	Determines whether the program updates the flash chip's Boot Parameter and System Monitor block.Enter one of these values: • **Yes (default)**: Re-programs the Boot Parameter and System Monitor partition. • **No**: Does not re-program the Boot Parameter and System Monitor partition.
Do you want to re-program the VxWorks partition?	Determines whether the program updates the flash chip's VxWorks block. Enter one of these values: • **Yes (default)**: Re-programs the VxWorks partition. • **No**: Does not re-program the VxWorks partition.

You can abort the update at any time by entering a period ('.').



You can enter all the arguments from the command line. For example:

```
C:\Radisys\ENP_SDK_2.0\Bin>FUtil -c 2 -f boot1200.bin
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

After entering the last value, the host portion of the program sends the new image to the board and updates the flash, displaying these messages:

```
Sending data to remote system...
Percent complete: 100
Updating flash...
Percent Complete: 100
Done
```

7. Reboot the ENP board to use the new flash image.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Glossary

µPI	Microcode Programming Interface.
AAPI	ARM Application Programming Interface.
AERM	Alignment Error Rate Monitor.
ANSI	(American National Standards Institute) An organization dedicated to advancement of national standards related to product manufacturing.
Bellcore	(Bell Communications Research) Formed in 1984 when AT&T divested. Provides centralized research and development services to the divested AT&T companies, such as standards, procedures, and software development.
BIB	(Backward Indicator Bit) A bit in the header of an SS7 signal unit toggled to indicate a negative acknowledgement by a remote signaling point.
BSN	(Backward Sequence Number) A field in the header of an SS7 signal unit used to acknowledge received signal units by the remote signaling point. It contains the sequence number of the signal unit being acknowledged.
Buffer descriptor	A data structure in SRAM that describes where the actual buffer resides in SDRAM. May also contain other packet specific information.
Changeover	A procedure within MTP3 to ensure that unacknowledged signaling traffic on an unavailable signaling link is retrieved back from MTP2 and retransmitted on an alternate signaling link.
DSP	Digital Signal Processor.
ETSI	(European Telecommunications Standards Institute) Dedicated to the development of telecommunications standards for users, manufacturers, suppliers, administrations, and PTTs. ETSI is the European counterpart of ANSI.
FBI	To be developed.
Firmware	Software kept in semipermanent memory. Firmware is non-volatile; it is not "forgotten" when power turns off. Firmware is typically burned in, or stored, on chips such as PROMs and EPROMs. Software so constantly called upon by a computer or phone system is the typical candidate for firmware.
FIB	(Forward Indicator Bit) A bit in the header of an SS7 signal unit that is toggled when retransmitting forward messages that have been negatively acknowledged by a remote signaling unit.
FISU	Fill-In Signal Unit.
FPGA	Field Programmable Gate Array.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

FSN	(Forward Sequence Number) A field in the header of an SS7 signal unit that contains the sequence number of the signal unit in which it is carried. It is used by the remote signaling point to ensure that all signal units are received in order.
FSNC	(Forward Sequence Number for Changeover) Indicates the value of the foward sequence number of last message signal unit accepted by remote MTP2. This value is used during the MTP3 changeover procedure.
Host	The operating environment—the chassis and operating system—in which the ARTIC 1107/2107 resides.
HSL	High Speed Link or High Speed Signaling Link.
IOP	Input/Output Processor.
IPMI	Intelligent Platform Management Interface.
IPv4	Internet Protocol, version 4.
IPv6	Internet Protocol, version 6.
ITU-T	(International Telecommunications Union Telecommunications Standardization Sector) A global organization that defines and adopts telecommunications standards, regulates the use of the readio frequency spectrum, and furthers telecommunications development.
IX	Internet Exchange.
IXA	Internet Exchange Architecture.
IXP	Internet Exchange Processor.
LAN	(Local Area Network) A short-distance data communications network, typically within a building or campus, used to link computers and peripheral devices under some form of standard control.
Link	The physical and electrical connection provided by MTP1 that directly connects a pair of signaling end points.
longword	32bit wide access.
LSSU	(Link Status Signal Unit) A packet sent between MTPs to provide SS7 information about the sending node and its links. This information is sent during initial alignment of the links, when there is an associated processor outage, or when link congestion is detected.
MAC	Media Access Controller.
MacPort	Shorthand for referring to an IX Bus device consisting of a MAC number and a port number.
Mpkt	The amount of data transferred in an IXP1200 FIFO element read or write (8 quad words or 64 bytes).
MSU	(Message Signal Unit) The real messages in the SS7 network which carry SS7 user part data (call control, database queries, network management and maintenance data) between signaling points.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MTP	(Message Transfer Part) The message transfer part provides the functions that enables the SS7 user part to transfer data across the SS7 network. The message transfer part is divided into 3 components: MTP1, MTP2, and MTP3.
MTP1	(Message Transfer Part Level 1) The OSI physical layer which defines the physical, electrical and functional characteristics of the digital signaling link. Interfaces defined include E-1, DS-1, V.35, DS-0, and DS-0A. Defined by ITU-T Q.702.
MTP2	(Message Transfer Part Level 2) The OSI data link layer that provides end-to-end message transmission across an individual signaling link. Defined by ITU-T Q.703.
MTP3	(Message Transfer Part Level 3) The OSI network layer which provides message routing between signaling points in the SS7 network. Defined by ITU-T Q.704.
OSI	(Open Systems Interconnection) An internationally accepted framework of standards for communication between different systems made by different vendors. ISO's goal is to create an open systems networking environment where any vendor's computer system, connected to any network, can freely share data with any other computer system on that network or a linked network.
Packet	An Ethernet or other communications protocol defined packet.
Packet buffer	The physical SDRAM buffer that contains packet data.
PCI	Peripheral Component Interconnect.
PCR	(Preventive Cyclic Retransmission) A form of error correction whereby previously transmitted, but as yet unacknowledged signal units are cyclically retransmitted in place of FISUs when no new signal units are available for transmission.
Pipe	A channelized T1 or E1 line.
PMC	(PCI Mezzanine Card) A standard form factor for mezzanine cards defined by IEEE P1386 and IEEE P1386.1.
Port	The physical or electrical interface through which one gains access. A point in the computer or telephone system where data may be accessed.
Primitive	Commands and their respective responses associated with services requested of the MTP.
PSTN	(Public Switched Telephone Network) The worldwide voice telephone network accessible to all those with telephones and access privileges.
RPQ	Used for the entire ARTIC family of products, not specific to SS7. This term prefaces many type definitions.
RTC	Real-Time Clock.
RTP	Real-Time Protocol.
SDL	(Signaling Data Link) Refers to either MTP2 or HSL.
SIB	Status Indication "B" (busy).
SIE	Status Indication "E" (emergency alignment).
SIF	(Signaling Information Field) The data "payload" within an MSU.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SIN	Status Indication "N" (normal alignment).
SIO	Status Indication "O" (out of alignment). Also, Service Information Octet, a field in an MSU used to associate signaling information with a particular SS7 user part.
SIOS	Status Indication "OS" (out of service).
SIPO	Status Indication "PO" (emergency alignment).
SNID	(Sub-network ID) A unique identifier that references a connection or link.
SS7	(Signaling System Number 7) A global standard defined by the ITU-T that defines the procedures and protocol to effect wireless and wireline call setup, routing and control in the PSTN.
SU	(Signal Unit) The equivalent of an individual message or frame in the SS7 network. Signal unit types include MSU, FISU, and LSSU.
SUERM	Signal Unit Error Rate Monitor.
TDM	Time Division Multiplex.
TDM2IX	The ENP board chip which provides TDM to IXBus conversion.
UDP	User Datagram Protocol.
WAN	(Wide Area Network) A data network typically extending a LAN outside the building, over telephone common carrier lines to link to other LANs in remote buildings in possibly remote cities. Also refers to the Wide Area Network firmware.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Index

N A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Numerics

21555 test *327*

A

ANSI, defined *361*
applications, sample
    ENP2506FwdDemo *28*
    ENP2506LoopbackTest *29*
    ForwardingDemo *23*
    IXF1002Ingress/Egress *27*
    TdmDemo *30*

B

b (boot) command *48*
Boot Manager
    commands
        b (boot) *48*
        c (change) *48*
        h (help) *49*
        p (print) *49*
Boot manager
    command list *48*

C

c (change) command *48*
clock configuration example *81*
commands
    Boot Manager, list *48*
    System Monitor, list *36*
connector, loopback *348*
conventions, notational *iv*
cpld test *328*

D

delete command *37*
Demo applications
    ENP2506FwdDemo *28*
    ENP2506LoopbackTest *29*
    Forwarding *23*
    IXF1002Ingress/Egress *27*
    TDM *30*
diagnostic tests
    21555 *327*
    cpld *328*
    e1t1 *330*
    ethernet *331*
    flash *333*
    h110 *334*
    ipmi *335*, *341*
    ixf1002 *338*
    ixf440 *336*
    macaddrs *340*
    scratch *342*
    sdram *343*
    sram *344*
    t8105 *345*
    tdm2ix *346*
    teeprom *347*

E

E1/T1 functions
    E1 and T1
        T1E1clockConfig *114*
        T1E1getBoardConfig *115*
        T1E1initCard *113*
        T1E1setLeds *116*
    E1 only
        getE1Signaling *128*
        setE1Config *123*
        setE1Signaling *124*
        setE1SignalingHandler *131*
    T1 only
        getT1Signaling *125*
        getT1SignalingRaw *127*
        getT1Status *126*
        setT1ChannelConfig *122*
        setT1ClearChannels *120*
        setT1Command *119*
        setT1Config *117*
        setT1IdleChannels *121*
        setT1Signaling *118*
        setT1SignalingHandler *129*
        setT1StatusHandler *130*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

E1/T1 library
function list 109
sample startup sequence 106
structures 132
E1/T1 structures
E1 only
E1SignalingHandler 167
t_E1_line_buildout 163
t_E1_line_coding 161
t_E1_signaling_mode 162
t_E1_user_config_struct 164
t_E1_user_signaling_data 166
T1 and E1
t_T1E1_BoardConfig 138
t_T1E1_card_type 135
t_T1E1_clock_mode 139
t_T1E1_ClockConfig 145
t_T1E1_framer_id 134
t_T1E1_int_rcv_sys_clk 142
t_T1E1_int_xmit_sys_clk 144
t_T1E1_led_state 136
t_T1E1_rclock_rclk 141
t_T1E1_rclock_ref 140
t_T1E1_user_signaling_data 137
t_T1E1_xclock_DCOX 143
T1 only
t_T1_framing_mode 147
t_T1_line_buildout 148
t_T1_line_coding 146
t_T1_signaling_data 151
t_T1_user_channel_config 156
t_T1_user_clear_channel_data 154
t_T1_user_command_data 152
t_T1_user_config_struct 149
t_T1_user_idle_struct 155
t_T1_user_raw_signaling_struct 158
t_T1_user_signaling_data 151
t_T1_user_status_struct 153
T1SignalingHandler 160
T1StatusHandler 159
E1SignalingHandler 167
e1t1 test 330
e-mail address, RadiSys v
ENP microcode resource manager
ENP2505_InitApp 186
ENP2506_InitApp 187
ENP3511_InitApp 188, 189
ENPCreateEpktStack 190
ENPDestroyEpktStack 191
ENPInitThreadID 192
function list 180
interrupt function list 175
ENP SDK, requirements 7
ENP-2505 v
ENP-2505 sample application 23
ENP2505_InitApp 186
ENP-2506 vi
ENP-2506 sample application 27, 28, 29
ENP2506_InitApp 187
ENP2506FwdDemo demo 28
ENP2506LoopbackTest demo 29
ENP-3511 vi
ENP-3511 sample application 30
ENP3511_InitApp 188, 189
ENPClosePCIChannel 71
ENPCreateEpktStack 190
ENPDestroyEpktStack 191
ENPInitThreadID 192
ENPOpenPCIChannel 72
ENPPCIAPIPrintError 73
ENPReadFromPCIChannel 74
ENPWriteToPCIChannel 75
ethernet test 331
examples
clock configuration 81
time slot switching 84

F

flash chip, re-programming
described 352
from VxWorks 356
using the Boot Manager 358
using the System Monitor 357
flash test 333
Forwarding demo 23
functions
E1/T1 function list 109
ENP microcode resource manager interrupts, list 175
ENP microcode resource manager, list 180
IXF1002 interrupt list 176
IXF1002 list 265
IXF440 interrupt list 175
IXF440 list 222
IXP1200 list 201
PCI communications, list 55
TDM interrupt list 175
TDM list 86
TDM2IX list 176, 300
futil.exe 353, 354

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

G

getE1Signaling *128*
getT1Signaling *125*
getT1SignalingRaw *127*
getT1Status *126*
glossary *361*

H

h (help) command *49*
h110 test *334*
help *v*

I

Ingress/Egress demo *27*
installing the software
  before you begin *7*
  running the Setup program *8*
  uninstalling *8*
interrupt library
  functions
    VxWorks
      ENPClosePCIChannel *71*
      ENPOpenPCIChannel *72*
      ENPPCIAPIPrintError *73*
      ENPReadFromPCIChannel *74*
      ENPWriteToPCIChannel *75*
    WindowsNT
      IXPHPCAsyncOpen *56*
      IXPHPCAsyncRead *58*
      IXPHPCAsyncWrite *60*
      IXPHPCClose *62*
      IXPHPCEnumerate *63*
      IXPHPCReset *65*
      IXPHPCSyncOpen *66*
      IXPHPCSyncRead *68*
      IXPHPCSyncWrite *69*
ipmi command *38*
ipmi test *335* , *341*
IXF1002
  Ethernet controller test *338*
  library, function list *265*
  library, interrupt function list *176*
IXF1002 microcode library
  functions
    IXF1002ClearNetStats *267*
    IXF1002ClosePort *268*
    IXF1002DefineTxQueue *269*
    IXF1002DisableAllInterrupt *270*
    IXF1002DisableInterrupt *271*
    IXF1002EnableAllInterrupt *272*
    IXF1002EnableInterrupt *273*
    IXF1002GetRxTxMode *274*
    IXF1002GetTxEpktCoung *275*
    IXF1002GetTxMpktCount *276*
    IXF1002HookPort *277*
    IXF1002OpenPort *278*
    IXF1002QueryNetStats *279*
    IXF1002ReadNetStatReg *280*
    IXF1002ReadPacket *281*
    IXF1002ResetPort *283*
    IXF1002SetGMIIMode *284*
    IXF1002SetGPCSMode *285*
    IXF1002SetInterruptHandler *286*
    IXF1002StartPort *287*
    IXF1002StopPort *288*
    IXF1002UnhookPort *289*
    IXF1002WritePacket *290*
    IXF1002WriteReg *291*
    IXF1002xReadReg *282*
ixf1002 test *338*
IXF1002ClearNetStats *267*
IXF1002ClosePort *268*
IXF1002DefineTxQueue *269*
IXF1002DisableAllInterrupt *270*
IXF1002DisableInterrupt *271*
IXF1002EnableAllInterrupt *272*
IXF1002EnableInterrupt *273*
IXF1002GetRxTxMode *274*
IXF1002GetTxEpktCount *275*
IXF1002GetTxMpktCount *276*
IXF1002HookPort *277*
IXF1002Ingress/Egress demo *27*
IXF1002OpenPort *278*
IXF1002QueryNetStats *279*
IXF1002ReadNetStatReg *280*
IXF1002ReadPacket *281*
IXF1002ReadReg *282*
IXF1002ResetPort *283*
IXF1002SetGMIIMode *284*
IXF1002SetGPCSMode *285*
IXF1002SetInterruptHandler *286*
IXF1002StartPort *287*
IXF1002StopPort *288*
IXF1002UnhookPort *289*
IXF1002WritePacket *290*
IXF1002WriteReg *291*
IXF440 *219*
  Ethernet controller test *336*
  library, function list *222*
  library, interrupt function list *175*
  MAC address test *340*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IXF440 microcode library
functions
IXF440ClearNetStats 223
IXF440ClosePort 224
IXF440GetDuplexMode 229
IXF440GetRxError 230
IXF440GetRxTxMode 231
IXF440GetTxError 232
IXF440HookPort 233
IXF440OpenPort 234
IXF440QueryNetStats 235
IXF440ReadPacket 236
IXF440ReadReg 237
IXF440ResetPort 238
IXF440SetDuplexMode 239
IXF440SetRxTxMode 241 , 242
IXF440UnhookPort 243
IXF440WritePacket 244
IXF440WriteReg 245
IXF440 uPI 219
receive thread 220
transmit thread 220
overview 213
queues
IXF440 receive queue 219
IXF440 transmit queue 219
structures
Epkt descriptor 218
IXF440 receive queue 219
ixf440 test 336
IXF440 transmit queue 219
IXF440 uPI 219
receive thread 220
transmit thread 220
IXF440ClearNetStats 223
IXF440ClosePort 224
IXF440GetDuplexMode 229
IXF440GetRxError 230
IXF440GetRxTxMode 231
IXF440GetTxError 232
IXF440HookPort 233
IXF440OpenPort 234
IXF440QueryNetStats 235
IXF440ReadPacket 236
IXF440ReadReg 237
IXF440ResetPort 238
IXF440SetDuplexMode 239
IXF440SetRxTxMode 241 , 242
IXF440UnhookPort 243
IXF440WritePacket 244
IXF440WriteReg 245
IXP1200 library
function list 201
IXPPopSramStack 204
IXPPushSramStack 203
IXPReadLockSramA32 205
IXPReadReg 206
IXPReadSramA32 208
IXPWriteReg 209
IXPWriteSramA32 211
IXPWriteUnlockSramA32 212
IXPHPCAsyncOpen 56
IXPHPCAsyncRead 58
IXPHPCAsyncWrite 60
IXPHPCClose 62
IXPHPCEnumerate 63
IXPHPCReset 65
IXPHPCSyncOpen 66
IXPHPCSyncRead 68
IXPHPCSyncWrite 69
IXPPopSramStack 204
IXPPushSramStack 203
IXPReadLockSramA32 205
IXPReadReg 206
IXPReadSramA32 208
IXPWriteReg 209
IXPWriteSramA32 211
IXPWriteUnlockSramA32 212

L

load command 39
loopback connector 348

M

MAC address test 340
macaddrs test 340
Manager, Boot commands
b (boot) 48
c (change) 48
h (help) 49
p (print) 49
Monitor, System commands
delete 37
ipmi 38
load 39
peek 40
poke 41
program 42
reboot 43
reflash 44
set 45
show 46

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

N

notational conventions *iv*

P

p (print) command *49*
PCI communications
    function list *55*
peek command *40*
poke command *41*
program command *42*

Q

queues
    IXF440 receive queue *219*
    IXF440 transmit queue *219*

R

RadiSys, contacting *v*
README file *v*
reboot command *43*
receive queue *219*
References *vi*
reflash command *44*
related products
    ENP-2505 *v*
    ENP-2506 *vi*
    ENP-3511 *vi*
re-programming the flash chip
    explained *353*
    from VxWorks *356*
    using the Boot Manager *358*
    using the System Monitor *357*
requirements for installing ENP SDK *7*

S

Sample applications
    ENP2506FwdDemo demo *28*
    ENP2506LoopbackTest demo *29*
    Forwarding demo *23*
    IXF1002Ingress/Egress demo *27*
    TDM demo *30*
samples
    clock configuration *81*
    time slot switching *84*
scratch test *342*
sdram test *343*
set command *45*
setE1Config *123*
setE1Signaling *124*
setE1SignalingHandler *131*
setT1ChannelConfig *122*
setT1ClearChannels *120*
setT1Command *119*
setT1Config *117*
setT1IdleChannels *121*
setT1Signaling *118*
setT1SignalingHandler *129*
setT1StatusHandler *130*
Setup program, running *8*
show command *46*
software
    specifications for ENP-2505 *3*
sram test *344*
startup sequence, E1/T1 *106*
structures
    E1/T1 *132*
    E1SignalingHandler *167*
    t_E1_line_buildout *163*
    t_E1_line_coding *161*
    t_E1_signaling_mode *162*
    t_E1_user_config_struct *164*
    t_E1_user_signaling_data *166*
    t_T1_framing_mode *147*
    t_T1_line_buildout *148*
    t_T1_line_coding *146*
    t_T1_signaling_data *151*
    t_T1_user_channel_config *156*
    t_T1_user_clear_channel_data *154*
    t_T1_user_command_data *152*
    t_T1_user_config_struct *149*
    t_T1_user_idle_struct *155*
    t_T1_user_raw_signaling_struct *158*
    t_T1_user_signaling_data *151*
    t_T1_user_status_struct *153*
    t_T1E1_BoardConfig *138*
    t_T1E1_card_type *135*
    t_T1E1_clock_mode *139*
    t_T1E1_ClockConfig *145*
    t_T1E1_framer_id *134*
    t_T1E1_int_rcv_sys_clk *142*
    t_T1E1_int_xmit_sys_clk *144*
    t_T1E1_led_state *136*
    t_T1E1_rclock_rclk *141*
    t_T1E1_rclock_ref *140*
    t_T1E1_user_signaling_data *137*
    t_T1E1_xclock_DCOX *143*
    T1SignalingHandler *160*
    T1StatusHandler *159*
    TDM list *86*
support *v*
System Monitor
    command list *36*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

commands
delete 37
ipmi 38
load 39
peek 40
poke 41
program 42
reboot 43
reflash 44
set 45
show 46

T

t_E1_line_buildout 163
t_E1_line_coding 161
t_E1_signaling_mode 162
t_E1_user_config_struct 164
t_E1_user_signaling_data 166
t_fallback_clk 100
t_netref_clk 101
t_ref_clk 102
t_source_dest 103
t_stream_rate 104
t_T1_framing_mode 147
t_T1_line_buildout 148
t_T1_line_coding 146
t_T1_signaling_data 151
t_T1_user_channel_config 156
t_T1_user_clear_channel_data 154
t_T1_user_command_data 152
t_T1_user_config_struct 149
t_T1_user_idle_struct 155
t_T1_user_raw_signaling_struct 158
t_T1_user_signaling_data 151
t_T1_user_status_struct 153
t_T1E1_BoardConfig 138
t_T1E1_card_type 135
t_T1E1_clock_mode 139
t_T1E1_ClockConfig 145
t_T1E1_framer_id 134
t_T1E1_int_rcv_sys_clk 142
t_T1E1_int_xmit_sys_clk 144
t_T1E1_led_state 136
t_T1E1_rclock_rclk 141
t_T1E1_rclock_ref 140
t_T1E1_user_signaling_data 137
t_T1E1_xclock_DCOX 143
t_TDMClockConfig 89
t_TDMConnection 91
t_TDMStreamConfig 98
t_TDMSwitchConfig 99
T1E1clockConfig 114
T1E1getBoardConfig 115
T1E1initCard 113
T1E1setLeds 116
T1SignalingHandler 160
T1StatusHandler 159
t8105 test 345
TDM demo 30
TDM routing and switching library
architecture 78
concepts 78
examples
clock configuration 81
time slot switching 84
function list 86
functions
TDMClearClockFault 87
TDMClearMemoryFault 88
TDMGetErrorStatus 92
TDMInitSwitch 93
TDMSetClockConfig 94
TDMSetStreamConfig 96
TDMSetSwitchConfig 97
interrupt function list 175
overview 77–79
structure list 86
structures
t_fallback_clk 100
t_netref_clk 101
t_ref_clk 102
t_source_dest 103
t_stream_rate 104
t_TDMClockConfig 89
t_TDMConnection 91
t_TDMStreamConfig 98
t_TDMSwitchConfig 99
TDM2IX microcode library
function list 176, 300
functions
TDM2IXClosePort 301
TDM2IXGetRxError 306
TDM2IXGetTxError 307
TDM2IXHookPort 308
TDM2IXOpenPort 309
TDM2IXReadPacket 310
TDM2IXReadReg 311
TDM2IXResetPort 312
TDM2IXSetRxLength 314
TDM2IXSetTxLength 315
TDM2IXStartPort 316
TDM2IXStopPort 317
TDM2IXUnhookPort 318
TDM2IXWritePacket 319
TDM2IXWriteReg 320
tdm2ix test 346

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TDM2IXClosePort *301*
TDM2IXGetRxError *306*
TDM2IXGetTxError *307*
TDM2IXHookPort *308*
TDM2IXOpenPort *309*
TDM2IXReadPacket *310*
TDM2IXReadReg *311*
TDM2IXResetPort *312*
TDM2IXSetRxLength *314*
TDM2IXSetTxLength *315*
TDM2IXStartPort *316*
TDM2IXStopPort *317*
TDM2IXUnhookPort *318*
TDM2IXWritePacket *319*
TDM2IXWriteReg *320*
TDMClearClockFault *87*
TDMClearMemoryFault *88*
TDMGetErrorStatus *92*
TDMInitSwitch *93*
TDMSetClockConfig *94*
TDMSetStreamConfig *96*
TDMSetSwitchConfig *97*
technical support *v*
teeprom test *347*
tests
  21555 *327*
  cpld *328*
  e1t1 *330*
  ethernet *331*
  flash *333*
  h110 *334*
  ipmi *335*, *341*
  ixf1002 *338*
  ixf440 *336*
  macaddrs *340*
  scratch *342*
  sdram *343*
  sram *344*
  t8105 *345*
  tdm2ix *346*
  teeprom *347*
time slot switching example *84*
transmit queue *219*
troubleshooting *v*

U

uninstalling INtime software *8*
uPI, IXF440 *219*
uPI, IXF440 receive thread *220*
uPI, IXF440 transmit thread *220*
URLs, RadiSys *v*

W

World-Wide Web URLs, RadiSys *v*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP