

PTP NT

Programmable Telemetry Processor
for Windows NT

*User's Manual
Version 1.40
November 1998*

TM 98-05-01



PTP NT

Programmable Telemetry Processor

for Windows NT

User's Manual

Copyright © 1998 Avtec Systems, Inc. All rights reserved. No part of this document may be reproduced without prior written permission from Avtec Systems.

The information in this document has been fully reviewed and is believed to be entirely reliable. AVTEC reserves the right, however, to modify any products herein to improve reliability, function or design. AVTEC does not assume any liability arising out of the application or use of any product or circuit described herein. AVTEC does not convey any license under its patent rights or the rights of others.

Support

Technical Support

This manual provides engineers with information necessary to operate the Programmable Telemetry Processor Software. Technical Support is available from Avtec Systems, Inc.

Avtec Systems, Inc.
Attn: PTP NT Technical Support
10530 Rosehaven St., Suite 300
Fairfax, VA 22030-2840
(703) 273-2211
t1m@avtec.com

Customer Service

Direct non-technical questions and business-related matters to Customer Support at the above number.

TABLE OF CONTENTS

SCOPE	XV
HOW TO USE THIS MANUAL	XV
RELATED DOCUMENTATION.....	XVI
CHAPTER 1.....	1-1
INTRODUCTION	1-1
PTP OVERVIEW	1-1
Section 1: PTP Architecture.....	1-1
Section 2: Theory of Operation.....	1-3
Section 3: Applications.....	1-6
CHAPTER 2.....	2-1
INSTALLING AND OPERATING THE PTP	2-1
Section 1: Installing PTP NT Software	2-1
Section 2: Configuring the PTP NT System	2-2
Section 3: Operating the PTP	2-13
Section 2: Execution Modes	2-14
Section 3: PTP NT Server	2-14
Section 4: PTP NT Console.....	2-16
CHAPTER 3.....	3-1
WORKING WITH MODULES AND DESKTOPS	3-1
Section 1: Creating a Desktop.....	3-2
Section 2: Saving and Loading a Desktop	3-8
Section 3: Other Desktop Functions	3-10
CHAPTER 4.....	4-1
PTP MODULES	4-1
DATA I/O MODULES	4-5
Section 1: Avtec Rate Adjust Module	4-5
Section 2: AVTEC Serial Input Module	4-10
Section 3: AVTEC Serial Output Module.....	4-25
Section 4: Com Port Transceiver Module	4-30
Section 5: CPU Timer Module	4-32
Section 6: Dump to Line Printer Module	4-34
Section 7: File Playback Module	4-36
Section 8: File Recorder Module	4-38
Section 9: File Spooler Module.....	4-43
Section 10: Network Sockets Module	4-46
Section 11: Network Sockets Ver 2 Module	4-48
AUXILLIARY I/O MODULES.....	4-50
Section 12: Apogee PSK Modulator Module (ApogeePSK).....	4-50
Section 13: Apogee Time Code Generator Module (Time).....	4-52
Section 14: Aydin Bit Sync Module (AydinBS).....	4-55
Section 15: Aydin PSK Demodulator Module (Aydin PSKDemod).....	4-57
Section 16: GDP PSK Modulator Module (GDP PSKMod)	4-60
Section 17: Microdyne PCR-2000 Receiver/Demodulator Module (PCR2000)	4-62
Section 18: National Instruments GPIB Interface Module (GPIB).....	4-66
Section 19: Odetics Time Module	4-69
Section 20: Veda Bit Synchronizer Module (VedaBitSync).....	4-73
DATA PROCESSING MODULES	4-76
Section 21: 8 Bit Sorter Module.....	4-76

Section 22: 16 Bit Sorter Module	4-79
Section 23: Best Source Select Module (BSS)	4-82
Section 24: Bit Density Correction Module	4-85
Section 25: Bit Error Rate Tester Module	4-86
Section 26: CCSDS Virtual Channel Processor Module	4-89
Section 27: CCSDS Virtual Channel Simulator Module	4-94
Section 28: Error Inject Module	4-99
Section 29: Gather Scatter Module	4-100
Section 30: In-Line UNZip Module	4-101
Section 31: In-Line Zip Module	4-102
Section 32: Null Transceiver Module	4-103
Section 33: Packet Processor Module	4-105
Section 34: TDM Simulator Module (TDMSim)	4-108
Section 35: Unknown Module	4-111
ENCAPSULATION/EXTRACTION MODULES	4-112
Section 36: ACE SFDU Formatter Module	4-112
Section 37: Ames Command Encapsulator Module	4-115
Section 38: AXAF SFDU Formatter Module	4-117
Section 39: DST SFDU Formatter Module	4-120
Section 40: EDOS Service Header Module	4-123
Section 41: IPDU Formatter Module	4-125
Section 42: IPDU Receiver Module	4-127
Section 43: LEO-T CDH Formatter Module	4-130
Section 44: LEO-T CDH Receiver Module	4-132
Section 45: LEO-T TFDH Transmitter Module	4-135
Section 46: NASCOM Blocker Module	4-137
Section 47: NASCOM Deblocker Module	4-140
Section 48: NASCOM RTP Formatter Module	4-142
Section 49: NASCOM RTP Receiver Module	4-145
CHAPTER 5	5-1
PRACTICAL EXAMPLES	5-1
Section 1 - Introduction	5-1
Section 2 - Building Desktops	5-1
Section 3 - Script Files	5-32
PTP NT SERVER FUNCTIONS	A-1
PTP NT Server Commands	A-1
Creating PTP NT Server Script Files using PTP NT Server Commands	A-5
Securing Control Sockets and TCP Server Sockets in PTP NT	A-6
IPDU HEADER SUMMARY	B-1
ACRONYMS AND ABBREVIATIONS	C-1

FIGURES

FIGURE 1-1: PC-BASED FRONT-END PROCESSOR BLOCK DIAGRAM	1-2
FIGURE 1-2: PTP FRONT-END PROCESSOR OPERATIONAL ENVIRONMENT	1-7
FIGURE 1-3: PTP MODULE CONNECTIONS FOR A FRONT-END PROCESSOR	1-8
FIGURE 2-1: AUXIO_CONFIG MAIN WINDOW.....	2-4
FIGURE 2-2: AUXIO_CONFIG CHANNEL CONFIGURATION DIALOG.....	2-5
FIGURE 2-3: FSTS_CONFIG MAIN WINDOW.....	2-7
FIGURE 2-4: FSTS_CONFIG CHANNEL CONFIGURATION DIALOG.....	2-8
FIGURE 2-5: PTP NT DISPLAY BEFORE LICENSE REGISTRATION.....	2-11
FIGURE 2-6: PTP NT DISPLAY DURING LICENSE REGISTRATION.....	2-12
FIGURE 2-7: PTP NT DISPLAY AFTER LICENSE REGISTRATION.....	2-12
FIGURE 2-8: PTP CLIENT-SERVER COMMUNICATION LINK.....	2-13
FIGURE 2-9: PTP NT SERVER WINDOW.....	2-15
FIGURE 2-10: CONSOLE WINDOW AT STARTUP.....	2-17
FIGURE 2-11: SYSTEM PULL DOWN MENU.....	2-17
FIGURE 2-12: NAME RESOLUTION SUB-MENU.....	2-18
FIGURE 2-13: CONNECT TO WINDOW.....	2-19
FIGURE 2-14: CONSOLE WINDOW AFTER CONNECTING TO A PTP SERVER.....	2-20
FIGURE 2-15: REMOTE DESKTOP PULL DOWN MENU.....	2-21
FIGURE 2-16: SERVER STATUS.....	2-22
FIGURE 2-17: REMOTE MODULE PULL DOWN MENU.....	2-23
FIGURE 3-1: REMOTE DESKTOP PULL DOWN MENU.....	3-2
FIGURE 3-2: BLANK DESKTOP WINDOW.....	3-3
FIGURE 3-3: ADD MODULE DIALOG WINDOW.....	3-3
FIGURE 3-4: CONSOLE WINDOW WITH MODULES LOADED.....	3-6
FIGURE 3-5: REMOTE MODULE PULL DOWN MENU.....	3-7
FIGURE 3-6: EDIT MODULE CONNECTIONS WINDOW.....	3-8
FIGURE 3-7: SAVE DESKTOP WINDOW.....	3-9
FIGURE 3-8: LOAD DESKTOP WINDOW.....	3-9
FIGURE 3-9: DIALOG WINDOW.....	3-10
FIGURE 4-1: EXAMPLE CONSOLE MODULE WINDOW.....	4-3
FIGURE 4-2: AVTEC RATE ADJUST MODULE.....	4-5
FIGURE 4-3: RATE ADJUST CONFIG WINDOW.....	4-5
FIGURE 4-4: AVTEC RATE ADJUST EXTENDED CONFIGURATION WINDOW.....	4-6
FIGURE 4-5: RATE ADJUST STATUS WINDOW.....	4-8
FIGURE 4-6: AVTEC SERIAL INPUT MODULE WINDOW.....	4-10
FIGURE 4-7: CONFIG AVTEC SERIAL INPUT WINDOW.....	4-10
FIGURE 4-8: FRAME SYNCHRONIZER STATE DIAGRAM.....	4-13
FIGURE 4-9: CONFIG AVTEC INPUT EXTENDED OPTIONS WINDOW.....	4-16
FIGURE 4-10: STATUS AVTEC SERIAL INPUT WINDOW.....	4-18
FIGURE 4-11: AVTEC SERIAL OUTPUT MODULE WINDOW.....	4-25
FIGURE 4-12: CONFIG AVTEC SERIAL OUTPUT WINDOW.....	4-26
FIGURE 4-13: CONFIG AVTEC SERIAL OUTPUT EXTENDED OPTIONS WINDOW.....	4-27

FIGURE 4-14: STATUS AVTEC SERIAL OUTPUT WINDOW.....	4-28
FIGURE 4-15: COM PORT TRANSCEIVER MODULE WINDOW.....	4-30
FIGURE 4-16: CONFIG COMPORT TRANSCEIVER WINDOW	4-30
FIGURE 4-17: COMMAND COM PORT TRANSCEIVER WINDOW.....	4-31
FIGURE 4-18: STATUS COM PORT TRANSCEIVER WINDOW	4-31
FIGURE 4-19: CPU TIMER MODULE WINDOW	4-32
FIGURE 4-20: CONFIG CPU TIMER WINDOW	4-32
FIGURE 4-21: STATUS CPU TIMER WINDOW.....	4-33
FIGURE 4-22: DUMP TO LINE PRINTER MODULE.....	4-34
FIGURE 4-23: CONFIG DUMPTOLP WINDOW	4-34
FIGURE 4-24: DUMPTOLP STATUS WINDOW.....	4-35
FIGURE 4-25: FILE PLAYBACK MODULE WINDOW.....	4-36
FIGURE 4-26: CONFIG FILE PLAYBACK WINDOW	4-36
FIGURE 4-27: STATUS FILE PLAYBACK WINDOW	4-37
FIGURE 4-28: FILE RECORDER MODULE WINDOW	4-38
FIGURE 4-29: CONFIG FILE RECORDER WINDOW	4-38
FIGURE 4-30: COMMAND FILE RECORDER WINDOW	4-39
FIGURE 4-31: STATUS FILE RECORDER WINDOW.....	4-40
FIGURE 4-32: FILE RECORDER FORMAT WITH RECORD MUX HEADER ENABLED.	4-41
FIGURE 4-33: FILE RECORDER FORMAT WITH RECORD TIME STAMP ENABLED.....	4-41
FIGURE 4-34: FILE RECORDER FORMAT WITH RECORD STATUS ENABLED	4-41
FIGURE 4-35: FILE RECORDER FORMAT WITH RECORD TIME STAMP AND RECORD STATUS ENABLED.....	4-41
FIGURE 4-36: FILE SPOOLER MODULE	4-43
FIGURE 4-37: FILE SPOOLER COMMAND WINDOW	4-43
FIGURE 4-38: FILE SPOOLER CONFIGURATION WINDOW	4-44
FIGURE 4-39: FILE SPOOLER STATUS WINDOW.....	4-44
FIGURE 4-40: SOCKETS MODULE WINDOW	4-46
FIGURE 4-41: CONFIG SOCKETS WINDOW	4-46
FIGURE 4-42: STATUS SOCKETS WINDOW.....	4-47
FIGURE 4-43: SOCKETS VERSION 2 MODULE.....	4-48
FIGURE 4-44: SOCKETS VERSION 2 CONFIG WINDOW	4-48
FIGURE 4-45: SOCKETS VERSION 2 STATUS WINDOW.....	4-49
FIGURE 4-46: APOGEE PSK MODULATOR MODULE WINDOW.....	4-50
FIGURE 4-47: CONFIG APOGEE PSK MODULATOR WINDOW.....	4-50
FIGURE 4-48: TIME MODULE WINDOW.....	4-52
FIGURE 4-49: CONFIG TIME WINDOW.....	4-52
FIGURE 4-50: STATUS TIME WINDOW	4-54
FIGURE 4-51: AYDIN BIT SYNC MODULE WINDOW	4-55
FIGURE 4-52: CONFIG AYDIN BIT SYNC WINDOW.....	4-55
FIGURE 4-53: STATUS AYDIN BIT SYNC WINDOW	4-56
FIGURE 4-54: PSKDEMOD MODULE WINDOW	4-57
FIGURE 4-55: CONFIG PSKDEMOD WINDOW	4-57
FIGURE 4-56: COMMAND PSKDEMOD WINDOW	4-58
FIGURE 4-57: STATUS PSKDEMOD WINDOW.....	4-59

FIGURE 4-58: PSKMOD MODULE WINDOW	4-60
FIGURE 4-59: CONFIG PSKMOD WINDOW	4-60
FIGURE 4-60: COMMAND GDP PSK MODULATOR WINDOW	4-61
FIGURE 4-61: PCR2000 MODULE WINDOW	4-62
FIGURE 4-62: CONFIG PCR2000 MODULE WINDOW	4-62
FIGURE 4-63: COMMAND PCR2000 MODULE WINDOW	4-63
FIGURE 4-64: STATUS PCR2000 MODULE WINDOW.....	4-64
FIGURE 4-65: GPIB MODULE	4-66
FIGURE 4-66: CONFIG GPIB MODULE WINDOW.....	4-66
FIGURE 4-67: COMMAND GPIB MODULE WINDOW.....	4-68
FIGURE 4-68: ODETICS TIME MODULE	4-69
FIGURE 4-69: CONFIG ODETICS TIME WINDOW.....	4-69
FIGURE 4-70: COMMAND ODETICS TIME BOARD WINDOW	4-69
FIGURE 4-71: ODETICS TIME STATUS WINDOW	4-70
FIGURE 4-72: VEDABITSYNC MODULE WINDOW	4-73
FIGURE 4-73: CONFIG VEDABITSYNC WINDOW.....	4-73
FIGURE 4-74: STATUS VEDABITSYNC WINDOW	4-74
FIGURE 4-75: 8 BIT SORTER MODULE.....	4-76
FIGURE 4-76: 8 BIT SORTER CONFIG WINDOW	4-76
FIGURE 4-77: 8 BIT SORTER STATUS WINDOW.....	4-77
FIGURE 4-78: 16 BIT SORTER MODULE.....	4-79
FIGURE 4-79: 16 BIT SORTER CONFIG WINDOW	4-79
FIGURE 4-80: 16 BIT SORTER STATUS WINDOW.....	4-80
FIGURE 4-81: BSS MODULE.....	4-82
FIGURE 4-82: CONFIG BSS MODULE WINDOW	4-82
FIGURE 4-83: COMMAND BSS MODULE WINDOW	4-83
FIGURE 4-84: BSS STATUS WINDOW.....	4-83
FIGURE 4-85: BIT DENSITY CORRECTION MODULE	4-85
FIGURE 4-86: BIT ERROR RATE TESTER MODULE WINDOW.....	4-86
FIGURE 4-87: CONFIG BIT ERROR RATE TESTER WINDOW.....	4-86
FIGURE 4-88: STATUS BIT ERROR RATE TESTER WINDOW	4-87
FIGURE 4-89: CCSDS VC PROCESSOR MODULE WINDOW	4-89
FIGURE 4-90: CONFIG CCSDS VC PROCESSOR WINDOW	4-89
FIGURE 4-91: CONFIG CCSDS VC PROCESSOR EXTENDED OPTIONS WINDOW	4-91
FIGURE 4-92: STATUS CCSDS VC PROCESSOR WINDOW	4-92
FIGURE 4-93: STATUS CCSDS VC PROCESSOR EXTENDED STATUS WINDOW	4-93
FIGURE 4-94: CCSDS VC SIMULATOR MODULE WINDOW	4-94
FIGURE 4-95: CONFIG CCSDS VC SIMULATOR WINDOW.....	4-94
FIGURE 4-96: CCSDS VC SIMULATOR WINDOW	4-98
FIGURE 4-97: ERROR INJECT MODULE WINDOW	4-99
FIGURE 4-98: COMMAND ERROR INJECT WINDOW	4-99
FIGURE 4-99: GATHER SCATTER MODULE	4-100
FIGURE 4-100: GATHER SCATTER CONFIG WINDOW	4-100
FIGURE 4-101: IN-LINE UNZIP MODULE	4-101

FIGURE 4-102: IN-LINE UNZIP STATUS WINDOW	4-101
FIGURE 4-103: IN-LINE ZIP MODULE	4-102
FIGURE 4-104: IN-LINE ZIP STATUS WINDOW	4-102
FIGURE 4-105: NULL TRANSCEIVER MODULE WINDOW	4-103
FIGURE 4-106: CONFIG NULL TRANSCEIVER WINDOW	4-103
FIGURE 4-107: STATUS NULL TRANSCEIVER WINDOW	4-104
FIGURE 4-108: PACKETPROCESSOR MODULE WINDOW	4-105
FIGURE 4-109: CONFIG PACKETPROCESSOR WINDOW	4-105
FIGURE 4-110: COMMAND PACKETPROCESSOR WINDOW	4-106
FIGURE 4-111: STATUS PACKETPROCESSOR WINDOW	4-106
FIGURE 4-112: TDM SIMULATOR MODULE.....	4-108
FIGURE 4-113: CONFIG TDM SIMULATOR WINDOW	4-108
FIGURE 4-114: TDM SIMULATOR STATUS WINDOW.....	4-110
FIGURE 4-115: UNKNOWN MODULE WINDOW	4-111
FIGURE 4-116: ACE SFDU MODULE	4-112
FIGURE 4-117: ACE SFDU CONFIG WINDOW.....	4-112
FIGURE 4-118: ACE SFDU STATUS WINDOW	4-114
FIGURE 4-119: AMES COMMAND ENCAPSULATOR MODULE WINDOW	4-115
FIGURE 4-120: CONFIG AMES COMMAND ENCAPSULATOR WINDOW	4-115
FIGURE 4-121: COMMAND AMES COMMAND ENCAPSULATOR WINDOW	4-116
FIGURE 4-122: AXAF SFDU MODULE	4-117
FIGURE 4-123: AXAF SFDU CONFIG WINDOW	4-117
FIGURE 4-124: AXAF SFDU STATUS WINDOW.....	4-119
FIGURE 4-125: DST SFDU MODULE.....	4-120
FIGURE 4-126: DST SFDU CONFIG WINDOW	4-120
FIGURE 4-127: DST SFDU STATUS WINDOW	4-122
FIGURE 4-128: EDOS SERVICE HEADER MODULE.....	4-123
FIGURE 4-129: EDOS SERVICE HEADER CONFIG WINDOW	4-123
FIGURE 4-130: EDOS SERVICE HEADER STATUS WINDOW.....	4-124
FIGURE 4-131: IPDU FORMATTER MODULE WINDOW.....	4-125
FIGURE 4-132: CONFIG IPDU FORMATTER WINDOW.....	4-125
FIGURE 4-133: IPDU RECEIVER MODULE WINDOW.....	4-127
FIGURE 4-134: CONFIG IPDU RECEIVER WINDOW.....	4-127
FIGURE 4-135: STATUS IPDU RECEIVER WINDOW	4-129
FIGURE 4-136: LEO-T CDH FORMATTER MODULE.....	4-130
FIGURE 4-137: LEO-T CDH FORMATTER CONFIG WINDOW	4-130
FIGURE 4-138: LEO-T CDH FORMATTER STATUS WINDOW.....	4-131
FIGURE 4-139: LEO-T CDH RECEIVER MODULE	4-132
FIGURE 4-140: LEO-T CDH RECEIVER CONFIG WINDOW	4-132
FIGURE 4-141: LEO-T CDH RECEIVER STATUS WINDOW.....	4-133
FIGURE 4-142: LEO-T TFDH TRANSMITTER MODULE	4-135
FIGURE 4-143: LEO-T TFDH TRANSMITTER CONFIG WINDOW	4-135
FIGURE 4-144: LEO-T TFDH TRANSMITTER STATUS WINDOW.....	4-136
FIGURE 4-145: NASCOM BLOCKER MODULE.....	4-137

FIGURE 4-146: NASCOM BLOCKER CONFIG WINDOW	4-137
FIGURE 4-147: NASCOM BLOCKER STATUS WINDOW.....	4-139
FIGURE 4-148: NASCOM DEBLOCKER MODULE	4-140
FIGURE 4-149: NASCOM DEBLOCKER CONFIG WINDOW.....	4-140
FIGURE 4-150: NASCOM DEBLOCKER STATUS WINDOW	4-141
FIGURE 4-151: RTP FORMAT WINDOW	4-142
FIGURE 4-152: CONFIG RTP FORMAT WINDOW	4-142
FIGURE 4-153: RTP FORMATTER STATUS WINDOW.....	4-144
FIGURE 4-154: RTP RECEIVER MODULE.....	4-145
FIGURE 4-155: RTP RECEIVER CONFIG WINDOW	4-145
FIGURE 4-156: RTP RECEIVER STATUS WINDOW.....	4-146
FIGURE 5-1:.....	5-2
FIGURE 5-2: AVTEC SERIAL OUTPUT MODULE CONFIGURATION	5-2
FIGURE 5-3: AVTEC SERIAL INPUT MODULE CONFIGURATION	5-3
FIGURE 5-4: AVTEC SERIAL OUTPUT MODULE CONNECTIONS	5-4
FIGURE 5-5: AVTEC SERIAL INPUT MODULE CONNECTIONS	5-4
FIGURE 5-6: BERT MODULE CONNECTIONS.....	5-5
FIGURE 5-7: AVTEC SERIAL OUTPUT STATUS.....	5-5
FIGURE 5-8: AVTEC SERIAL INPUT STATUS	5-6
FIGURE 5-9: BERT STATUS	5-6
FIGURE 5-10: VIRTUAL CHANNEL SIMULATOR MODULE CONFIGURATION	5-11
FIGURE 5-11: IPDU FORMATTER MODULE CONFIGURATION	5-11
FIGURE 5-12: NETWORK SOCKETS MODULE CONFIGURATION	5-12
FIGURE 5-13: CPU TIMER MODULE CONFIGURATION	5-12
FIGURE 5-14: VIRTUAL CHANNEL SIMULATOR MODULE CONNECTIONS	5-13
FIGURE 5-15: IPDU FORMATTER MODULE CONNECTIONS.....	5-13
FIGURE 5-16: CPU TIMER MODULE CONNECTIONS	5-14
FIGURE 5-17: IPDU RECEIVER MODULE CONFIGURATION.....	5-15
FIGURE 5-18: VIRTUAL CHANNEL PROCESSOR CONFIGURATION.....	5-16
FIGURE 5-19: VC PROCESSOR EXTENDED OPTIONS CONFIGURATION	5-17
FIGURE 5-20: NULL TRANSCEIVER 1 CONFIGURATION.....	5-17
FIGURE 5-21: NULL TRANSCEIVER 2 CONFIGURATION.....	5-17
FIGURE 5-22: FILE RECORDER 1 CONFIGURATION	5-18
FIGURE 5-23: FILE RECORDER 2 CONFIGURATION	5-18
FIGURE 5-24: NETWORK SOCKETS MODULE CONFIGURATION	5-19
FIGURE 5-25: VIRTUAL CHANNEL SIMULATOR STATUS.....	5-20
FIGURE 5-26: VIRTUAL CHANNEL PROCESSOR STATUS.....	5-20
FIGURE 5-27: NULL TRANSCEIVER 1 STATUS	5-21
FIGURE 5-28: NULL TRANSCEIVER 2 STATUS	5-21
FIGURE 5-29: FILE PLAYBACK MODULE CONFIGURATION.....	5-23
FIGURE 5-30: VIRTUAL CHANNEL PROCESSOR CONFIGURATION.....	5-23
FIGURE 5-31: VC PROCESSOR EXTENDED OPTIONS CONFIGURATION	5-24
FIGURE 5-32: NETWORK SOCKETS 1 CONFIGURATION.....	5-24
FIGURE 5-33: NETWORK SOCKETS 2 CONFIGURATION.....	5-25

FIGURE 5-34: CPU TIMER MODULE CONFIGURATION 5-25
FIGURE 5-35: VEDA BIT SYNC MODULE CONFIGURATION..... 5-27
FIGURE 5-36: AVTEC SERIAL INPUT MODULE CONFIGURATION 5-28
FIGURE 5-37: AVTEC SERIAL INPUT EXTENDED OPTIONS CONFIGURATION 5-28
FIGURE 5-38: VIRTUAL CHANNEL PROCESSOR CONFIGURATION..... 5-29
FIGURE 5-39: VIRTUAL CHANNEL PROCESSOR EXTENDED OPTIONS CONFIGURATION..... 5-29
FIGURE 5-40: IPDU FORMATTER MODULE CONFIGURATION 5-30
FIGURE 5-41: FILE RECORDER MODULE CONFIGURATION 5-30
FIGURE 5-42: NETWORK SOCKETS MODULE CONFIGURATION 5-30
FIGURE 5-43: TIME MODULE CONFIGURATION..... 5-31

TABLES

TABLE 2-1: SYSTEM PULL DOWN MENU OPTIONS	2-18
TABLE 2-2: REMOTE DESKTOP PULL DOWN MENU OPTIONS.....	2-21
TABLE 2-3: REMOTE MODULE PULL DOWN MENU OPTIONS.....	2-23
TABLE 3-1: AVAILABLE MODULE FUNCTIONS	3-4
TABLE 4-1: RATE ADJUST CONFIG PARAMETERS.....	4-6
TABLE 4-2: RATE ADJUST EXTENDED CONFIGURATION PARAMETERS.....	4-7
TABLE 4-3: RATE ADJUST STATUS PARAMETERS	4-9
TABLE 4-4: RATE ADJUST QUEUE STATUS PARAMETERS.....	4-9
TABLE 4-5: CONFIG AVTEC SERIAL INPUT BOARD PARAMETERS	4-11
TABLE 4-6: CONFIG AVTEC SERIAL INPUT SYNCHRONIZER PARAMETERS.....	4-13
TABLE 4-7: CONFIG AVTEC SERIAL INPUT EXTENDED OPTIONS PARAMETERS	4-16
TABLE 4-8: STATUS AVTEC SERIAL INPUT PARAMETERS	4-18
TABLE 4-9: STATUS AVTEC SERIAL INPUT R/S UNCORRECTABLE ERRORS PARAMETERS.....	4-20
TABLE 4-10: STATUS AVTEC SERIAL INPUT R/S CORRECTABLE ERRORS PARAMETERS.....	4-20
TABLE 4-11: STATUS AVTEC SERIAL INPUT EARTH RECEIVE TIME (ERT).....	4-20
TABLE 4-12: MONARCH AND MONARCH-E 48-BYTE APPENDED STATUS FORMAT	4-21
TABLE 4-13: APPENDED STATUS HARDWARE FLAGS	4-21
TABLE 4-14: REED-SOLOMON DECODER CHIP ANNOTATION FORMAT	4-23
TABLE 4-15: REED-SOLOMON DECODER CHIP TERSE QUALITY DEFINITION	4-24
TABLE 4-16: CONFIG AVTEC SERIAL OUTPUT TX CLOCK/DATA PARAMETERS	4-26
TABLE 4-17: CONFIG AVTEC SERIAL OUTPUT EXTENDED OPTIONS PARAMETERS	4-27
TABLE 4-18: STATUS AVTEC SERIAL OUTPUT PARAMETERS.....	4-28
TABLE 4-19: STATUS AVTEC SERIAL OUTPUT FRAME PARAMETERS.....	4-29
TABLE 4-20: CONFIG COM PORT TRANSCEIVER PARAMETERS.....	4-30
TABLE 4-21: COMMAND COM PORT TRANSCEIVER PARAMETERS.....	4-31
TABLE 4-22: STATUS COM PORT TRANSCEIVER WINDOW PARAMETERS	4-31
TABLE 4-23: CONFIG CPU TIMER PARAMETERS	4-32
TABLE 4-24: STATUS CPU TIMER PARAMETERS.....	4-33
TABLE 4-25: DUMPTOLP CONFIG PARAMETERS	4-34
TABLE 4-26: DUMPTOLP STATUS PARAMETERS.....	4-35
TABLE 4-27: CONFIG FILE PLAYBACK PARAMETERS.....	4-36
TABLE 4-28: STATUS FILE PLAYBACK PARAMETERS	4-37
TABLE 4-29: CONFIG FILE RECORDER PARAMETERS	4-38
TABLE 4-30: COMMAND FILE RECORDER PARAMETER.....	4-40
TABLE 4-31: STATUS FILE RECORDER PARAMETERS.....	4-40
TABLE 4-32: PTP MUX HEADER FORMAT.....	4-41
TABLE 4-33: NASA PB-4 TIME CODE FORMAT	4-42
TABLE 4-34: FILE SPOOLER COMMAND WINDOW PARAMETERS	4-44
TABLE 4-35: FILE SPOOLER CONFIGURATION PARAMETERS	4-44
TABLE 4-36: FILE SPOOLER STATUS FIELDS.....	4-45
TABLE 4-37: CONFIG SOCKETS PARAMETERS	4-46
TABLE 4-38: STATUS SOCKETS PARAMETERS.....	4-47

TABLE 4-39: SOCKETS VERSION 2 CONFIG PARAMETERS	4-48
TABLE 4-40: SOCKETS VERSION 2 STATUS PARAMETERS	4-49
TABLE 4-41: CONFIG APOGEE PSK MODULATOR PARAMETERS	4-51
TABLE 4-42: CONFIG TIME PARAMETERS	4-53
TABLE 4-43: CONFIG TIME MODULE TIME LATCHED REGISTERS PARAMETERS	4-53
TABLE 4-44: STATUS TIME PARAMETERS	4-54
TABLE 4-45: CONFIG AYDIN BIT SYNC INPUT PARAMETERS	4-55
TABLE 4-46: CONFIG AYDIN BIT SYNC INPUT LOOP PARAMETERS	4-56
TABLE 4-47: CONFIG AYDIN BIT SYNC OUTPUT PARAMETERS	4-56
TABLE 4-48: CONFIG AYDIN BIT SYNC VITERBI PARAMETERS	4-56
TABLE 4-49: STATUS AYDIN BIT SYNC PARAMETERS	4-56
TABLE 4-50: CONFIG PSKDEMOD PARAMETERS	4-57
TABLE 4-51: CONFIG PSKDEMOD GAIN PARAMETERS	4-58
TABLE 4-52: COMMAND PSKDEMOD PARAMETER	4-58
TABLE 4-53: STATUS PSKDEMOD PARAMETERS	4-59
TABLE 4-54: CONFIG PSKMOD PARAMETERS	4-60
TABLE 4-55: COMMAND GDP PSKMOD PARAMETERS	4-61
TABLE 4-56: CONFIG PCR2000 MODULE PARAMETERS	4-62
TABLE 4-57: CONFIG PCR2000 MODULE VIDEO PARAMETERS	4-63
TABLE 4-58: CONFIG PCR2000 MODULE GAIN CONTROL PARAMETERS	4-63
TABLE 4-59: COMMAND PCR2000 MODULE PARAMETERS	4-63
TABLE 4-60: PCR2000 MODULE STATUS PARAMETERS	4-64
TABLE 4-61: PCR2000 REPORTED VIDEO STATUS PARAMETERS	4-65
TABLE 4-62: PCR2000 REPORTED STATUS FLAGS PARAMETERS	4-65
TABLE 4-63: PCR2000 AUXIO DRIVER STATUS PARAMETERS	4-65
TABLE 4-64: CONFIG GPIB MODULE OPEN DEVICE PARAMETERS	4-67
TABLE 4-65: CONFIG GPIB COMMAND AND STATUS PARAMETERS	4-67
TABLE 4-66: COMMAND GPIB MODULE PARAMETERS	4-68
TABLE 4-67: CONFIG ODETICS TIME MODULE PARAMETERS	4-69
TABLE 4-68: ODETICS TIME COMMANDS	4-70
TABLE 4-69: ODETICS TIME BOARD TIME/STATUS PARAMETERS	4-71
TABLE 4-70: ODETICS TIME BOARD SATELLITE STATUS PARAMETERS	4-71
TABLE 4-71: ODETICS TIME BOARD POSITION STATUS PARAMETERS	4-71
TABLE 4-72: ODETICS TIME BOARD ERROR ESTIMATOR STATUS PARAMETERS	4-71
TABLE 4-73: ODETICS TIME BOARD GENERAL STATUS PARAMETERS	4-72
TABLE 4-74: ODETICS TIME BOARD STATUS PARAMETERS	4-72
TABLE 4-75: CONFIG VEDABitSYNC PARAMETERS	4-73
TABLE 4-76: CONFIG VEDABitSYNC OUTPUT PARAMETERS	4-74
TABLE 4-77: STATUS VEDABitSYNC PARAMETERS	4-74
TABLE 4-78: 8 BIT SORTER CONFIG INPUT PARAMETERS	4-76
TABLE 4-79: 8 BIT SORTER CONFIG OUTPUT RANGE PARAMETERS	4-77
TABLE 4-80: 8 BIT SORTER CONFIG DATA LOGGING PARAMETERS	4-77
TABLE 4-81: 8 BIT SORTER RECEIVE STATUS PARAMETERS	4-77
TABLE 4-82: 8 BIT SORTER OUTPUT STATUS PARAMETERS	4-78

TABLE 4-83: 8 BIT SORTER STATUS PARAMETERS.....	4-78
TABLE 4-84: CONFIG 16 BIT SORTER INPUT PARAMETERS	4-79
TABLE 4-85: CONFIG 16 BIT SORTER OUTPUT RANGE PARAMETERS.....	4-80
TABLE 4-86: CONFIG 16 BIT SORTER DATA LOGGING PARAMETERS	4-80
TABLE 4-87: 16 BIT SORTER RECEIVE STATUS PARAMETERS.....	4-80
TABLE 4-88: 16 BIT SORTER OUTPUT STATUS PARAMETERS	4-80
TABLE 4-89: 16 BIT SORTER STATUS PARAMETERS.....	4-81
TABLE 4-90: CONFIG BSS PARAMETERS	4-82
TABLE 4-91: COMMAND BSS MODULE PARAMETERS.....	4-83
TABLE 4-92: BSS STATUS PARAMETERS.....	4-84
TABLE 4-93: CONFIG BIT ERROR RATE TESTER ERROR INJECTION PARAMETERS	4-87
TABLE 4-94: STATUS BIT ERROR RATE TESTER STATUS PARAMETERS.....	4-87
TABLE 4-95: STATUS BIT ERROR RATE TESTER TRANSMITTER ERRORS PARAMETERS.....	4-88
TABLE 4-96: STATUS BIT ERROR RATE TESTER RECEIVER FRAMES PARAMETERS	4-88
TABLE 4-97: STATUS BIT ERROR RATE TESTER RECEIVER BIT ERRORS PARAMETERS.....	4-88
TABLE 4-98: CONFIG CCSDS VC PROCESSOR PARAMETERS	4-90
TABLE 4-99: CONFIG CCSDS VC PROCESSOR R/S ECC PARAMS PARAMETERS	4-90
TABLE 4-100: CONFIG CCSDS VC PROCESSOR EXTENDED OPTIONS PARAMETERS	4-91
TABLE 4-101: STATUS CCSDS VC PROCESSOR PARAMETERS.....	4-92
TABLE 4-102: STATUS CCSDS VC PROCESSOR EXTENDED STATUS PARAMETERS	4-93
TABLE 4-103: CONFIG CCSDS VC SIMULATOR PARAMETERS.....	4-96
TABLE 4-104: CONFIG CCSDS VC SIMULATOR REED-SOLOMON PARAMETERS	4-97
TABLE 4-105: STATUS CCSDS VC SIMULATOR PARAMETERS	4-98
TABLE 4-106: COMMAND ERROR INJECT PARAMETERS.....	4-99
TABLE 4-107: GATHER SCATTER CONFIG PARAMETERS.....	4-100
TABLE 4-108: IN-LINE UNZIP STATUS FIELDS	4-101
TABLE 4-109: IN-LINE ZIP STATUS FIELDS.....	4-102
TABLE 4-110: CONFIG NULL TRANSCEIVER PARAMETERS.....	4-103
TABLE 4-111: CONFIG PACKETPROCESSOR PARAMETERS.....	4-105
TABLE 4-112: COMMAND PACKETPROCESSOR PARAMETER.....	4-106
TABLE 4-113: STATUS PACKETPROCESSOR PARAMETERS.....	4-107
TABLE 4-114: CONFIG TDMSIM OUTPUT FRAME SYNC PARAMETERS	4-108
TABLE 4-115: CONFIG TDMSIM ID COUNTER PARAMETERS	4-109
TABLE 4-116: CONFIG TDMSIM FILL DATA PARAMETERS.....	4-109
TABLE 4-117: CONFIG TDMSIM CRC GENERATION PARAMETERS	4-109
TABLE 4-118: TDM SIMULATOR STATUS PARAMETERS.....	4-110
TABLE 4-119: ACE SFDU PRIMARY HEADER CONFIG PARAMETERS	4-113
TABLE 4-120: ACE SFDU SECONDARY HEADER CONFIG PARAMETERS	4-113
TABLE 4-121: ACE SFDU STATUS PARAMETERS	4-114
TABLE 4-122: CONFIG AMES COMMAND ENCAPSULATOR BARKERCODE PARAMETERS.....	4-115
TABLE 4-123: CONFIG AMES COMMAND ENCAPSULATOR PREAMBLE PARAMETERS	4-116
TABLE 4-124: CONFIG AMES COMMAND ENCAPSULATOR POSTAMBLE PARAMETERS.....	4-116
TABLE 4-125: AXAF SFDU PRIMARY HEADER CONFIG PARAMETERS.....	4-118
TABLE 4-126: AXAF SFDU SECONDARY HEADER CONFIG PARAMETERS.....	4-118

TABLE 4-127: AXAF SFDU STATUS PARAMETERS.....	4-119
TABLE 4-128: DST SFDU PRIMARY HEADER CONFIG PARAMETERS.....	4-121
TABLE 4-129: DST SFDU SECONDARY HEADER CONFIG PARAMETERS.....	4-121
TABLE 4-130: DST SFDU STATUS PARAMETERS.....	4-122
TABLE 4-131: EDOS SERVICE HEADER CONFIGURATION PARAMETERS.....	4-123
TABLE 4-132: EDOS SERVICE HEADER STATUS PARAMETERS.....	4-124
TABLE 4-133: CONFIG IPDU FORMATTER SOURCE CODE PARAMETERS.....	4-126
TABLE 4-134: CONFIG IPDU FORMATTER DESTINATION CODE PARAMETERS.....	4-126
TABLE 4-135: CONFIG IPDU FORMATTER MESSAGE TYPE PARAMETERS.....	4-126
TABLE 4-136: CONFIG IPDU FORMATTER SPACECRAFT ID PARAMETERS.....	4-126
TABLE 4-137: CONFIG IPDU RECEIVER SOCKET PARAMETERS.....	4-127
TABLE 4-138: CONFIG IPDU RECEIVER PARAMETERS.....	4-128
TABLE 4-139: CONFIG IPDU RECEIVER DESTINATION CODE FILTER PARAMETERS.....	4-128
TABLE 4-140: STATUS IPDU RECEIVER PARAMETERS.....	4-129
TABLE 4-141: STATUS IPDU RECEIVER LAST IPDU HEADER PARAMETERS.....	4-129
TABLE 4-142: LEO-T CDH FORMATTER CONFIG PARAMETERS.....	4-130
TABLE 4-143: LEO-T CDH FORMATTER STATUS PARAMETERS.....	4-131
TABLE 4-144: LEO-T CDH RECEIVER SOCKET CONFIG PARAMETERS.....	4-132
TABLE 4-145: LEO-T CDH RECEIVER DESTINATION CODE CONFIG PARAMETERS.....	4-133
TABLE 4-146: LEO-T CDH RECEIVER CONFIG PARAMETERS.....	4-133
TABLE 4-147: LEO-T CDH RECEIVER STATUS PARAMETERS.....	4-134
TABLE 4-148: LEO-T CDH RECEIVER LAST HEADER STATUS PARAMETERS.....	4-134
TABLE 4-149: LEO-T TFDH TRANSMITTER CONFIG PARAMETERS.....	4-135
TABLE 4-150: LEO-T TFDH TRANSMITTER STATUS PARAMETERS.....	4-136
TABLE 4-151: NASCOM BLOCKER CONFIG INPUT PARAMETERS.....	4-137
TABLE 4-152: NASCOM BLOCKER CONFIG BLOCKING PARAMETERS.....	4-138
TABLE 4-153: NASCOM BLOCKER CONFIG BLOCK TYPE PARAMETERS.....	4-138
TABLE 4-154: NASCOM BLOCKER CONFIG TIME PARAMETERS.....	4-139
TABLE 4-155: NASCOM BLOCKER STATUS PARAMETERS.....	4-139
TABLE 4-156: NASCOM DEBLOCKER CONFIG PARAMETERS.....	4-140
TABLE 4-157: NASCOM DEBLOCKER STATUS PARAMETERS.....	4-141
TABLE 4-158: CONFIG RTP FORMAT PARAMETERS.....	4-142
TABLE 4-159: MSS DESTINATION TABLE EXAMPLE.....	4-144
TABLE 4-160: RTP FORMATTER STATUS PARAMETERS.....	4-144
TABLE 4-161: RTP RECEIVER CONFIG PARAMETERS.....	4-145
TABLE 4-162: RTP RECEIVER STATUS PARAMETERS.....	4-146

SCOPE

This manual provides users with the information necessary to operate the PTP NT Programmable Telemetry Processor (PTP). The PTP is a flexible and powerful satellite telemetry and command front-end system. The PTP accepts multiple telemetry streams and outputs time-tagged frame or packet data over a network to workstations in a distributed telemetry analysis system. The PTP also supports high speed data logging for store and forward operation. The system includes a command gateway that accepts input from a networked command processor and outputs to the uplink.

HOW TO USE THIS MANUAL

This manual is grouped into three sections to (1) introduce the user to the PTP and its main menus, (2) walk the user through building and manipulating a desktop, and (3) document the available module functions.

Chapters 1 introduces the PTP, and summarizes the PTP architecture.

Chapter 2 describes how to install and operate the PTP software.

Chapter 3 describes how to work with PTP modules and desktops.

Chapter 4 provides a detailed description for all of the PTP modules.

Chapter 5 provides User Tutorials.

Appendix A provides information on PTP Server Functions, which includes PTP Server commands, creating script files, and adding security to PTP NT Control ports and TCP Server sockets ports.

Appendix B provides summaries for various data and command encapsulation formats.

Appendix C provides a list of acronyms and abbreviations.

This User's Manual also provides a list of related documentation—functional descriptions and other manuals—written about various components of the PTP. Acronyms and abbreviations used in this manual are defined in the Acronyms and Abbreviations list.

RELATED DOCUMENTATION

The following documents contain information you may find useful to reference as you read this manual:

1. Avtec Systems, Inc. *AT-HSIO2 High Speed Serial I/O Board with Frame Synchronizer Functional Description*. Fairfax, Virginia, 1995.
2. Avtec Systems, Inc. *AV_FSTS Frame Synchronizer/Telemetry Simulator Dynamic Link Library for Windows NT Programmer's Guide*. Fairfax, Virginia, 1997.
3. Avtec Systems, Inc. *AV_PTP Programmable Telemetry Processor Remote Interface Library Programmer's Guide*. Fairfax, Virginia, 1997.
4. Avtec Systems, Inc. *MONARCH-E Frame Synchronizer/Telemetry Simulator with Reed-Solomon Encoder/Decoder Functional Description*. Fairfax, Virginia, 1997.

Additional Related Documentation

1. Apogee Labs, Inc. *Model ISA-PSK PSK Modulator User's Manual*. North Wales, Pennsylvania, 1996.
2. Apogee Labs, Inc. *Model ISA-STG2 Synchronized Time Generator User's Manual*. North Wales, Pennsylvania, 1996.
3. Aydin Corporation, *PC329 Tunable BPSK Demodulator Installation and Operation Manual*. Horsham, Pennsylvania, 1996.
4. Aydin Corporation, *PC335 Bit Synchronizer Installation and Operation Manual*. Horsham, Pennsylvania, 1996.
5. GDP Space Systems, *PSK004 ISA-BUS BPSK Modulator User's Manual*. Horsham, Pennsylvania, 1995.
6. Veda Systems, Inc. *DB200 Convolutional Decoder Module Technical Reference Manual*. California, Maryland, September 1994.
7. Veda Systems, Inc. *Series-2/10 and DBS200 PC Bit Synchronizer Card Technical Reference Manual*. California, Maryland, January 1996.
8. Microdyne Corporation. *Model PCR-2000 Telemetry PC Receiver Operations Manual*. Ocala, Florida, January 1996.

Chapter 1

INTRODUCTION

AVTEC's Programmable Telemetry Processor (PTP) is a PC-based, multi-channel telemetry and command processing system. The PTP's unique data handling capabilities are ideal for a telemetry front-end system which performs data acquisition, real-time network transfer, and store and forward operations. Support for both Time-division Multiplexed (TDM) and CCSDS packetized telemetry formats provides the flexibility to support multiple spacecraft with a single front-end system.

The PTP acts as a gateway that accepts multiple telemetry streams and outputs time-tagged frame or packet data over a network to workstations in a distributed satellite control and analysis system. The PTP also includes a command gateway that accepts input from the network and outputs serial commands to the uplink. Key elements of the PTP system are AVTEC's MONARCH-E PCI-based CCSDS/TDM Telemetry Processor/Simulator board, a network-based, distributed computing architecture, and the Windows NT operating system.

The PTP is useful throughout the lifecycle of a satellite system. During development, integration, and test, the front-end system can be used as a test tool in a distributed test environment. During operations, the system is installed at remote ground stations, with network links back to operations center(s) for telemetry and command processing and analysis.

The PTP can be controlled locally or remotely via the network using a socket interface. The use of standard network protocols (TCP/IP) facilitates integration with existing telemetry analysis and command processing systems.

PTP OVERVIEW

Section 1: PTP Architecture

PTP Hardware

The PTP is a PC-based system running under the Windows NT operating system. The hardware architecture is shown in **Figure 1-1**. The PC contains PCI and ISA I/O busses as well as the standard PC peripherals and application-specific interfaces. The number of I/O boards is limited by the number of available ISA and PCI slots in the PC. The PTP software controls and monitors all aspects of the PC configuration.

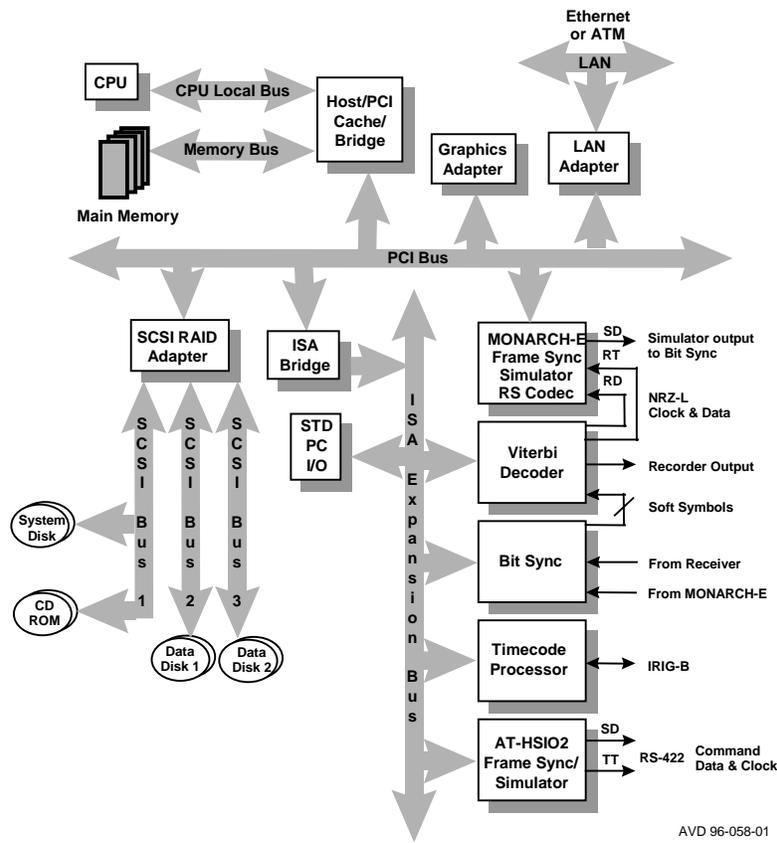


Figure 1-1: PC-based Front-end Processor Block Diagram

Standard PC interfaces supported by PTP include:

- Ethernet interface
- SCSI and IDE hard disks
- CD ROM drive
- COM Port
- LPT Port

Application-specific interfaces supported by the PTP include:

- AT-HSIO2 Frame Synchronizer/Telemetry Simulator
- MONARCH-E PCI Frame Synchronizer/Telemetry Simulator with Reed-Solomon Encoder/Decoder

- Bit Synchronizer (*Aydin PC 335 or Veda TSCN-300*) with Viterbi decoder
- Command BPSK Modulator (*Apogee ISA-PSK*)
- Tunable BPSK Modulator (*General Data Products PSK004*)
- Tunable Subcarrier BPSK Demodulator (*Aydin PC329*)
- IRIG-B and NASA-36 Time Code Processor (*Apogee ISA-STG2 or Odetics GPSync*)

The PTP acquires telemetry data streams using the Bit Synchronizer with Viterbi decoder and the Frame Synchronizer portion of the AT-HSIO2 or MONARCH-E. Both the AT-HSIO2 and the MONARCH-E perform frame synchronization using an adaptive strategy, serial-to-parallel conversion, and time tagging. The MONARCH-E supports rates up to 25 Mbps with hardware support for derandomization, CRC, and Reed-Solomon decoding. The AT-HSIO2 supports data rates up to 2 Mbps. Both boards are more like network interface cards than traditional frame synchronizer/decom cards in that they provide a stream of frame data to the PC rather than individual raw measurements and tags. Because of this flexible stream architecture, the PTP can support both TDM framed telemetry as well as packetized telemetry.

The AT-HSIO2 and MONARCH-E are bi-directional boards that can also support Telemetry Simulation, Command Output, and Bit Error Rate Testing (BERT). The MONARCH-E Telemetry Simulator performs Reed-Solomon encoding, interleaving, CRC encoding, sync marker insertion, pseudo-randomization, and convolutional encoding at data rates up to 25 Mbps. The AT-HSIO2 outputs serial streams at up to 2 Mbps. The boards also support bit error rate testing and data quality monitoring for testing and pre-pass verification functions.

PTP Software

The PTP software consists of a server application (PTP NT.EXE), a graphical user interface (CONSOLE.EXE), and the module dynamic link libraries (DLL). The PTP NT server controls the PTP hardware and performs all of the real-time data processing. The CONSOLE application provides an easy-to-use interface for controlling the configuration of the PTP system. The CONSOLE communicates with the PTP NT through TCP/IP sockets. The CONSOLE can run locally on the PTP system or on a remote PC that is connected to the PTP system via a TCP/IP network. The PTP NT application can also be remotely controlled by a user application through the AV_PTP Remote Interface Library.

Section 2: Theory of Operation

The PTP Desktop

The user creates a data acquisition and processing solution using the PTP desktop. The PTP supports the acquisition, processing, and routing of multiple, simultaneous data streams using a combination of software modules, I/O boards, and PC peripherals. PTP data acquisition and processing is completely software configurable. The PTP operates three types of PTP modules:

Auxiliary I/O, Data I/O, and Data Processing. The user creates a desktop by loading a combination of modules to perform specific tasks.

Auxiliary I/O Modules control and monitor auxiliary I/O boards such as a Bit Synchronizer, a BPSK modulator, a BPSK demodulator, or a Time Code Processor. These modules do not transfer real-time data streams into or out of the system. Only control and status information is passed between the CPU and the auxiliary I/O board.

Data I/O Modules control the data I/O devices in the system, including network interfaces, file devices, and the AT-HSIO2 or MONARCH-E boards. The Data I/O Modules transfer real-time data streams into or out of the system using the application-specific interfaces and standard PC peripherals.

Data Processing Modules perform some software processing on data as it flows through the module. An example is the CCSDS Virtual Channel Processor (VCP) which filters frame data based on Virtual Channel ID (VCID) and data quality.

PTP Module Functions

Auxiliary I/O Modules

- Bit Synchronizers
- Viterbi Decoder
- PSK Demodulator
- PSK Modulators
- S-Band Receiver/Demodulator
- GPIB (IEEE488) Interface
- Time Code Processors

Data I/O Modules

- File Playback (File playback from hard disk, piped device, or CD-ROM)
- File Recorder (File record to hard disk or piped device)
- File Spooler (Spool files to hard disk or piped device)
- Avtec Serial Input (AT-HSIO2 or MONARCH-E Frame Synchronizer)
- Avtec Serial Output (AT-HSIO2 or MONARCH-E PCM Simulator)
- Avtec Rate Adjust (AT-HSIO2 or MONARCH-E PCM Simulator)
- Network Socket (UDP, UDP multicast, TCP client, TCP server)
- ComPorts Transceiver

- CPU Timer (User-configurable event outputs)
- Dump to Line Printer (Output ASCII formatted data to printer or file)

Data Processing Modules

- Bit Error Rate Tester
- Bit Density Correction
- Error Inject
- In-Line Zip/UNZip
- Null Transceiver
- Best Source Select
- CCSDS Packet Processor
- 8/16 Bit Sorter
- Gather Scatter Module
- CCSDS Virtual Channel Processor
- CCSDS Virtual Channel Simulator
- TDM Simulator
- AMES Command Encapsulator
- EDOS Service Header
- IPDU Formatter
- IPDU Receiver
- AXAF SFDU Formatter
- DS-T SFDU Formatter
- ACE SFDU Formatter
- LEO-T Telemetry Transmitter (TFDHF)
- LEO-T Command Formatter (CDHF)
- LEO-T Command Receiver (CDHR)
- NASCOM Blocker/Deblocker
- NASCOM RTP Formatter/Receiver

Data I/O Modules and Data Processing Modules are “connected” together to perform real-time data processing tasks. The user selects the data flow between the various modules. Each Data Module has data inputs and outputs and an event input and output.

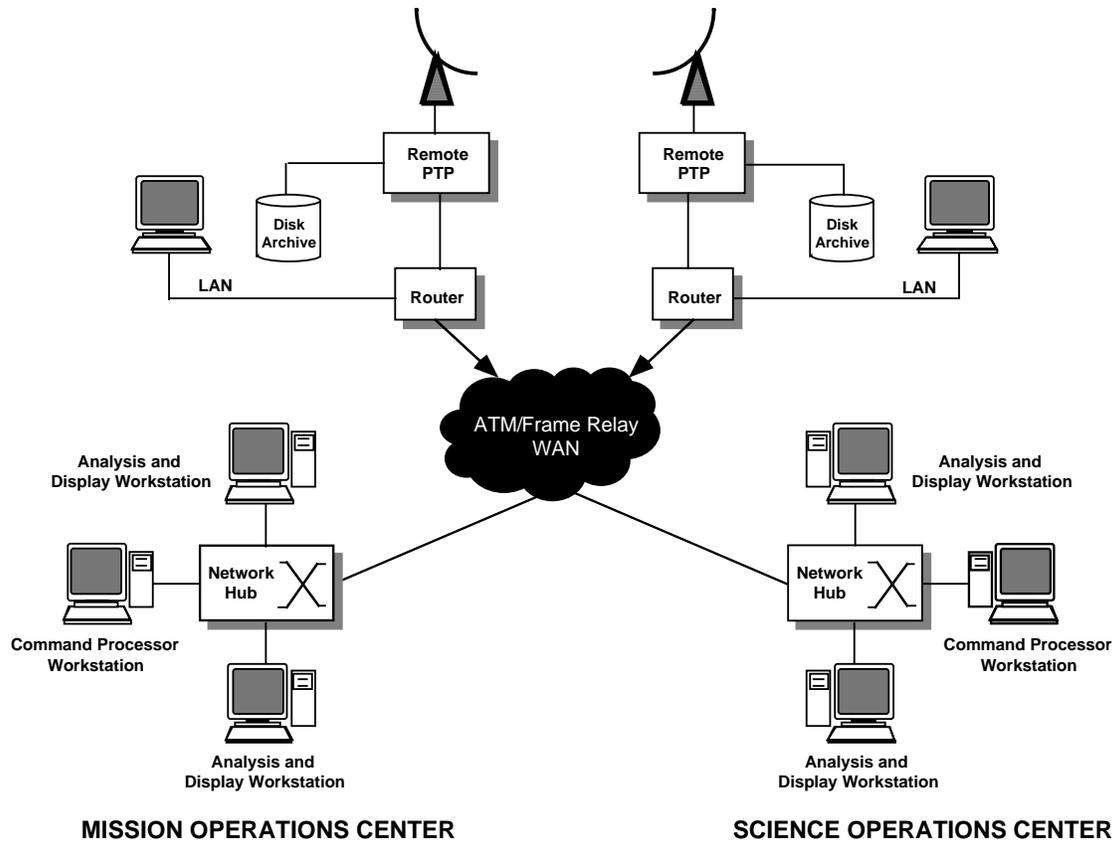
A module's data output is connected to another module to pass data buffers from the first module to the second. For example, if the data output of the Serial Input Module is connected to the Recorder Module, the Serial Input Module will pass frame data from the I/O board to the Recorder Module each time it receives a frame on the serial data line. The data output of the Serial Input Module could also be connected to the Socket Module so that each frame of data is also transmitted to the network.

A module's event output is connected to another module to signal an event to the second module. For example, the Serial Output Module provides an event output every time it completes writing a frame of data to the I/O board. This event output is used to tell another module to provide another buffer of data to the Serial Output Module. To playback a binary file to the serial output, the Playback Module's data output is connected to the Serial Output Module and the Serial Output Module's event output is connected to the Playback Module. The event connection provides a flow control between the two modules.

Section 3: Applications

PTP Front-end Telemetry and Command Processor

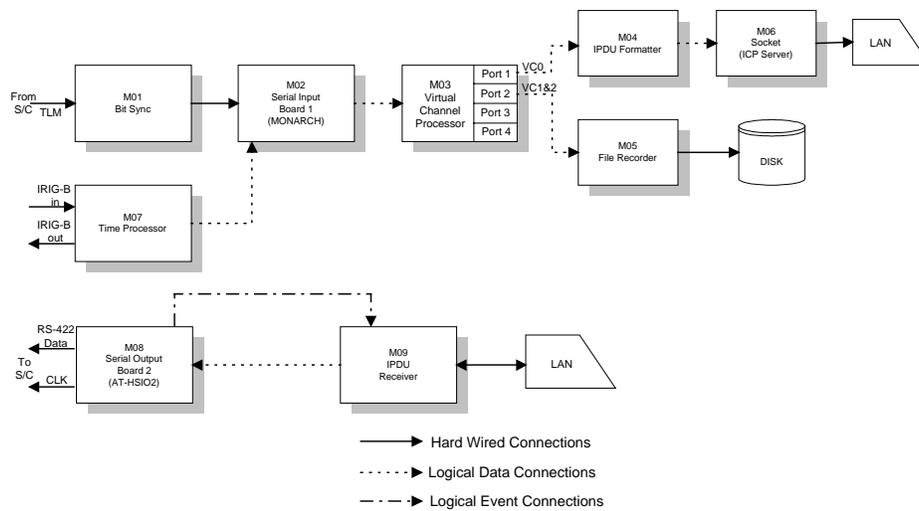
The PTP acts as a gateway between a network and the spacecraft uplink/downlink. **Figure 1-2** shows the PTP in a typical operational environment where multiple remote ground stations are connected to control centers via a wide area network (WAN).



AVD 96-018-01

Figure 1-2: PTP Front-end Processor Operational Environment

The PTP can input serial telemetry streams and route them in various ways, especially useful for front-end telemetry and command systems. **Figure 1-3** shows the PTP modules used in a front-end Telemetry and Command Processor.



AVD 97-026-01

Figure 1-3: PTP Module Connections for a Front-end Processor

The modules used for the telemetry data flow are:

- Bit Synchronizer/Viterbi Decoder
- File Recorder
- IPDU Formatter
- Network Socket
- Serial Input (*MONARCH-E*)
- Time Processor
- Virtual Channel Processor

The PTP accepts a serial telemetry stream using the Bit Synchronizer with Viterbi Decoder. The clock and data output from the Bit Synchronizer with Viterbi Decoder are wired to the MONARCH-E clock and data input. The MONARCH-E performs frame synchronization, derandomization, Reed-Solomon decoding, and time tagging. The Serial Input Module controls the operation of the MONARCH-E. The Serial Input Module's data output port is connected to the CCSDS VCP Module. The Serial Input Module passes telemetry frames with time tag and quality annotation to the CCSDS VCP module. The CCSDS VCP filters frames based on VCID and data quality. The VCP has multiple data output ports. Each data output port can be configured to output a particular subset of Virtual Channels. In this example, the VCP port 1 is configured to pass only frames with VCID 0 and is connected to the IPDU Formatter Module. The IPDU Formatter receives Virtual Channel Data Units (VCDU) from the VCP port 1 and prepends an IPDU header to the data. The IPDU Formatter is connected to a Socket Module that

is configured as a TCP server. The Socket Module sends VCID 0 frames with IPDU header over the network in real time. The VCP port 2 is configured to pass only frames with VCID 1 or 2 and to discard frames with other VCIDs or with sequence errors. The VCP port 2 is connected to the Recorder Module. The VCID 1 and 2 frames output from VCP port 2 are sent to the Recorder module which stores the data to disk.

The modules used for the command data flow are the IPDU Receiver Socket and the Serial Output (AT-HSIO2).

The PTP accepts commands from the network using another instance of the IPDU Receiver. The IPDU Receiver is a special type of Socket module that supports IPDU encapsulated data. It is configured as a TCP server and accepts IPDU encapsulated Command Link Transmission Units (CLTU) from a network client. The IPDU Receiver data output is connected to the Serial Output Module. The IPDU receiver strips the IPDU header from accepted network packets and passes the command data to the Serial Output Module. The Serial Output module serializes the data using the AT-HSIO2 serial output.

PTP Test Tools

An extensive set of test tools make the PTP useful throughout satellite and ground system development, integration and test, and final operations. Different life cycle stages do not require separate tools, which reduces cost and eliminates compatibility issues. In addition, using the PTP in a distributed test environment enables early integration and remote access to live test data.

The PTP provides the following test functions:

- Bit Error Rate Testing
- Data Logging and Playback
- Data Quality Monitoring (TDM and CCSDS)
- Network Protocol Conversion
- Telemetry Simulation (TDM and CCSDS)

The PTP provides a full suite of tools for satellite and ground system development, integration, and testing. As a BERT, the PTP transmits and receives a pseudo-random pattern (2047) useful for end-to-end system testing. Data quality monitoring (DQM) functions can verify communications links during development and integration, and are essential for maintaining statistics on operational links. Telemetry Simulation is critical in testing ground station systems during development and integration. Finally, the PTP can log telemetry data at high rates as it is collected. With AVTEC's PTP, you can log the data while gathering and viewing DQM statistics in real time, or while sending filtered data over the network to a remote processing station. After the test, you can access the logged data locally or via the network for further analysis and processing. The PTP will playback the logged data for more realistic ground system testing or for regression testing during development.

Chapter 2

INSTALLING AND OPERATING THE PTP

This chapter describes how to install and configure the PTP NT software. It also gives an overview of each of the PTP NT software components and their modes of operation. If you have received a fully configured system, you may skip to the Section 3: Operating the PTP.

Section 1: Installing PTP NT Software

The PTP NT software is a collection of application programs and dynamic link libraries that are designed to run on a PC/Windows NT system. The minimum system requirements for the PTP NT software are a PC with a Pentium 133 MHz processor and 32 MB of RAM running Windows NT 4. For data rates in excess of 5 Mbps, a Pentium II processor with at least 64 MB of RAM is recommended.

The PTP NT software is distributed on a variety of media:

- 1) CD-ROM
- 2) Floppy disks (4)
- 3) Online (ftp download from avtec.com)

To install the PTP NT software, perform the following steps.

- 1) Logon as a user with administrator privileges.
- 2) Be sure that none of the PTP NT software components are executing on the system.
- 3) Run SETUP.EXE from the PTP_NT installation disk 1. (Note that PTP_NT must be installed in the \PTP_NT directory.)

The Setup program will copy all of the PTP NT files to your computer. Setup also creates a PTP NT program group in the Start Menu under Programs. The entries in the PTP NT program group are shown in **Table 2-1**.

Table 2-1: PTP NT Program Group Options

Item	Description
AUXIO_Config	AUXIO (Auxiliary I/O) device driver Registry configuration utility.
Console	PTP Console graphical user interface.

Item	Description
Eeprom_Update	Command line program used to update the MONARCH-E onboard EEPROM from version 1.0 to 1.1. Instructions for running the Eeprom_update utility are given below. This update only applies to Rev. B MONARCH boards.
FSTS_Config	FSTS (Frame Synchronizer/Telemetry Simulator) device driver Registry configuration utility.
List_Servers	Utility which displays the IP addresses of PTP Servers and associated clients that are detected on the local network.
Monitor	System remote control and monitor application. For more details, please see the AV_PTP Remote Interface Library Programmer's Guide.
PTP_NT	Programmable Telemetry Processor for Windows NT application.
Readme	Readme file with release notes.

Section 2: Configuring the PTP NT System

The PTP NT system must be configured before operating the PTP NT software. The remaining configuration includes the following steps:

- 1) Miscellaneous Windows NT settings
- 2) Auxiliary I/O Device Driver (AUXIO) Configuration
- 3) Frame Synchronizer/Telemetry Simulator (FSTS) Device Driver Configuration
- 4) Eeprom_Update Utility (only for Rev. B MONARCH boards)
- 5) PTP NT Software License Registration

If no application specific hardware (Avtec AT-HSIO2 or MONARCH boards or Auxiliary I/O devices) is installed in the system, then steps 2 through 4 may be omitted.

Step 1) Miscellaneous Windows NT Settings

A user must have Power User or Administrator privileges to run PTP NT. A user's group attributes are set using the Windows NT User Manager (under Administrative Tools). The PTP NT software also requires that several Windows NT options be configured as outlined below:

- 1) Windows NT Explorer View->Options->View Tab - Select "Show all files".
- 2) Display Properties -> Plus! Tab - Uncheck "Show window contents while dragging".
- 3) System Properties -> Performance Tab - Set Application Performance Boost to None.

The PTP NT software also requires that the TCP/IP protocol be installed on the computer. If no network adapter is installed in the computer, the MS Loopback Adapter can be used in conjunction with the TCP/IP protocol. For more information, please refer to the Windows NT User's Manual.

The computer may be optionally configured to login automatically on boot up. To setup the computer to login automatically on boot up, perform the following steps.

- 1) Add a user called PTP_NT with password PTP_NT. Add PTP_NT to the power user group.
- 2) Run Regedit32.exe (in the System32 directory)
- 3) Go to HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon
- 4) AddValue:

AutoAdminLogon	REGSZ	1
DefaultPassword	REGSZ	PTP_NT
- 5) ModifyValue:

DefaultUserName	REGSZ	PTP_NT
-----------------	-------	--------

To bypass the autologon, hold down the Shift key during shutdown and reboot. This will allow you to logon as another user.

Step 2) Auxiliary I/O Device Driver (AUXIO) Configuration

The AV_AUXIO Auxiliary I/O library provides I/O port access to user applications. It is used by PTP_NT to control Auxiliary I/O devices such as Bit Synchronizers, Time Code Processors, Modulators, and Demodulators. If the system does not contain any Auxiliary I/O devices, then this section may be skipped.

The Windows NT Registry settings for AUXIO are defined using the AUXIO_Config utility. The AUXIO Windows NT Registry settings define how many boards are in the system. The AUXIO Windows NT Registry settings also define which systems resources (I/O addresses) are required by the AUXIO boards.

To run the AUXIO_Config utility, select AUXIO_Config from the Start->Programs->PTP_NT program group. The AUXIO_Config main window will be displayed as shown in **Figure 2-1** below.

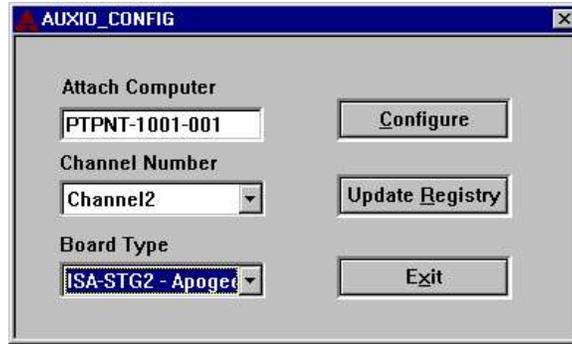


Figure 2-1: AUXIO_Config main window.

The parameters in the AUXIO_Config main window are outlined in **Table 2-2** below.

Table 2-2: AUXIO_Config Main Windows Parameters

Parameter	Description
Attach Computer	Displays the Computer Name for the system to be configured.
Channel Number	The AUXIO Channel number of the board to be configured. AUXIO assigns a logical channel number to each board.
Board Type	Type of board to be configured for the selected channel. The valid Board Types are shown in Table 2-3 below.
Configure	Displays the channel configuration dialog for the selected channel.
Update Registry	Writes the current configuration information to the Windows NT registry.
Exit	Exit the AUXIO_Config program.

The supported AUXIO Board Types are listed in **Table 2-3** below. **Table 2-3** also includes the I/O port requirements for each board type. This information can be used along with the Windows NT Diagnostics I/O Resource utility to select the base address for each of the AUXIO boards. The AUXIO_Config utility automatically checks to be sure that a selected address does not conflict with any other boards in the system.

Table 2-3: AUXIO_Config Available Board Types

Board Type	Description	Num I/O Ports
Bit Sync – Veda	Veda Bit Synchronizer	9
Viterbi Dec – Veda	Veda Viterbi Decoder	4
ISA-STG2 – Apogee	Apogee Synchronized Time Generator (IRIG-	16

	B or NASA-36)	
Bit Sync – Aydin	Aydin PC335 Bit Synchronizer	4
PSK Mod – GDP	GDP PSK004 BPSK Subcarrier Modulator	18
Demod – Aydin	Aydin PC329 BPSK Subcarrier Demodulator	4
PSK Mod – Apogee	Apogee ISA-PSK BPSK Command Modulator (16 KHz subcarrier)	2
PCR2000 – Microdyne	Microdyne PCR2000 FM Receiver	4

The configuration information for a specific board is entered using the Channel Configuration dialog shown in **Figure 2-2** below.



Figure 2-2: AUXIO_Config Channel Configuration Dialog.

The parameters in the Channel Configuration Dialog are defined in **Table 2-4** below.

Table 2-4: AUXIO_Config Channel Configuration Dialog.

Parameter	Description
Channel Number	Displays the channel number to be configured.
Board Type	Displays the board type for the board to be configured.
Board Type	Type of board to be configured for the selected channel.
I/O Address	Used to enter the I/O Base Address for the selected channel. This should correspond to the I/O address switch settings for the selected board.
OK	Updates the local copy of the configuration information and returns to the AUXIO_Config main window if no I/O address conflict is detected. An error dialog will be displayed if AUXIO_Config detects an I/O address conflict with another device in the system. Note that the new configuration information is not actually written to the Windows NT Registry until “Update Registry” is selected.
Cancel	Rejects any configuration changes and returns to the AUXIO_Config main window.

To configure the AUXIO device driver for a particular device (for example the Apogee ISA-STG2), perform the following steps:

- 1) Run AUXIO_Config (from the Start->Programs->PTP_NT sub menu)
- 2) Set Channel Number to **Channel 0**
- 3) Set Board Type to **ISA-STG2 - Apogee**
- 4) Click **Configure**
Channel Configuration Dialog
- 5) Set IO Port Address to **240**
- 6) Click **OK**
- 7) Click **Update Registry**
Update AUXIO configuration information to Registry?
- 8) Click **Yes**

Steps 2 through 6 can be repeated with a different Channel Number (Board Type and IO Port Address) to configure additional AUXIO boards that are in the system.

To remove an AUXIO channel from the Windows NT Registry perform the following steps:

- 1) Run AUXIO_Config (from the Start->Programs->PTP_NT sub menu)
- 2) Set Channel Number to the desired value
- 3) Set Board Type to **NONE**
- 4) Click **Configure**
Changing board type will release Channel 0 resources, Continue?
- 5) Click **Yes**
Please select valid Board Type
- 6) Click **OK**
- 7) Click **Update Registry**
Update AUXIO configuration information to Registry?
- 8) Click **Yes**

The Windows NT Registry changes will take effect the next time the AUXIO driver is loaded at boot time.

Step 3) Frame Synchronizer/Telemetry Simulator (FSTS) Device Driver Configuration

The AV_FSTS Frame Synchronizer/Telemetry Simulator driver is used by PTP NT to control the AT-HSIO2 and MONARCH-E high speed serial I/O boards. The Windows NT Registry settings for FSTS are defined using the FSTS_Config.exe utility. The FSTS driver reads the Windows NT Registry settings to determine how many boards are in the system. The FSTS Windows NT Registry settings also define which systems resources (I/O address, IRQ line, and DMA channel) are required by the AT-HSIO2 boards.

To run the FSTS_Config utility, select FSTS_Config from the Start->Programs->PTP_NT program group. The FSTS_Config main window will be displayed as shown in **Figure 2-3** below.

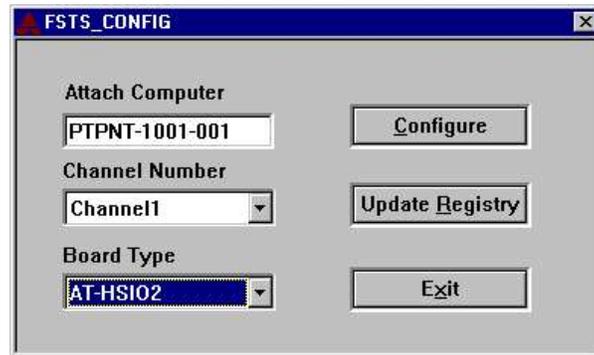


Figure 2-3: FSTS_Config main window.

The parameters in the FSTS_Config main window are outlined in **Table 2-5** below.

Table 2-5: FSTS_Config Main Windows Parameters

Parameter	Description
Attach Computer	Displays the Computer Name for the system to be configured.
Channel Number	The FSTS Channel number of the board to be configured. FSTS assigns a logical channel number to each board.
Board Type	Type of board to be configured for the selected channel. Valid options are AT-HSIO2 and MONARCH. AT-HSIO2-IQ is displayed in the list, but is not a valid option for PTP NT.
Configure	If the selected Board Type is AT-HSIO2, then the Channel Configuration dialog will be displayed. The Channel Configuration Dialog is used to define the I/O address, IRQ line, Rx DMA channel, and Tx DMA channel that are required for the selected board. The I/O address defines the mapping between the FSTS logical channel number and the physical board. If the selected Board Type is MONARCH, then a "Channel Configured" message will be displayed. There is no Channel Configuration Dialog for the MONARCH because its system resource requirements are determined at boot time using Plug-n-Play autoconfiguration. For systems with multiple MONARCH boards, the FSTS logical channel number to physical board location depends on the order that the PCI bus is scanned at boot. The MONARCH boards are assigned FSTS logical channel numbers in the order they are detected on the PCI bus.
Update Registry	Writes the current configuration information to the

	Windows NT registry.
Exit	Exit the FSTS_Config program.

The configuration information for a specific AT-HSIO2 board is entered using the Channel Configuration dialog shown in **Figure 2-4** below.



Figure 2-4: FSTS_Config Channel Configuration Dialog.

The parameters in the Channel Configuration Dialog are defined in **Table 2-6** below.

Table 2-6: FSTS_Config AT-HSIO2 Channel Configuration Dialog Parameters

Parameter	Description
Channel Number	Displays the channel number to be configured.
Board Type	Displays the board type for the board to be configured.
IRQ	Used to assign the IRQ line for the selected channel. The AT-HSIO2 board supports IRQ 9, 10, 11, 12, and 15. Only the available IRQ lines are shown. The ISA interrupt lines are not shared so each AT-HSIO2 card in the system requires its own IRQ line. The IRQ lines for ISA cards must typically be reserved in the BIOS PCI/Plug-n-Play configuration.
I/O Address	Used to enter the I/O Base Address for the selected channel. This should correspond to the I/O address switch settings for the selected board. This field defines the FSTS logical channel number to physical board mapping.
RxDrq	Used to assign the DMA channel to the receiver (serial input) for the selected board. Only the available DMA channels are shown. A 16-bit DMA channel (5,6, or 7) will support higher data transfer rates than an 8-bit DMA channel (0,1, or 3).
TxDq	Used to assign the DMA channel to the transmitter (serial output) for the selected board. Only the available DMA channels are shown. If the transmitter will be used for commanding, an 8-bit DMA channel (0,1, or 3) should be

	used.
OK	Updates the local copy of the configuration information and returns to the FSTS_Config main window if no I/O address conflict is detected. An error dialog will be displayed if FSTS_Config detects an I/O address conflict with another device in the system. Note that the new configuration information is not actually written to the Windows NT Registry until "Update Registry" is selected.
Cancel	Rejects any configuration changes and returns to the FSTS_Config main window.

Important AT-HSIO2 Configuration Notes:

- 1) **The AT-HSIO2 Base Address should be set to an address on a 32-byte boundary within the range 0x200 to 0x3FF.**
- 2) **IRQ9 should be reserved in the BIOS PCI/Plug-n-Play configuration when using the AT-HSIO2 board to prevent IRQ9 from being assigned to a PCI device.**
- 3) **DMA Channels 0 or 3 may not work for PTP systems with Tyan TITAN-III S1468 motherboards with AMIBIOS. Switching to Award BIOS corrects this problem.**

The Windows NT Diagnostics utility (Resources Tab) provides useful information about the allocation of system resources (I/O address, IRQ lines, and DMA channels). This information can be used to select the base address for each of the AT-HSIO2 boards. The FSTS_Config utility will automatically check to be sure that a selected address does not conflict with any other boards in the system.

To configure the FSTS device driver for a MONARCH channel, perform the following steps:

- 1) Run FSTS_Config (from the Start->Programs->PTP_NT sub menu)
- 2) Set Channel Number to **Channel 0**
- 3) Set Board Type to **MONARCH**
- 4) Click **Configure**
Fsts_config Channel Configured
- 5) Click **OK**
- 6) Click **Update Registry**
Update Fsts configuration information to Registry?
- 7) Click **Yes**

Steps 2 through 5 can be repeated with a different Channel Number to configure additional MONARCH boards that are in the system.

To configure the FSTS device driver for an AT-HSIO2 channel, perform the following steps:

- 1) Run FSTS_Config (from the Start->Programs->PTP_NT sub menu)
- 2) Set Channel Number to **Channel 0**
- 3) Set Board Type to **AT-HSIO2**
- 4) Click **Configure**
Channel Configuration Dialog
- 5) Set IRQ to **12**
- 6) Set IO Port Address to **200**
- 7) Set RxDrq to **6**
- 8) Set TxDrq to **7**
- 9) Click **OK**
- 10) Click **Update Registry**
Update FSTS configuration information to Registry?
- 11) Click **Yes**

Steps 2 through 9 can be repeated with a different Channel Number (and I/O address, IRQ, RxDrq, and TxDrq) to configure additional AT-HSIO2 boards that are in the system.

To remove an FSTS channel from the Windows NT Registry perform the following steps:

- 1) Run FSTS_Config (from the Start->Programs->PTP_NT sub menu)
- 2) Set Channel Number to the desired value
- 3) Set Board Type to **NONE**
- 4) Click **Configure**
Changing board type will release Channel 0 resources, Continue?
- 5) Click **Yes**
- 6) Click **Update Registry**
Update FSTS configuration information to Registry?
- 7) Click **Yes**

The Windows NT Registry changes will take effect the next time the FSTS driver is loaded at boot time. Once the FSTS device driver configuration is complete, shutdown the system and reboot Windows NT so that the FSTS driver will be reloaded with the new Registry settings. Check the Event Viewer (Administrative Tools) to make sure that the FSTS driver loaded properly (there should be no FSTS event messages in the Event Log). If there are any FSTS event messages, check to be sure that the I/O base address in the FSTS Config settings agrees with the switch settings on the board. If the switch settings are correct, the FSTS event message may indicate that a board in the system is not working properly and needs to be returned to Avtec for repair.

Step 4) EEPROM Update Utility (Rev. B MONARCH boards only)

Eeprom_Update is a command line program that is used to update the MONARCH (or MONARCH-E) onboard EEPROM from version 1.0 to 1.1. This update applies only to Rev. B MONARCH boards.

Since Eeprom_Update is a console based application, each of the steps below should be executed within a command window.

- 1) Start the FSTS driver if it is not already started
" net start FSTS "
- 2) Run the utility
" eeprom_update "
- 3) The utility will scan each installed MONARCH board for type, PCB mask and eeprom revision. If an out of date combination is detected it will ask the user if the board should be upgraded. Answering "y" will proceed with the upgrade.
- 4) Reboot the computer to allow the changes to take effect.
- 5) Note that if the utility is rerun before rebooting it will incorrectly identify recently upgraded boards which will cause the utility to re-upgrade those boards. This is unnecessary, but will not cause a problem.

Step 5) PTP NT Software License Registration

The PTP NT software is licensed for installation on a single machine so the PTP NT license is node locked to a particular machine. The first time that PTP NT (Start ->Programs->PTP_NT->PTP_NT) is executed after installation the display will appear as shown in **Figure 2-5**.

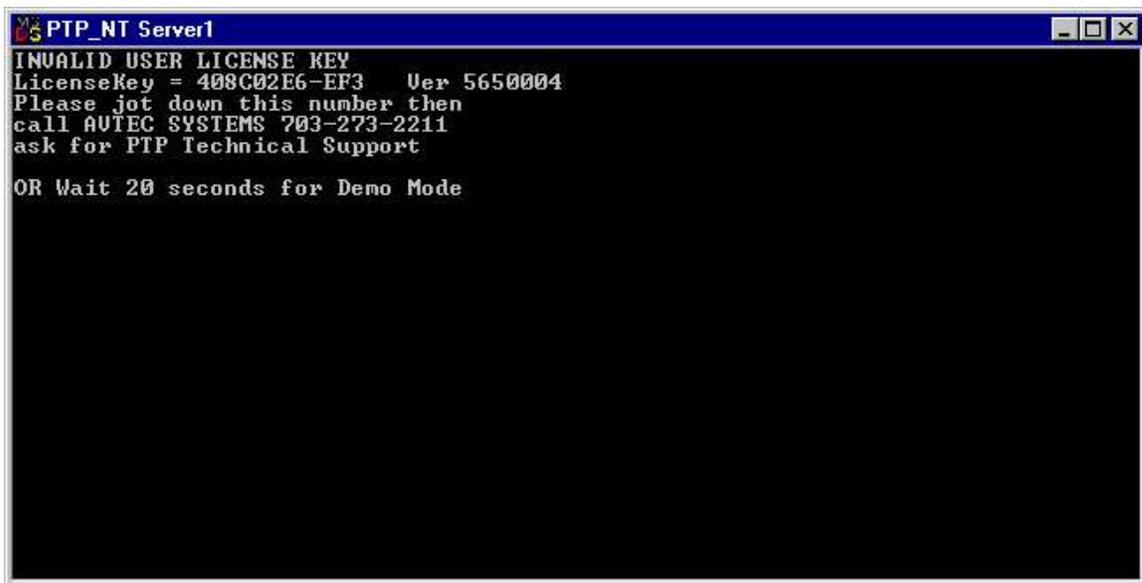


Figure 2-5: PTP NT Display before License Registration

Contact Avtec Systems with the License Key that is displayed on the second line, 408C02E6-EF3 in this example. You will be provided with a new license key that will unlock the software

for the machine on which it is installed. In this example, the new license key is A069E901-B843A5A5. To register the new license type ADD LICENSE in the PTP NT console window. Enter the new license key at the prompt as shown in **Figure 2-6**.



```

PTP_NT Server1
INVALID USER LICENSE KEY
LicenseKey = 408C02E6-EF3   Ver 5650004
Please jot down this number then
call AVTEC SYSTEMS 703-273-2211
ask for PTP Technical Support

OR Wait 20 seconds for Demo Mode

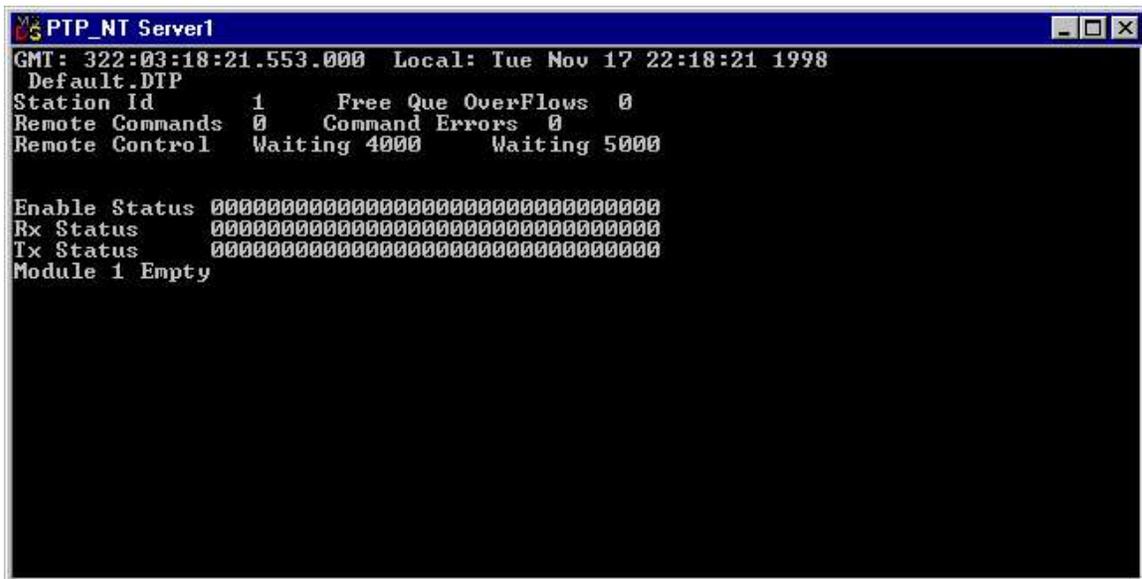
ADD LICENSE
Enter New License #  A069E901-B843A5A5

Registering License ....

```

Figure 2-6: PTP NT Display during License Registration

After the PTP registers the new license, the PTP NT display will appear as shown in **Figure 2-7**.



```

PTP_NT Server1
GMT: 322:03:18:21.553.000   Local: Tue Nov 17 22:18:21 1998
Default.DTP
Station Id      1      Free Que OverFlows  0
Remote Commands 0      Command Errors  0
Remote Control  Waiting 4000   Waiting 5000

Enable Status  00000000000000000000000000000000
Rx Status     00000000000000000000000000000000
Tx Status     00000000000000000000000000000000
Module 1 Empty

```

Figure 2-7: PTP NT Display after License Registration

The PTP store the new license key in the USER_KEY file in the \Ptp_User subdirectory. The license must be re-registered if the Computer Name (under Network Properties - Identification) is

changed. A new registration key will be also be required to install the PTP software on another machine.

Section 3: Operating the PTP

The PTP software consists of a server application (PTP NT.EXE), a graphical user interface (CONSOLE.EXE), and the module DLLs. The PTP NT server controls the PTP hardware and performs all of the real-time data processing. The CONSOLE application provides an easy-to-use interface for controlling the configuration of the PTP system. The CONSOLE communicates with the PTP NT through TCP/IP sockets. It can run locally on the PTP system or on a remote PC that is connected to the PTP system via a TCP/IP network. The PTP NT application can also be remotely controlled by a user application through the AV_PTP Remote Interface Library.

Communication between the CONSOLE interface and the PTP NT server are performed using a software library over TCP/IP sockets. The console and the server can run on the same machine or on two different machines. Communication between the console and the server is always via TCP/IP. **Figure 2-8** illustrates the communication link between the console (*or client*) and the server.

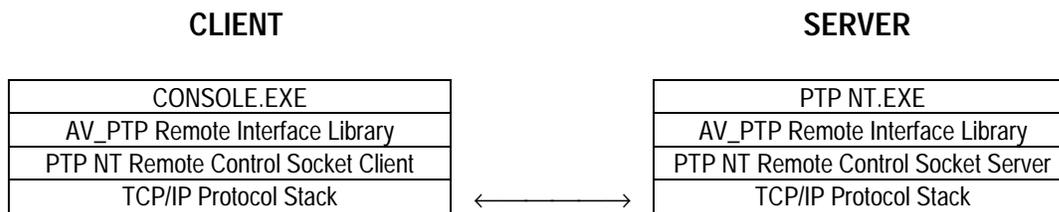


Figure 2-8: PTP Client-Server Communication Link

To remotely access the PTP, a client (CONSOLE) must know the IP address and socket number of the targeted server (PTP). Whenever it receives a remote command, the PTP responds with a message, even if only to report an error or give status. The remote PTP does not require the destination address because it acts as a server. It allows connections instead of initiating them (*during other non-Remote Control communications, the PTP may act as either a client or a server*). PTP status messages may be configured as multicast, allowing multiple control devices to monitor at the same time. In addition, users can restrict access to PTP control ports and TCP Server sockets ports by creating *Well-Known Hosts* files. Please see Section 3 of this chapter, or **Appendix A** for more information on creating these security files.

The CONSOLE client is not required to operate the PTP system. A user may develop custom client applications to control the PTP using the AV_PTP Remote Interface Library. The remote interface library is available for, Windows NT, and Solaris. For more information about AV-PTP, please refer to Avtec's AV-PTP Programmer's Guide.

The PTP Server supports a single point of control, which means that only one client can control the server at a time. However, multiple clients can simultaneously receive status from a PTP to monitor its operation.

Section 2: Execution Modes

The PTP may be executed in either turnkey mode or manual mode. In manual mode, the PTP executes upon the entry of a command or mouse click. In turnkey mode, the PTP automatically executes upon system startup. The PTP server (PTP NT.EXE) should be started first, before the PTP user interface (CONSOLE.EXE). Multiple PTP Servers can be loaded executed by executing PTP_NT.EXE multiple times. Each PTP Server is numbered incrementally from 1 to differentiate between multiple Servers. The first PTP Server is assigned control port 4000, the second PTP server is assigned control port 4001, and so on.

Manual Mode

To execute the PTP server from the command line window, type the following command:

PTP NT.EXE Start→Programmer→PTP_NT→PTP

To execute the PTP server from the Windows NT desktop, double click on the PTP NT icon.

To execute the CONSOLE client from the command line window, type the following command:

CONSOLE.EXE Start→Programmer→Console

To execute the Console client from the Windows NT desktop, double click on the CONSOLE icon.

Turnkey Mode

To configure the PTP to automatically execute upon system startup, copy the PTP NT icon and CONSOLE icons into the *Startup* folder. Each time the system is powered down and restarted, the PTP will automatically open to a user-specified desktop. To specify a particular desktop to be opened at system startup, use a script file. You can write a script file (*use file name script*.ptp, where * refers to the Server number*) and the server will automatically execute the commands in that script file when you open a server window of the same number. See **Chapter 5**, Practical Examples, as well as **Appendix A**, for more detail.

Section 3: PTP NT Server

The PTP NT server controls the PTP hardware and performs all of the real-time data processing. To open a server, launch from start→menu. A PTP NT Server Window will appear, as shown in **Figure 2-9**.

- *Tx Status* – Bit-mapped activity flag based on programmable activity monitor time; 1=module transmitted a buffer within the activity monitor time out
- *Module 1 Empty* – If a module is loaded, this field name is replaced with module-specific data for the remainder of the display

When modules are loaded, the bottom part of the PTP NT Server Window displays status for a particular module.

You can switch the display among loaded modules by typing the command:

```
view module_number <CR>
```

where module_number is the number (1,2,3, ...) of the module to display.

You can quit the PTP NT application by typing command:

```
quit <CR>
```

Please see **Appendix A** for a detailed list of all commands that can be executed from the PTP NT Server application.

The PTP server is usually loaded on systems that contain one or more AVTEC hardware devices, i.e., MONARCH-E, AT-HSIO2, etc. You can also simulate telemetry data on your desktop without the hardware by using the appropriate PTP modules.

The PTP Server application broadcasts status messages to the network on a multicast address (UDP). Console clients detect these broadcast messages to determine how many PTP systems are on the network.

The PTP Server supports a single point of control which means that only one client can control the server at any one time. However, multiple clients can receive status simultaneously from a PTP to monitor its operation.

Section 4: PTP NT Console

Once a PTP server is running, open a Console Window to control the PTP server. The Console Window is your desktop for creating data processing solutions with the available modules. To open a console window, launch from start→menu. A Console Window will appear, as shown in **Figure 2-10**.

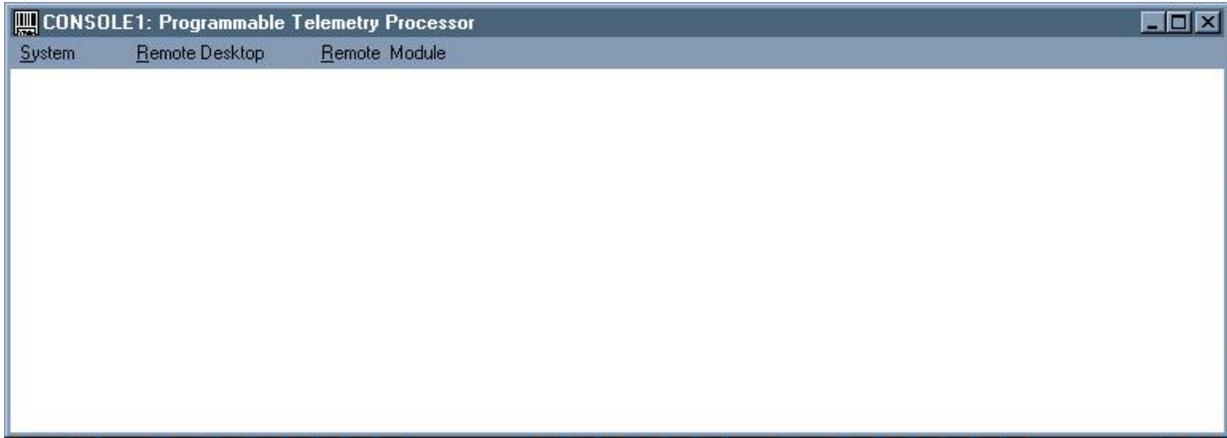


Figure 2-10: Console Window at Startup

The Console Title Bar displays the status of the Console application.

Main Menu

The Main Menu is the PTP's command center. Each option on the main menu calls up a pull down menu. The Main Menu consists of the following options: *System*, *Remote Desktop* and *Remote Module*, as shown in **Figure 2-3**. Each option is discussed further in the paragraphs following.

System Pull Down Menu

The *System* option pull down menu, shown in **Figure 2-11**, provides the capability to perform the operations shown in **Figure 2-11** below.

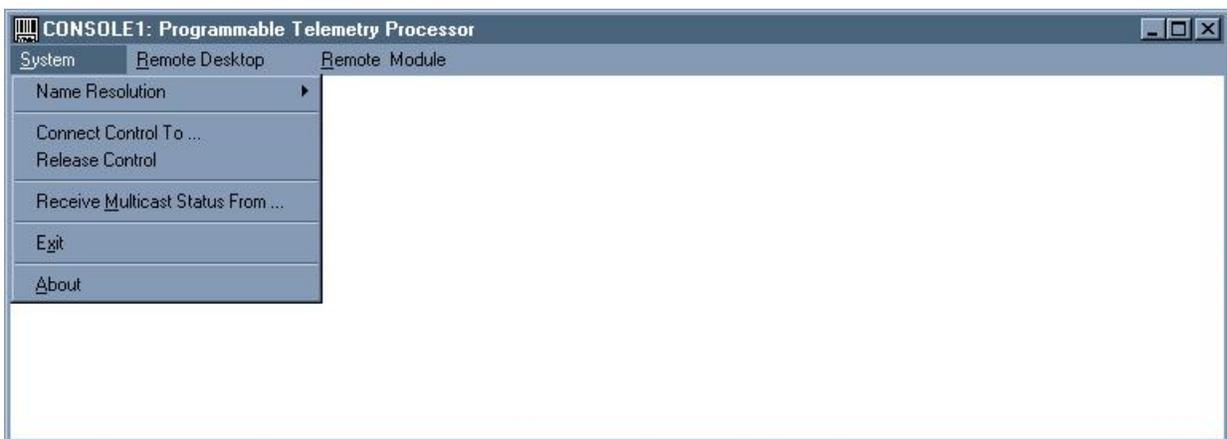


Figure 2-11: System Pull Down Menu

The System menu consists of the following options, as shown in **Table 2-7**:

Table 2-7: System Pull Down Menu Options

Feature	Function Key(s)	Description
Name Resolution	→	Controls the use of name resolution within the Console application. If a domain name server is available, Name Resolution can be enabled to identify PTP servers by name. Otherwise, PTP servers must be identified by IP address.
Connect Control To		Connects to a PTP server. Displays a dialog box with a list of PTP servers on the network.
Release Control		Disconnects from a PTP server
Receive Multicast Status From		Receives status from a PTP server
Exit		Exits PTP NT
About		Provides version number for the console and for PTP_NT Server

Name Resolution under the System menu controls the use of name resolution within the Console application. As shown in **Figure 2-12**, an arrow to the right of a selection indicates that it includes a subsidiary menu. Moving the mouse to the highlighted item and clicking the left button activates the subsidiary menu, which will appear either to the right or left of the pull down menu. **Figure 2-12** shows the *Name Resolution* sub-menu.

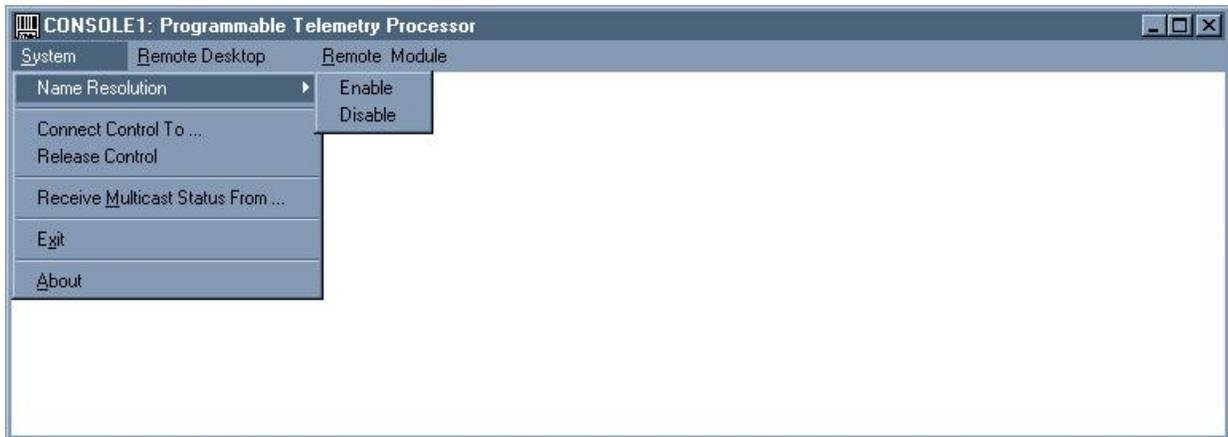


Figure 2-12: Name Resolution Sub-menu

Once you have opened a console window, you may connect to a server. **Note: You must have a server running on your network in order to connect your console to PTP NT.**

To connect to a server, select *System*→ *Connect Control To*. The console will display the *Connect To* window shown in **Figure 2-13**. The *Connect To* window lists the PTP servers that are currently available on the network. The console identifies servers by their IP addresses.



Figure 2-13: Connect To Window

Note: The server availability “list” freezes when you select *Connect Control To*. The *Connect To* window will not notify you if other servers come online after you select *Connect Control To*.

Each list entry in the *Connect To* dialog box contains the following fields:

- *04000* – Control socket port number for the PTP server (PTP Station ID)
- *206.161.148.202* – IP address for the PTP server
- *Names Off* – Indicates that name resolution is disabled. This field would show the host name for the PTP server if name resolution was enabled.
- *0.0.0.0* – IP address of the client that currently controls the PTP server. An address of all zeroes indicates that no client is controlling the PTP server.

Click the *Connect* box to connect the Console to the PTP server. **Note:** *You must connect the console to a PTP server even if both applications are running on the same machine.*

Once the Console has connected to a PTP server, the console window will appear as shown in **Figure 2-14**. If any modules are loaded in the PTP server, the Console Window will display the dialogs for those modules.

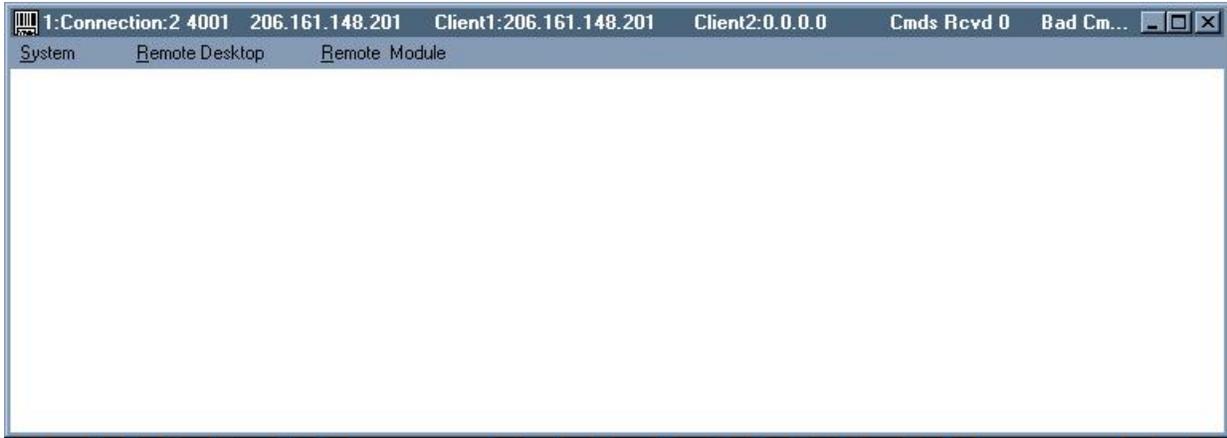


Figure 2-14: Console Window after Connecting to a PTP Server

The Console title bar (*a blue strip across the top of the console window*) contains six information fields, defined below:

- *Not Connected* or *1:Waiting:1* or *1:Connection:1* or *Status* – Indicates the console-to-PTP server status
- *4001* – Station ID of the PTP server, (the TCP control socket port number)
- *206.161.148.201* – Internet address of the PTP server
- *Client 1:206.161.148.201* – Internet address of the Console client
- *Client 2: 0.0.0.0* – No control client on record control port
- *Cnds Rcvd* – Counts the commands received by the PTP NT server
- *Bad Cnds* – Counts the bad commands received by the PTP NT server

Once the console is connected to a PTP server, you can operate the PTP system using the other menu options.

To change servers, pull down the *System* menu and select *Release Control*. Then you can connect to a different server by following the steps above.

To monitor the status of a server, select *System*→*Receive*→*Multicast*→*Status From*. This mode will display the periodic broadcast status from the PTP.

Remote Desktop Pull Down Menu

You control desktop operations within the *Remote Desktop* pull down menu (**Figure 2-15**).

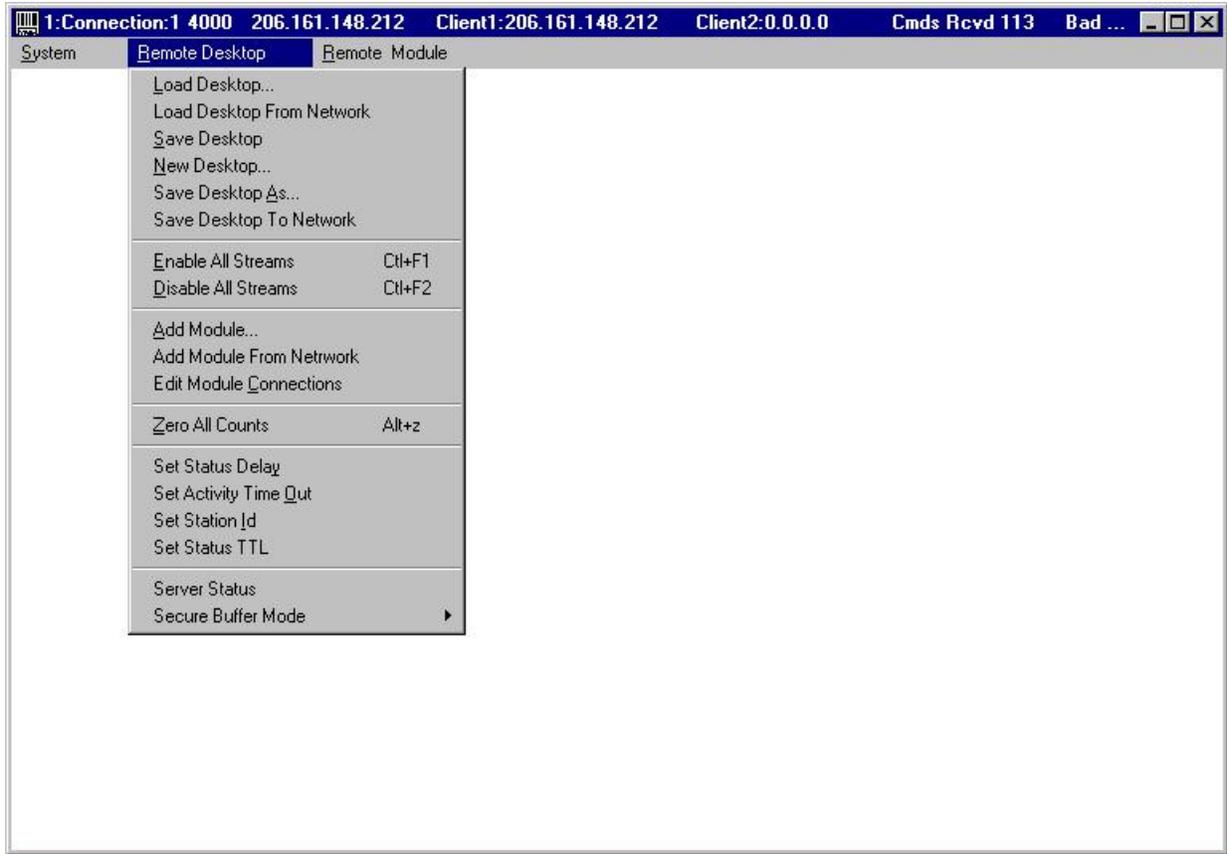


Figure 2-15: Remote Desktop Pull Down Menu

The *Remote Desktop* pull down menu includes the following selections, shown in **Table 2-8**:

Table 2-8: Remote Desktop Pull Down Menu Options

Feature	Function Key(s)	Description
<u>L</u> oad Desktop...		Loads an existing desktop that was saved from an earlier operation. This command searches the \ptp-user\desktops folders for available desktop. It does not provide folder exploring capabilities. If the desktop was saved with the modules enabled, it will resume updating all active data fields upon opening.
<u>L</u> oad Desktop From Network		Same as <i>Load Desktop</i> , but allows the user to search directories on the LAN for desktop files (provides folder exploring).
<u>S</u> ave Desktop...		Saves the current desktop to a .DTP file. Uses the current desktop file name shown on line 2 of the PTP server display.
<u>N</u> ew Desktop...		Generates a completely new desktop, returning the operator to the PTP Opening Window
<u>S</u> ave Desktop <u>A</u> s...		Opens Save Desktop Directory/File Window to save the current desktop to a new .DTP file in the \ptp-user\desktops folder.
<u>S</u> ave Desktop To Network		Same as <i>Save Desktop As</i> , but allows the user to save the desktop anywhere on the local computer or LAN.
<u>E</u> nable All Streams...	Ctl+F1	Enables all of the modules on the desktop

Disable All Streams...	Ctrl+F2	Disables all of the modules on the desktop
Add Module...		Opens Load Module Directory/File Window for user to select a module to add to the desktop from the \ptp-nt\Modules folder on the local machine.
Add Module From Network		Same as <i>Add Module</i> , but allows the user to select modules from anywhere on the local computer or LAN.
Edit Module Connections		Edits the data and event connections for the modules loaded on the desktop.
Zero All Counts...	Alt+z	Zeroes counters in all windows, active or not
Set Status Delay		Sets the time delay in milliseconds (msec) between periodic status updates by the PTP server. The default value is 1000 msec.
Set Activity Time Out		Sets time out the interval for the PTP server activity monitor. The activity monitor checks for data activity in the loaded modules and displays the results in the bitmap Rx status and Tx station in the PTP NT server. The default value is 1000 msec.
Set Station Id		Sets the Station ID for the PTP server
Set Status TTL		Sets the Time-to-Live (TTL) (i.e. number of hops) for a status message before the message is killed. This is a global command and applies to all module status messages. The default status TTC is 1 (status message will not be ?????)
Server Status (see figure 2-9)		Displays the PTP NT server status
Secure Buffer Mode		Can be enabled or disabled. When enabled, whenever a module sends one data stream to multiple modules, creates a copy of the data stream for each module. When disabled, whenever a module sends one data stream to multiple modules, the same file pointer location is given to all modules.

More information about working with desktops is provided in **Chapter 3: Working with Modules And Desktops**.

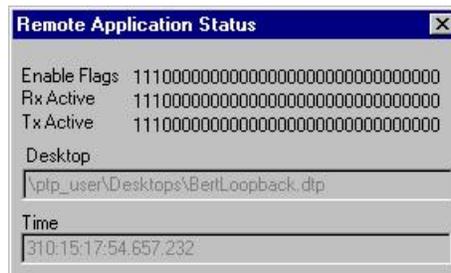


Figure 2-16: Server Status

Remote Module Pull Down Menu

Use the *Remote Module* pull down menu option to perform operations on windows or modules that are currently highlighted, as shown in **Figure 2-17**.

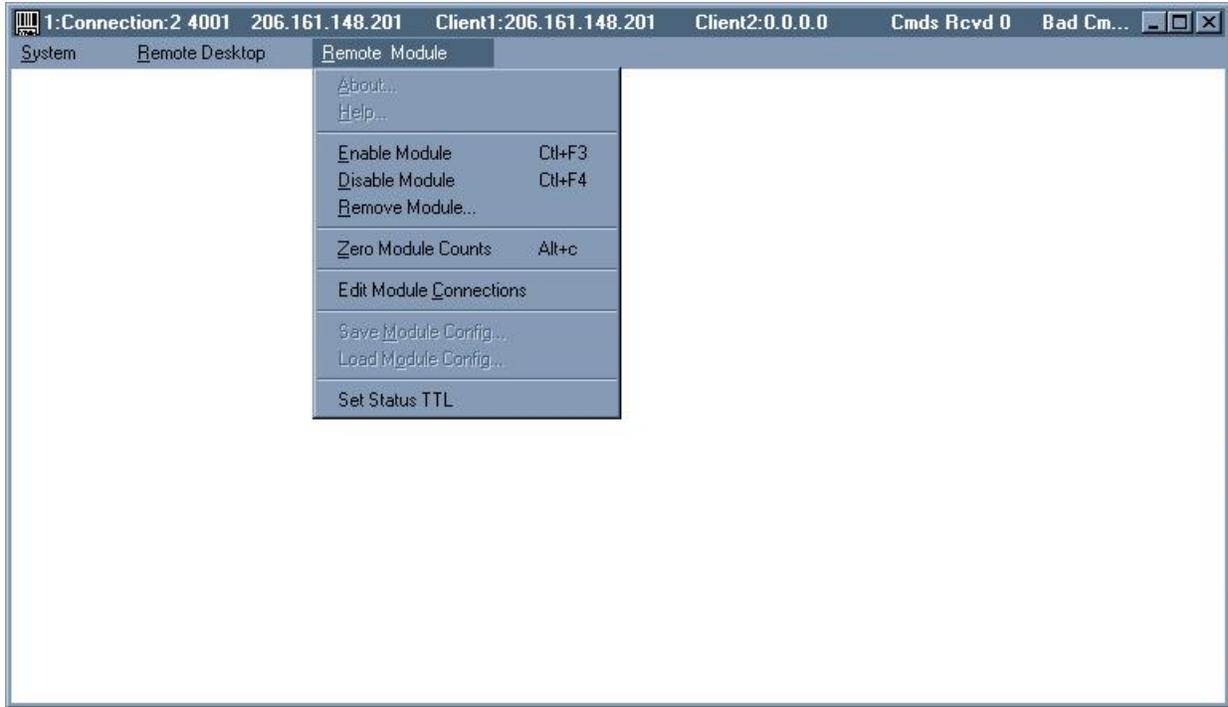


Figure 2-17: Remote Module Pull Down Menu

Table 2-9 illustrates the selections available in the Remote Module pull down menu:

Table 2-9: Remote Module Pull Down Menu Options

Feature	Function Key(s)	Description
About...		not implemented
Help...		not implemented
Enable Module...	Ctrl+F3	Enables the selected module
Disable Module...	Ctrl+F4	Disables the selected module
Remove Module...		Removes the selected module.
Zero Module Counts...	Alt+c	Zeroes the status counts for the selected module
Edit Module Connections		Edits the data and event connections for the selected module
Save Module Config...		not implemented
Load Module Config...		not implemented
Set Status TTL		Sets the Time-to-Live (TTL) (i.e. number of hops) for a status message before the message is not forwarded by an IP ?????. This applies only to the selected module. The default TTL is 1.

More information about working with modules is provided in **Chapter 3: Working with Modules And Desktops**.

Chapter 3

WORKING WITH MODULES AND DESKTOPS

The user creates a data acquisition and processing solution using the PTP desktop. The PTP supports the acquisition, processing, and routing of multiple, simultaneous data streams using a combination of software modules, I/O boards, and PC peripherals. PTP data acquisition and processing is completely software configurable.

The basic operational unit in the PTP is a module. A module is a separately maintained run-time link program that performs user-specific processing in real time. Modules are the elements that work together to form a PTP desktop. Each module has unique capabilities and settings.

The PTP operates three types of modules: Auxiliary I/O, Data I/O, and Data Processing. The user creates a desktop by loading a combination of modules to perform specific tasks.

Auxiliary I/O Modules control and monitor auxiliary I/O boards such as a Bit Synchronizer, a BPSK modulator, a BPSK demodulator, or a Time Code Processor. These modules do not transfer real-time data streams into or out of the system. Only control and status information is passed between the CPU and the auxiliary I/O board.

Data I/O Modules control the data I/O devices in the system, including network interfaces, file devices, and the AT-HSIO2 or MONARCH-E boards. The Data I/O Modules transfer real-time data streams into or out of the system using the application-specific interfaces and standard PC peripherals.

Data Processing Modules perform some software processing on data as it flows through the module. An example is the CCSDS VCP which filters frame data based on VCID and data quality.

The collection of modules required to perform a specific task is called a desktop.

Creating a desktop consists of four steps:

1. Connect the Console in control mode
2. Load the modules required to perform the desired tasks
3. Configure each of the modules
4. Edit each of the modules' data and event connections
5. Save the desktop configuration

Section 1: Creating a Desktop

Using the *Remote Desktop* pull down menu, you can create, load, or save a completely customized desktop. **Figure 3-1** shows the *Remote Desktop* pull down menu.

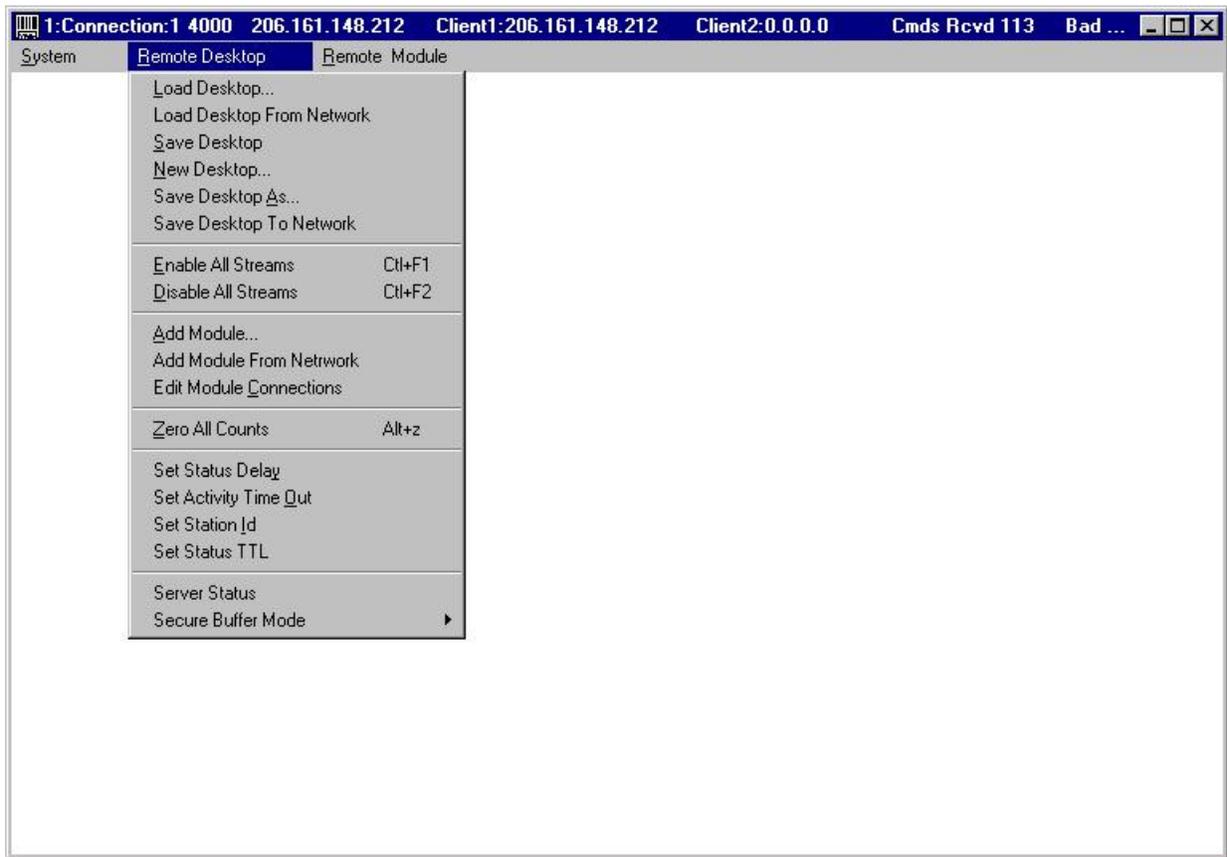


Figure 3-1: Remote Desktop Pull Down Menu

To create a new desktop, select *Remote Desktop*→*New Desktop...* from the main PTP menu. A blank console window will appear, as shown in **Figure 3-2**.



Figure 3-2: Blank Desktop Window

If you are already working in one desktop, and create another (*new*) desktop, the PTP will clear any modules currently displayed in the PTP window and replace them with a new desktop and blank console window (**Figure 3-2**). This initial window is the canvas on which you create your customized Windows application using PTP modules.

Adding a Module

You can define the data flow of your desktop by loading and connecting modules. Select *Remote Desktop* → *Add Module...* to add a module to your desktop. A *Load Module* window will appear, as shown in **Figure 3-3**.

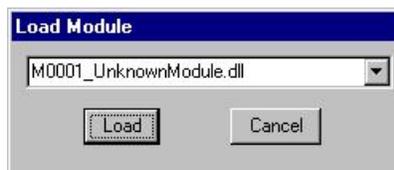


Figure 3-3: Add Module Dialog Window

The procedure below allows you to add (*also called load*) a module.

1. From the main PTP menu, select *Remote Desktop*.

2. From the *Remote Desktop* pull down menu, select *Add Module*.
3. The *Load Module* window (**Figure 3-3**) will appear, with all access windows activated.
4. Select the directory from which you want to load a module.
5. Select the module to be opened (<filename>.DLL).
6. Select *Open*.

Table 3-1 lists the available modules and their functions.

Table 3-1: Available Module Functions

Module	Function
Data I/O Modules	
Avtec Rate Adjust	Serial transmit to AT-HSIO2 or MONARCH-E PCM Simulator with buffering to allow module output data rate less than module input data rate.
Avtec Serial Input	Serial receive from AT-HSIO2 or MONARCH-E Frame Synchronizer
Avtec Serial Output	Serial transmit to AT-HSIO2 or MONARCH-E PCM Simulator
CPU Timer	Provides an event output at a user-configurable rate. Useful for simulating output end-of-frames when no hardware is available in the system.
Com Port Transceiver	Allows sending and receiving of data via PC's Com Ports.
Dump to Line Printer (DumptoLP)	Allows formatted data output to a printer or file.
File PlayBack	Replays a previously recorded data file
File Recorder	Records the output of a module to disk
File Spooler (Spooler)	Allows spooling of data to a file for playback at a slower/faster rate.
Network Sockets (Sockets)	Sends or receives data over a network
Network Sockets Ver 2 (Sockets Ver 2)	TCP Server mode only socket with support for multiple TCP Clients.
Auxiliary I/O Modules	
Apogee PSK Modulator (ApogeePSK)	Configures and controls the Apogee ISA-PSK BPSK Subcarrier Generator
Apogee Time Code Generator (Time)	Configures and controls the Apogee ISA-STG time board
Aydin Bit Sync (AydinBS)	Configures and controls the Aydin PC335 Bit Synchronizer
Aydin PSK Demodulator (Aydin PSKDemod)	Configures and controls the Aydin PC329 PSK Demodulator
GDP PSK Modulator (GDP PSKMod)	Configures and controls the GDP PSK004 PSK Modulator
Microdyne PCR2000 (PCR2000)	Configures and controls the Microdyne PCR2000 Receiver/Demodulator

National Instruments GPIB (GPIB)	Configures and controls the National Instruments GPIB Interface Board
Odetics GPSunc-ISA Time Board (Odetics)	Configures and controls the Odetics GPSync-ISA Time Board
Veda Bit Sync (VedaBitSync)	Configures and controls the Veda bit synchronizer
Data Processing Modules	
8 Bit Sorter	Sorts packets based on 8 bit value in a user-specified byte location
16 Bit Sorter	Sorts packets based on 16 bit value in a user-specified byte location
Best Source Select (BSS)	Used to compare identical incoming streams and output "best" stream based on frame count
Bit Density Correction	Guarantees signal transitions in a bit stream by XORing stream with 2047 pseudo-random pattern
Bit Error Rate Tester	Sends (and receives and verifies) a pseudorandom 2047-bit pattern to quality check a communications link
CCSDS Virtual Channel Processor	Processes and displays statistics on CCSDS VCDUs
CCSDS Virtual Channel Simulator	Provides a user-configurable source of CCSDS VCDUs
Error Inject	Induces errors in a data stream
Gather Scatter	Extracts user-specified bytes from a data stream
In-Line UNZip	Performs data decompression
In-Line Zip	Performs data compression
Null Transceiver	Provides data display and muxing capabilities
Packet Processor	Processes and displays statistics on CCSDS packetized telemetry
TDM Simulator (TDMSim)	Provides user-configurable source of TDM telemetry
Unknown Module	Loaded automatically when a module exists on the PTP server but not on the machine running Console
Encapsulation/Extraction Modules	
ACE SFDU Formatter	Outputs an ACE SFDU formatted data stream
Ames Command Encapsulator	Adds configurable fields to commands in preparation for transmission
AXAF SFDU Formatter	Outputs an AXAF SFDU formatted data stream
DS-T SFDU Formatter	Outputs a DS-T SFDU formatted data stream
EDOS Service Header	Prepends EDOS Service Header to a buffer
IPDU Formatter	Prepends an IPDU header to a buffer
IPDU Receiver	Receives IPDU formatted packets from the network
LEO-T CDH Formatter	Prepends LEO-T Command Delivery Header to a command buffer
LEO-T CDH Receiver	Receives LEO-T formatted commands
LEO-T TFDH	Prepends the LEO-T Telemetry Frame Delivery Header to a telemetry buffer
NASCOM Blocker	Outputs NASCOM 4800 Bit Blocks
NASCOM Deblocker	Receives NASCOM 4800 Bit Blocks from the network and outputs a serial data stream
NASCOM RTP Formatter	Receives a 4800BB and outputs an IP encapsulated frame
NASCOM RTP Receiver	Receives an IP encapsulated frame and outputs a

	NASCOM 4800BB.
--	----------------

When you have loaded several modules, the Console Window will appear as shown in (Figure 3-4).

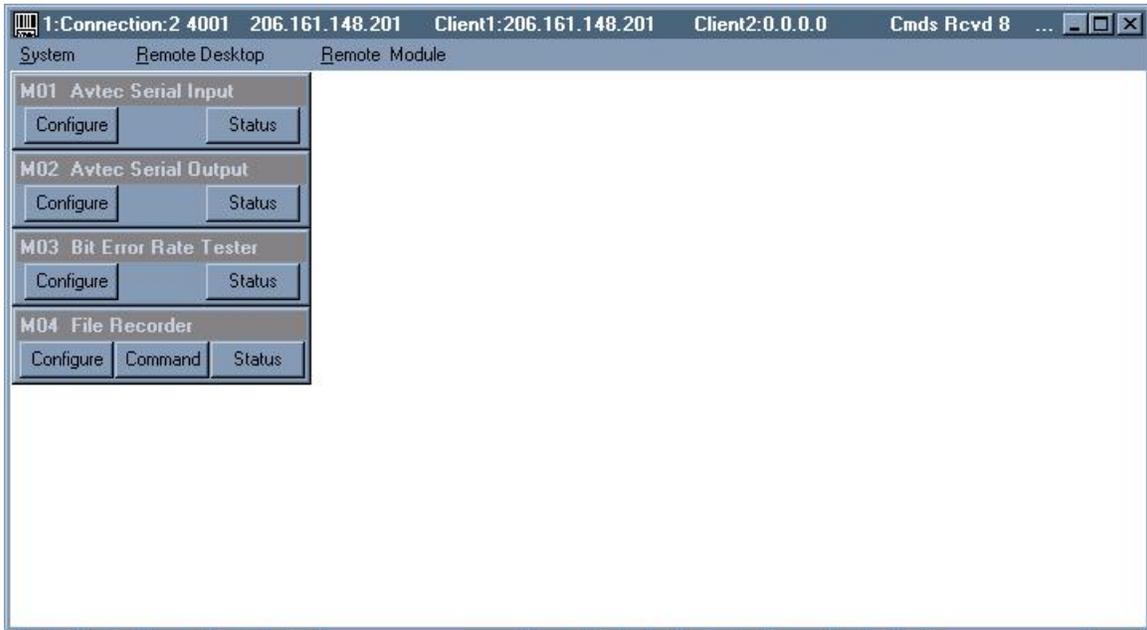


Figure 3-4: Console Window with Modules Loaded

The Title Bar for each module window contains the module number (*e.g.*, *M01*) and the module name (*e.g.*, *AVTEC Serial Input*). Module windows may contain up to three buttons to control and monitor the module: *Configure*, *Command*, and *Status*.

The *Configure* button displays the module configuration dialog. The configuration dialog is used to set the module parameters. The configuration of each module type is described in **Chapter 4: PTP Modules**.

The *Command* button displays the module command dialog. The command dialog is used to send commands to the module. The commands for each module type are described in **Chapter 4: PTP Modules**.

The *Status* button displays the module status dialog. The status dialog displays the status of the module. The status for each module type is described in **Chapter 4: PTP Modules**.

Working with a Highlighted Module

The term *highlighted module* generally refers to the module you were most recently working with. You can highlight a module (the Title Bar will turn blue) by simply clicking on it with the left mouse button. Once a module has been highlighted, you can perform a variety of operations

on it by accessing the *Remote Module* pull down menu in the PTP main menu. The functions you access from the *Remote Module* pull down menu can only be applied to the currently selected module window. **Figure 3-5** shows the *Remote Module* pull down menu.

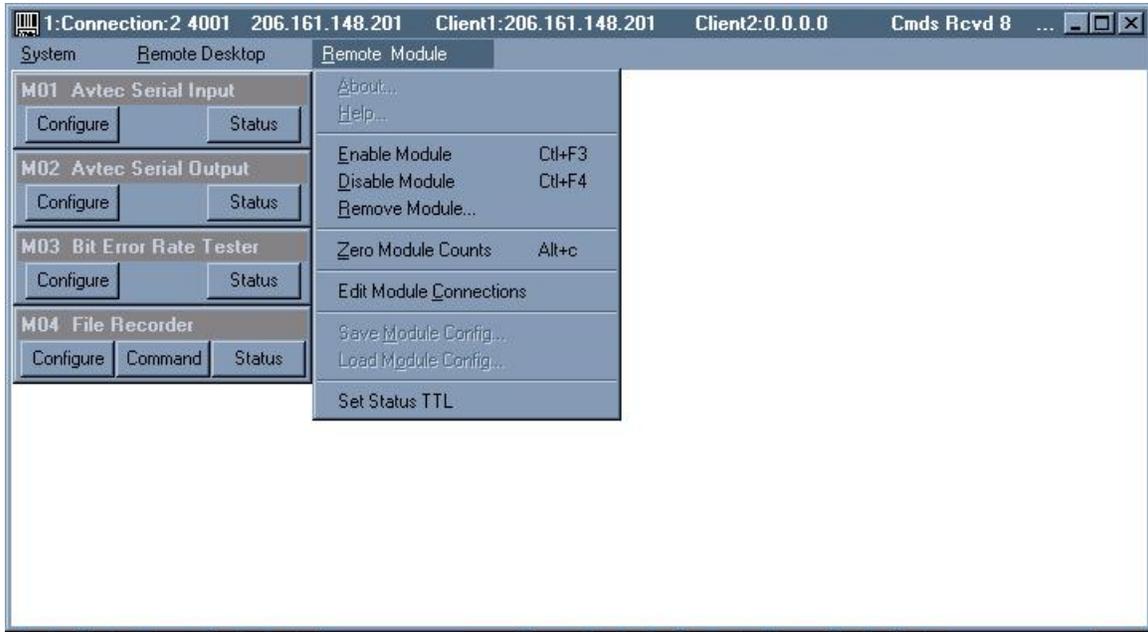


Figure 3-5: Remote Module Pull Down Menu

When working with the *Remote Module* pull down menu, you control the individual modules in the following ways:

- *Enable Module* – Enables the selected module
- *Disable Module* – Disables the selected module
- *Remove Module* – Removes the selected module from the desktop (*not currently implemented*)
- *Zero Module Counts* – Clears all of the status counters for the selected module
- *Edit Module Connections* – Edits the data and event connections for the module as described below

Connecting Modules

Data I/O Modules and Data Processing Modules are “connected” together to perform real-time data processing tasks. The user selects the data flow between the various modules. Each Data Module has data inputs and outputs and an event input and output.

To edit module connections, select *Remote Module* → *Edit Module Connections*. An *Edit Module Connections* window will appear, as shown in **Figure 3-6**.

From the main PTP menu, select Remote Desktop.

1. From the *Remote Desktop* pull down menu, select *Save Desktop As*.
2. The *Save Desktop* window (**Figure 3-7**) will appear, with all access windows activated.
3. Select the directory to which you want to save your desktop.
4. Type a file name in the *File name* window. We suggest you maintain the naming convention of .DTP (<filename>.DTP) for the file extensions.
5. Select *Save*.

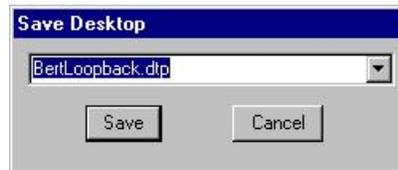


Figure 3-7: Save Desktop Window

Save Desktop As also allows you to rename the current desktop as a new file, thereby preserving the original as well.

Save Desktop saves the current configuration setting to the currently defined file name. Note: No warning is given before overwriting a current file. *Save Desktop to Network* allows you to save the current desktop to a folder located anywhere on the local machine or LAN.

Loading a Saved Desktop

(The *Load Desktop* option allows you to load previously stored desktops).

1. From the main PTP menu, select *Remote Desktop*.
2. From the *Remote Desktop* pull down menu, select *Load Desktop*.
3. The *Load Desktop* window (**Figure 3-8**) will appear, with all access windows activated.
4. Select the directory from which you want to load a desktop.
5. Select the desktop to be opened (<filename>.DTP).
6. Select *Open*.



Figure 3-8: Load Desktop Window

An existing desktop saved from an earlier operation will be loaded. If the desktop was saved with the modules enabled then, upon opening, it will resume updating active data fields.

Load Desktop from Network allows you to load a desktop from any folder on the local machine from a LAN

Section 3: Other Desktop Functions

To control data flow, use the following commands, located in the *Remote Desktop* main PTP menu:

- *Enable All Streams* – Allows you to enable all of the modules on your desktop at once. Selecting this command disables all streams and then enables them all again.
- *Disable All Streams* – Allows you to disable all modules. To restart data flows, select *Enable All Streams*.
- *Zero All Counts* – Returns all counters on the desktop to zero.
- *Set Status Delay* – Sets the delay time between multicast module status messages in milliseconds (minimum is 100, default is 1000). To set status delay:
 1. From the main PTP menu, select *Remote Desktop*.
 2. From the *Remote Desktop* pull down menu, select *Set Status Delay*.
 3. A *Dialog* window (**Figure 3-9**) will appear.
 4. Enter the desired delay time.
 5. Select *Set*.



Figure 3-9: Dialog Window

- *Set Activity Time Out* – Sets the maximum time between module inputs/outputs before the activity monitor will indicate no data. To set activity timeout, select *Set Activity Time-Out* and follow the steps above. Default is 1000 msec.
- *Set Station ID* – Allows you to specify the Station ID for the PTP Server. To set the Status ID, select *Set Station ID* and follow the steps above.
- *Set Status TTL* – Allows the user to set the Time-to-Live (TTL) for status messages, i.e. the number of hops a status message takes before it is killed. To set the status message TTL, select *Set Status TTL* and follow the steps above.
- *Secure Buffer Mode (Enable/Disable)* – This command enables or disables *Secure Buffer Mode* operation of the PTP. When data is passed from one module to another, what is actually passed is a “pointer” to the data. If the same data is passed to multiple

modules, all modules that are to receive the data will get the same pointer. This can be a problem if any one of these modules is going to alter the data in any way. With *Secure Buffer Mode* enabled, when one data stream is passed to multiple modules, a copy of the data is made for each module. In this case, if one module changes the data, it will not affect the data being used by the other modules. To enable or disable *Secure Buffer Mode*, select *Secure Buffer Mode* and then click on either *Enable* or *Disable*.

Chapter 4

PTP MODULES

The basic operational unit in the PTP is a module. A module is a separately maintained run-time link program that performs user-specific processing in real time. Modules are the elements that work together to form a PTP desktop. Each module has unique capabilities and settings. There are three types of PTP modules: Auxiliary I/O, Data I/O, and Data Processing.

Auxiliary I/O Modules control and monitor auxiliary I/O boards such as a Bit Synchronizer, a BPSK modulator, a BPSK demodulator, or a Time Code Processor. These modules do not transfer real-time data streams into or out of the system. Only control and status information is passed between the CPU and the auxiliary I/O board.

Data I/O Modules control the data I/O devices in the system, including network interfaces, file devices, and the AT-HSIO2 or MONARCH-E boards. The Data I/O Modules transfer real-time data streams into or out of the system using the application-specific interfaces and standard PC peripherals.

Data Processing Modules perform software processing on data as it flows through the module. An example is the CCSDS Virtual Channel Processor which filters frame data based on VCID and data quality.

Module	Function
Data I/O Modules	
Avttec Rate Adjust	Serial transmit to AT-HSIO2 or MONARCH-E PCM Simulator with buffering to allow module output data rate less than module input data rate.
Avttec Serial Input	Serial receive from AT-HSIO2 or MONARCH-E Frame Synchronizer
Avttec Serial Output	Serial transmit to AT-HSIO2 or MONARCH-E PCM Simulator
CPU Timer	Provides an event output at a user-configurable rate. Useful for simulating output end-of-frames when no hardware is available in the system.
Com Port Transceiver	Allows sending and receiving of data via PC's Com Ports.
Dump to Line Printer (DumptoLP)	Allows formatted data output to a printer or file.
File PlayBack	Replays a previously recorded data file
File Recorder	Records the output of a module to disk

Module	Function
File Spooler (Spooler)	Allows spooling of data to a file for playback at a slower/faster rate.
Network Sockets (Sockets)	Sends or receives data over a network
Network Sockets Ver 2 (Sockets Ver 2)	TCP Server mode only socket with support for multiple TCP Clients.
Auxiliary I/O Modules	
Apogee PSK Modulator (ApogeePSK)	Configures and controls the Apogee ISA-PSK BPSK Subcarrier Generator
Apogee Time Code Generator (Time)	Configures and controls the Apogee ISA-STG time board
Aydin Bit Sync (AydinBS)	Configures and controls the Aydin PC335 Bit Synchronizer
Aydin PSK Demodulator (Aydin PSKDemod)	Configures and controls the Aydin PC329 PSK Demodulator
GDP PSK Modulator (GDP PSKMod)	Configures and controls the GDP PSK004 PSK Modulator
Microdyne PCR2000 (PCR2000)	Configures and controls the Microdyne PCR2000 Receiver/Demodulator
National Instruments GPIB (GPIB)	Configures and controls the National Instruments GPIB Interface Board
Odetics GIPSunc-ISA Time Board (Odetics)	Configures and controls the Odetics GPSync-ISA Time Board
Veda Bit Sync (VedaBitSync)	Configures and controls the Veda bit synchronizer
Data Processing Modules	
8 Bit Sorter	Sorts packets based on 8 bit value in a user-specified byte location
16 Bit Sorter	Sorts packets based on 16 bit value in a user-specified byte location
Best Source Select (BSS)	Used to compare identical incoming streams and output "best" stream based on frame count
Bit Density Correction	Guarantees signal transitions in a bit stream by XORing stream with 2047 pseudo-random pattern
Bit Error Rate Tester	Sends (and receives and verifies) a pseudorandom 2047-bit pattern to quality check a communications link
CCSDS Virtual Channel Processor	Processes and displays statistics on CCSDS VCDUs
CCSDS Virtual Channel Simulator	Provides a user-configurable source of CCSDS VCDUs
Error Inject	Induces errors in a data stream
Gather Scatter	Extracts user-specified bytes from a data stream
In-Line UNZip	Performs data decompression
In-Line Zip	Performs data compression
Null Transceiver	Provides data display and muxing capabilities
Packet Processor	Processes and displays statistics on CCSDS packetized telemetry
TDM Simulator (TDMSim)	Provides user-configurable source of TDM telemetry
Unknown Module	Loaded automatically when a module exists on the PTP server but not on the machine running Console

Module	Function
Encapsulation/Extraction Modules	
ACE SFDU Formatter	Outputs an ACE SFDU formatted data stream
Ames Command Encapsulator	Adds configurable fields to commands in preparation for transmission
AXAF SFDU Formatter	Outputs an AXAF SFDU formatted data stream
DS-T SFDU Formatter	Outputs a DS-T SFDU formatted data stream
EDOS Service Header	Prepends EDOS Service Header to a buffer
IPDU Formatter	Prepends an IPDU header to a buffer
IPDU Receiver	Receives IPDU formatted packets from the network
LEO-T CDH Formatter	Prepends LEO-T Command Delivery Header to a command buffer
LEO-T CDH Receiver	Receives LEO-T formatted commands
LEO-T TFDH	Prepends the LEO-T Telemetry Frame Delivery Header to a telemetry buffer
NASCOM Blocker	Outputs NASCOM 4800 Bit Blocks
NASCOM Deblocker	Receives NASCOM 4800 Bit Blocks from the network and outputs a serial data stream
NASCOM RTP Formatter	Receives a 4800BB and outputs an IP encapsulated frame
NASCOM RTP Receiver	Receives an IP encapsulated frame and outputs a NASCOM 4800BB.

The Console presents a base window for each module similar to the one shown in **Figure 4-1**.



Figure 4-1: Example Console Module Window

The Title Bar for each module window contains the module number (*e.g.*, *M01*) and the module name (*e.g.*, *File Recorder*). Module windows may contain up to three buttons to control and monitor the module: *Configure*, *Command*, and *Status*.

The *Configure* button displays the module configuration dialog. The configuration dialog is used to set the module parameters.

The *Command* button displays the module command dialog. The command dialog is used to send commands to the module.

The *Status* button displays the module status dialog. The status dialog displays the status of the module.

The Console continuously receives configuration and status information from the PTP Server multicast output. To change any of the settings for any module, perform the following steps:

1. Click the module window *Configure* button to display the config dialog
2. Make changes to the Parameters (the Updates check box will automatically clear)

3. Click the Send button to apply the new parameters (*the update box will automatically be re-enabled*)

Note: Parameter changes do not take effect until the “send” button is checked.

DATA I/O MODULES

Section 1: Avtec Rate Adjust Module

The Avtec Rate Adjust Module provides a menu to output serial data stream where the onboard transmit frequency is dynamically adjusted to match the rate of the incoming data. The Rate Adjust Module creates a queue of data buffers in system memory. The transmit clock frequency is adjusted up or down based on the amount of data in the queue. Output parameters are identical to the Avtec Serial Output Module. Additional options include buffer depth control and rate damping control. The Rate Adjust Module is typically used with the NASCOM Deblocker or Scatter/Gather Modules to reconstruct a serial data stream that was encapsulated for transmission across a network.

The Rate Adjust Module communicates with the I/O board using the AV_FSTS Frame Synchronizer/Telemetry Simulator driver for Windows NT. The AV_FSTS driver refers to the boards by channel number (*Channel 0, 1, etc.*). The mapping between AV_FSTS channel number and physical I/O board is defined in the Windows NT registry using the FSTS_CONFIG.EXE utility.

Figure 4-2 shows the Rate Adjust Module window.



Figure 4-2: Avtec Rate Adjust Module

Figure 4-3 shows the Config Rate Adjust Module window.

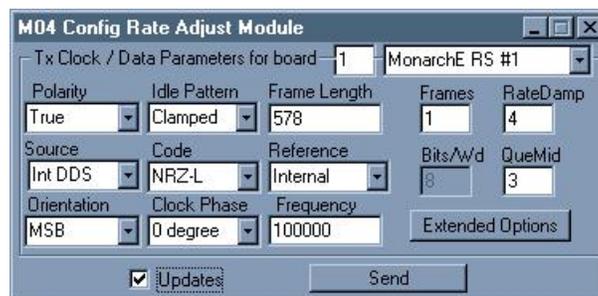


Figure 4-3: Rate Adjust Config Window

Table 4-1 defines the parameters in the Config Rate Adjust Module window.

Table 4-1: Rate Adjust Config Parameters

Parameter	Description
Tx Clock/Data Parameters for board	Defines which board is to be used by the Rate Adjust Module
Polarity	Configures Data Polarity: True or Inverted
Idle Pattern	Specifies idle transmission: Clamped (all zeros) or Toggle (alternating 0/1 pattern)
Frame Length	Frame length in words (or bytes if bits per word = 8)
Source	Clock source: Internal DDS, Internal PLL, External TTL, or External RS-422
Code	Output code type: NRZ-L/M/S or B10-L/M/S
Reference	Reference source for the Internal DDS: Internal, External TTL, or External RS-422
Orientation	MSB first or LSB first
Clock Phase	0 degrees (data transitions on rising edge of the clock) or 180 degrees (data transitions on the falling edge of the clock)
Frequency	Output Data Rate in Hz
Bits/Wd	Number of bits per word (4-16), default = 8
QueMid	Minimum number of buffers that Rate Adjust Module will try to maintain in queue
RateDamp	Variable that defines the quickness in adjusting clock from input data rate to output data rate, valid values are 0 – 32: 0 = fixed rate, 1 = minimum number of steps, 32 = maximum number of steps
Frames	Number of frames per buffer

Figure 4-4 shows the Rate Adjust Module's Extended Configuration window. This window allows configuration of CCSDS related parameters, including enabling and disabling of Randomization and CRC, Reed Solomon, and Convolutional encoding. These are available only when using the MONARCH-E Frame Synchronizer/Telemetry Simulator board.

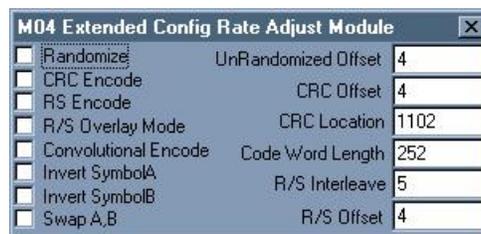


Figure 4-4: Avtec Rate Adjust Extended Configuration Window

Table 4-2 defines the parameters available in the Rate Adjust Extended Configuration window.

Table 4-2: Rate Adjust Extended Configuration Parameters

Parameter	Description
Randomize	Controls the pseudo-randomizer. Checked is enabled.
CRC Encode	Controls the CRC encoder. Checked is enabled.
RS Encode	Controls the RS encoder. Checked is enabled.
R/S Overlay Mode	The MONARCH-E RS encoder can append the RS check symbols (32 per codeword) to the output frame (not checked) or overwrite part of the output frame with the RS check symbols (checked)
Convolutional Encode	Controls the MONARCH-E rate 1/2, constraint length 7 convolutional encoder. Checked enables the convolutional encoder. Note the transmit clock rate corresponds to the symbol rate (or 1/2 the bit rate) when the convolutional encoder is enabled.
Invert SymbolA	Controls the inversion of the convolutional encoder SymbolA (or G1 = 1111001) output. Checked inverts G1 before output.
Invert SymbolB	Controls the inversion of the convolutional encoder SymbolB (or G2 = 1011011) output. Checked inverts G2 before output.
Swap A,B	Controls the order the convolutional encoder symbols are transmitted. Unchecked, SymbolA (G1) is transmitted before SymbolB (G2). Checked, SymbolB (G2) is transmitted before SymbolA (G1).
UnRandomized Offset	Defines the randomizer offset from the start of the frame. To exclude a 32-bit sync pattern from randomization, this value should be set to 4.
CRC Offset	Defines the starting point for CRC calculations. To include the sync pattern, set this value to 0. To exclude a 32-bit sync pattern, set this value to 4.
CRC Location	Defines the location where the 16-bit CRC remainder should be written in the frame
Code Word Length	Defines the Reed-Solomon code word length from 33 to 255.
R/S Interleave	Defines the RS interleave depth from 1 to 8.
R/S Offset	Defines the starting point of the RS codeblock within the frame.

Figure 4-5 shows the Rate Adjust Module Status window.

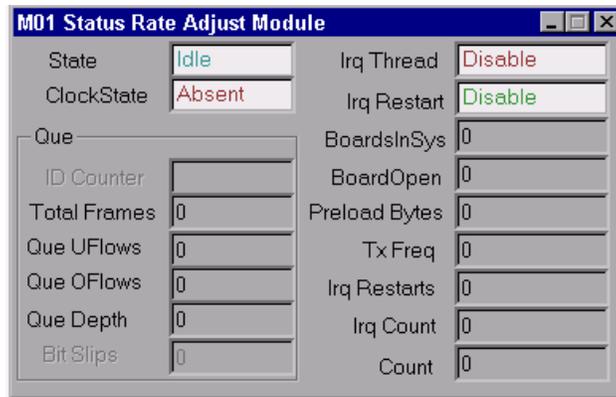


Figure 4-5: Rate Adjust Status Window

Table 4-3: Rate Adjust Status Parameters

Parameter	Description
State	Date state: Idle or Transmit
ClockState	Absent or Present
Irq Thread	Displays Rate Adjust Output Thread state
Irq Restart	Displays Automatic Restart State. Enabled on error.
BoardInSys	Displays number of AT-HSIO2 and MONARCH-E boards in system as recorded in registry
BoardOpen	Specifies current board in use
Preload Bytes	Number of bytes that are preloaded in the AV_FSTS driver
Tx Freq	Current output frequency in Hz
Irq Restarts	Count of Rate Adjust Serial Output thread restarts on error
Irq Count	Count of interrupts
Count	Count of frames written to AV_FSTS driver

Table 4-4 defines the parameters in the Que subsection of the Rate Adjust Module Status window.

Table 4-4: Rate Adjust Que Status Parameters

Parameter	Description
ID Counter	(Future)
Total Frames	Count of frames transmitted
Que UFlows	Number of occurrences of no data
Que OFlows	Number of occurrences that the Rate Adjust went over the maximum allowable number of buffers
QueDepth	Number of buffers open
Bit Slips	Number of bit slips

Section 2: AVTEC Serial Input Module

The Serial Input module supports the Frame Synchronizer portion of the AT-HSIO2 and the Frame Synchronizer/Reed-Solomon Decoder portion of the MONARCH-E. The Serial Input module reads frame data from the I/O board and outputs buffers with frame data and quality annotation to other modules.

This section provides an overview of the serial input control and status functions. Please refer to the AT-HSIO2 or MONARCH-E Functional Descriptions (References #1 and 4) for more detailed information on particular parameters.

The Serial Input Module communicates with the I/O board using the AV_FSTS Frame Synchronizer/Telemetry Simulator driver for Windows NT. The AV_FSTS driver refers to the boards by channel number (*Channel 0, 1, etc.*). The mapping between AV_FSTS channel number and physical I/O board is defined in the Windows NT registry using the FSTS_CONFIG.EXE utility.

Figure 4-6 shows the dialog that appears after loading an instance of the Serial Input Module.



Figure 4-6: AVTEC Serial Input Module Window

When you click the configure button, a Config AVTEC Serial Input window will appear, as shown in **Figure 4-7**. The configuration parameters shown in **Figure 4-7** apply to both the AT-HSIO2 and the MONARCH-E.

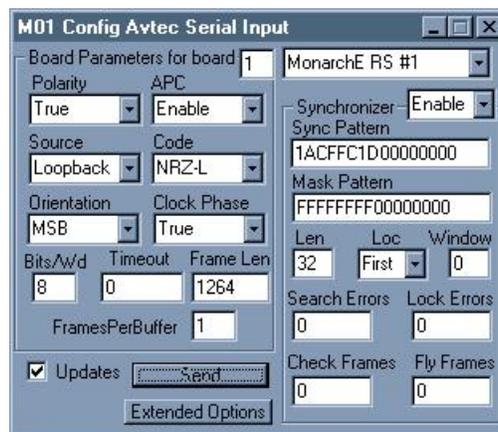


Figure 4-7: Config AVTEC Serial Input Window

The entry after 'Board Parameters for board' defines which AV_FSTS channel, and thus which I/O board, is assigned to this module. In **Figure 4-7**, this value is 0 which means that no board is assigned to the module. **The board number should be set to the desired AV_FSTS channel number plus one. For example, to assign the AV_FSTS channel 1 receiver to the serial input module, set the board number to 2. The list box to the right of the board number can also be used to assign a board to this module. The list box indicates whether the selected board is an AT-HSIO2, MONARCH, or MONARCH-E.**

Table 4-5 defines the fields in the Config AVTEC Serial Input Board Parameters sub-menu.

- Board Parameters for Board X sub-menu

Table 4-5: Config AVTEC Serial Input Board Parameters

Parameter	Description
Polarity	Provides two selectable setup values: true or inverted. The set up defines the expected board input data as either true or inverted.
APC	Automatic Polarity Correction (APC) provides two selectable setup values: enable or disable. If enabled, it automatically corrects the data polarity based on the detected sync pattern polarity. If an inverted sync pattern is detected and APC is enabled, the APC logic inverts the data. The Serial Input Status Polarity field will be Normal if the APC is not active and Inverted if the APC logic is active.
Source	Provides three setup values: Ext TTL, Ext 422, and Loopback. Defines the source of the input clock and data. For Ext TTL and Ext 422, the input clock and data are taken from the board's DB37 connector. For Loopback, the board output clock and data are internally connected to the input.
Code	Provides six setup values: NRZ-L, M, and S and Biphase-L,M, and S. Defines the input PCM data type. The AT-HSIO2 and MONARCH-E support Biphase codes by delaying the clock with respect to the data and will only work for Loopback. A Bit Synchronizer is required to support external Biphase data sources.
Orientation	Provides two setup values: MSB and LSB. Defines the operation of the serial to parallel converter. Select MSB for data that was transmitted Most Significant Bit first. Select

Parameter	Description
	LSB for data that was transmitted Least Significant Bit first.
Clock Phase	Provides two setup values: True or Inverted. If True is selected, the Serial Input hardware expects the input data to change on the rising edge of the clock and it will latch the data on the falling edge of the clock. If Inverted is selected, the Serial Input hardware expects the input data to change on the falling edge of the clock and it will latch the data on the rising edge.
Bits/Wd	Provides a default value of eight. Any value from 4 to 16 can be entered. The setup defines the word size that the hardware uses for serial to parallel conversion. For CCSDS telemetry streams, the default value of eight should be used. If Bits/Wd is less than eight, the input data will be stored in bytes. If Bits/Wd is greater than eight, the input data will be store in 16-bit words.
Timeout	Provides a default value of 0. This parameter should be set to 0 for proper operation.
Frame Len	This field defines the number of telemetry words in each frame input. If Bits/Wd is less than or equal to eight, then the Frame Len is equal to the number of bytes in the input frame. If Bits/Wd is greater than eight, then the number of bytes in the input frame is twice the Frame Len. The minimum frame length is 8 bytes. The maximum frame length is 4096 bytes.
Frames Per Buffer	This field defines the number of telemetry frames that the Serial Input Module provides to other modules in a single buffer. The default value is 1 frame per buffer. If this value is greater than one, then the Serial Input Module will gather groups of consecutive telemetry frames before posting a buffer to other modules. This value should be set to keep the buffer rate under 1000 buffers per second. $\text{Frames Per Buffer} = \text{Bit Rate} / (\text{Frame Length in Bits} * 1000)$.

The Synchronizer sub-menu controls the operation of the I/O board frame synchronizer. The frame synchronizer control logic uses an adaptive strategy for acquiring minor frame synchronization which consists of four states: SEARCH, CHECK, LOCK, and FLYWHEEL.

A state diagram of the minor frame synchronizer control logic is shown in **Figure 4-8**. While in the SEARCH state, the board searches the input data stream for an acceptable sync pattern. Once an acceptable pattern is found, the board enters the CHECK or LOCK state depending on the programmed number of check frames. The LOCK state is entered when N consecutive acceptable frames are received. The number of check frames N is programmable from 0 to 15 frames. If an unacceptable frame is received while in the CHECK state, the board returns to the SEARCH state. The board remains in the LOCK state as long as consecutive acceptable frames are received. If an unacceptable frame is received, the board advances to the FLYWHEEL or SEARCH state depending on the programmed number of flywheel frames. If M consecutive unacceptable frames are received while in the FLYWHEEL state, then the board returns to the SEARCH state. The number of flywheel frames M is programmable from 0 to 15 frames. If an acceptable frame is received while in the FLYWHEEL state, then the board returns to the LOCK state. This method of operation ensures that the board will remain in LOCK even when the received data stream is corrupted by random bit errors.

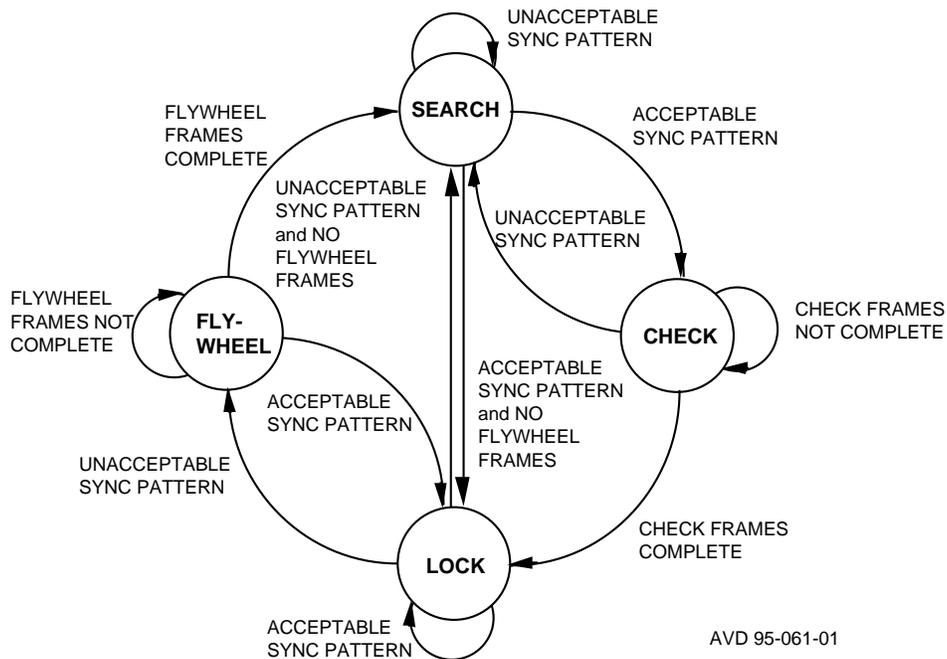


Figure 4-8: Frame Synchronizer State Diagram

Table 4-6 defines the fields in the Config AVTEC Serial Input Synchronizer sub-menu.

Table 4-6: Config AVTEC Serial Input Synchronizer Parameters

Parameter	Description
Synchronizer	Provides two selections: Enable or Disable. If Enable is selected, the Serial Input will only provide frame data buffers to PTP_NT if an acceptable sync pattern is detected

Parameter	Description
	in the input data stream. If Disable is selected, the Serial Input will perform serial to parallel conversion and provide data buffers to PTP_NT as long as the input clock is active without any regard to frame boundaries.
Sync Pattern	Sixteen digit hexadecimal (64-bit) value that defines the frame synchronization pattern. All 16 digits should be entered even if they are zero. The actual bits used for synchronization are defined by the Mask Pattern. The default value of 1ACFFC1D00000000 is the 32-bit sync pattern for CCSDS telemetry streams.
Mask Pattern	Sixteen digit hexadecimal (64-bit) value that defines the sync pattern mask. Sync Pattern bits whose corresponding Mask Pattern bits are zero are ignored by the input correlator. The correlator threshold is calculated as the number of ones in the Mask Pattern minus the number of allowable errors. The default value of FFFFFFFF00000000 is the 32-bit mask pattern for CCSDS telemetry streams.
Len	This field defines the synchronization pattern length in bits. The synchronization pattern length is programmable from 0 to 64 bits. <i>This field is not used and will not affect operation, but may be useful for documentation purposes.</i>
Loc	Provides two selections: First or Last. This field defines the location of the synchronization pattern within the frame. For sync pattern first, the Serial Input will provide data buffers that begin with the sync pattern. The sync pattern and mask pattern should be left-justified in the 64-bit field provided. For sync pattern last, the Serial Input will provide data buffers with the sync pattern at the end of the buffer. The sync pattern and mask should be right-justified in the 64-bit field provided. The Sync Pattern and Mask Pattern should represent how the pattern appears in the serial stream. The Serial Input does not bit reverse the sync pattern or the frame data.
Window	This field defines the size of the frame synchronizer bit slip window. The bit slip window defines the number of bit periods before and after the end of the frame in which the frame synchronizer will look at the output of the correlator. If the Window is set to 0, then the sync pattern for the next frame must occur immediately after the end of the current frame. The Bit Slip Window is programmable from 0 to +/- 3 bit periods.
Search Errors	This field defines the number of allowable bit errors in the synchronization pattern during search AND check modes.

Parameter	Description
	If a frame synchronization pattern is within the acceptable bit error tolerance, it is recognized as frame: it is identified as either a search frame or a check frame depending on the mode. The number of Search Errors is programmable from 0 to 15.
Lock Errors	This field defines the number of allowable bit errors in the synchronization pattern during lock AND flywheel modes. If a frame synchronization pattern is within the acceptable bit error tolerance, it is recognized as frame, and it is identified as a lock or a flywheel frame depending on the mode. The number of Lock Errors is programmable from 0 to 15.
Check Frames	This is the number of consecutive frames that have a synchronization pattern within the Search Errors value and do not exceed the Window value that must occur in the data stream before the synchronization logic moves from check mode to lock mode. The value is frequently set to zero so that the synchronization logic moves immediately from search mode into lock mode. The number of Check Frames is programmable from 0 to 7.
Fly Frames	This is the number of consecutive flywheel frames that can occur before the synchronization logic goes back to search. The synchronization logic goes into flywheel mode when it is in lock mode but receives a frame with synchronization pattern bit errors that exceed the Lock Errors value or a frame slip that exceeds the Windows value. The number of Fly Frames is programmable from 0 to 7.

When you click the Extended Options button in the Config AVTEC Serial Input window, the window shown in **Figure 4-9** appears:

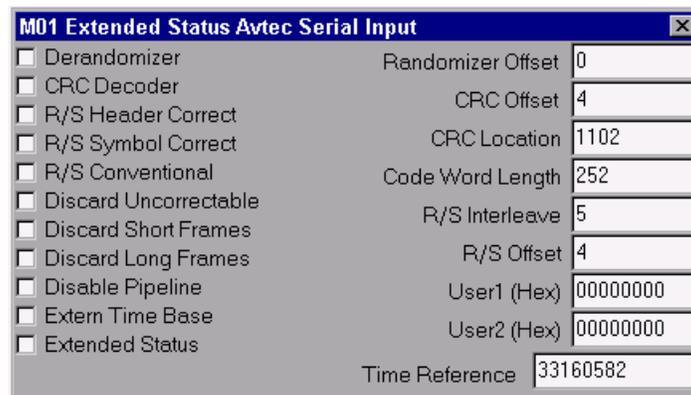


Figure 4-9: Config AVTEC Input Extended Options Window

Table 4-7 defines the fields in the AVTEC Serial Input Extended Options window. **The Serial Input Extended Options only apply to the MONARCH-E Frame Synchronizer with Reed-Solomon Decoder.**

Table 4-7: Config AVTEC Serial Input Extended Options Parameters

Parameter	Description
Derandomizer	Controls CCSDS derandomization. Checked is enabled.
Randomizer Offset	Defines in bytes the randomizer offset from the start of the frame. The Randomizer offset is programmable from 0 to Frame Length - 1. For CCSDS telemetry streams the value of 4 should be used to exclude the 32-bit attached sync marker.
CRC Decoder	Controls CCSDS CRC checking. Checked is enabled. Note that the MONARCH-E performs the CRC calculation and comparison prior to Reed-Solomon decoding. When CRC and Reed-Solomon decoding are used simultaneously, the CRC decoder may report CRC errors that are then corrected by the Reed-Solomon decoder. The CCSDS Virtual Channel Processor module can be used to perform CRC check in software after hardware Reed-Solomon decoding is performed by the MONARCH-E.
CRC Offset	Defines the CRC calculation start point offset. The two typical setups are 0 and 4. A 0 setup means CRC calculation begins as the start of the frame. A 4 setup means that a 32 bit synchronization pattern is excluded from the CRC calculation. The CRC Offset is programmable from 0 to CRC Location - 2.
CRC Location	Defines in bytes the location of the 16 bit CRC remainder value. For CCSDS Grade-3 service (no Reed-Solomon check symbols), the 16-bit CRC is the last 2 bytes in the frame. For CCSDS Grade-2 service (Reed-Solomon check symbols are present), the 16-bit CRC is the last 2 bytes in the VCDU or the 2 bytes immediately preceding the Reed-Solomon check symbols. The CRC Location is programmable from 0 to Frame Length - 2.
R/S Header Correct	Controls the correction of the VCDU header using the RS(10,6) decoder. Checked is enabled.
R/S Symbol Correct	Controls the correction of the VCDU using the RS(255,223) decoder. Checked is enabled.
R/S Conventional	Unchecked selects the dual-basis for the RS code as defined by the CCSDS. Checked selects the conventional basis.
Code Word Length	Defines the length of the RS(255,223) codeword. The MONARCH-E supports codeword lengths from 33 to 255.
R/S Interleave	Defines the RS code interleave depth. The MONARCH-E supports interleave depths from 1 to 8.

Parameter	Description
R/S Offset	Defines the start of the RS codeblock in the frame. To exclude a 32-bit sync pattern, this value should be set to 4.
User1 (Hex)	User-defined field that will appear in the appended quality annotation
User2 (Hex)	User-defined field that will appear in the appended quality annotation
Discard Uncorrectable	Controls the filtering of uncorrectable frames. Checked, the serial input will discard uncorrectable frames. Unchecked, the serial input will pass uncorrectable frames.
Discard Short Frames	Controls the filtering of short frames. Checked, the serial input will discard short frames. Unchecked, the serial input will pass short frames.
Discard Long Frames	Controls the filtering of long frames. Checked, the serial input will discard long frames. Unchecked, the serial input will pass long frames.
Disable Pipeline	Enables or disables the Reed-Solomon decoder pipeline. Checked is disabled. The Reed-Solomon decoder pipeline can be disabled for low data rate streams (less than 32 Kbps) to reduce the latency introduced by the decoder. This field should not be checked for processing high data rate streams.
Extern Time Base	This parameter determines the time base for the time stamp counter. When enabled, the external time source from either the AUX_TTL or AUX_422 input (on the DB37 connector) drives the onboard timer/counter. The 'Source' field in the Serial Input Config (Loopback, Ext TTL, or Ext 422) also controls the external time source. For the Loopback and Ext TTL settings, the AUX_TTL signal is selected for the external time base. For the Ext 422 setting, the AUX_422 input is selected for the external time base. When Extern Time Base is not checked, the PCI clock/4 drives the onboard timer/counter.
Time Reference	This parameter defines the frequency (in Hz) of the clock that drives the onboard timer/counter. When 'Extern Time Base' is not checked, this is the frequency of the system PCI clock. When Extern Time Base is enabled, this value should be set to the frequency of the external time base which is typically 5 MHz or 10 MHz
Extended Status	Controls the MONARCH-E appended status feature. If checked, the MONARCH-E will append quality annotation and time stamp to every frame. The format of the MONARCH-E appended status is outlined below.

The MONARCH-E supports CCSDS Reed-Solomon codes with virtual fill and interleave depths up to 8. The following equation shows the relationship between frame length (Frame Len), Code Word Length, interleave depth (R/S Interleave), and unencoded offset (R/S Offset):

$$\text{Frame Len} = \text{R/S Offset} + (\text{Code Word Length} * \text{R/S Interleave})$$

The amount of virtual fill in a frame is given by:

$$\text{Virtual Fill} = (255 - \text{Code Word Length}) * \text{R/S Interleave}$$

The MONARCH-E supports CRC decoding as defined in the CCSDS recommendations. The 16-bit CRC is the last two bytes in the VCDU. When Reed-Solomon coding is not used, the CRC location is Frame Length – 2. When Reed-Solomon coding is used, the CRC location is given by:

$$\text{CRC Location} = \text{Frame Length} - (32 * \text{R/S Interleave}) - 2.$$

A typical CCSDS transfer frame uses a Code Word Length of 252 bytes with R/S Interleave of 5. For this case, the equations above give:

$$\text{Frame Len} = 4 + (252 * 5) = 1264 \text{ bytes}$$

$$\text{CRC Location} = 1264 - (32 * 5) - 2 = 1102.$$

When you click the Status button on the AVTEC Serial Input module window, a Status AVTEC Serial Input window appears, as shown in **Figure 4-10**.

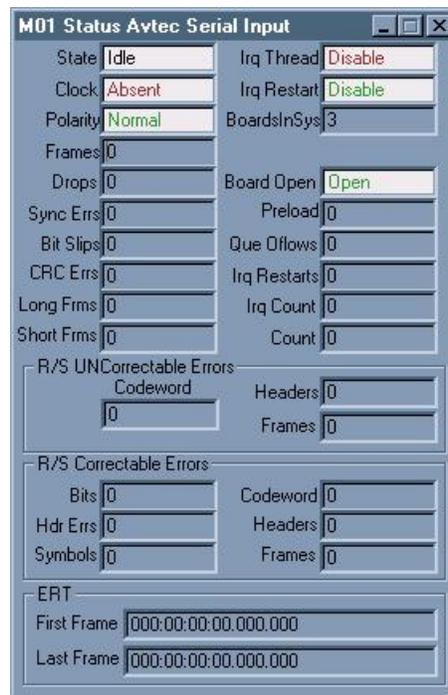


Figure 4-10: Status AVTEC Serial Input Window

Table 4-8 through **Table 4-11** define the fields in the Status AVTEC Serial Input Window.

Table 4-8: Status AVTEC Serial Input Parameters

Parameter	Description
State	Receiver state: Idle, Search, Check Lock, Flywheel

Parameter	Description
Clock	Absent or Present. Indicates whether the boards clock signal input is active (togglng).
Polarity	Normal or Inverted. Indicates whether the Auto-Polarity Correction logic is active. The Polarity is Normal if the Auto-Polarity Correction logic is not active. The Polarity is Inverted if the Auto-Polarity Correction logic is active.
Frames	Count of frames received in the Lock or Flywheel States. Frames received in the Check State are not included in this count. To be recognized, a frame must meet all of the error boundaries defined by the Serial Input configuration. This means that a frame must not exceed the maximum allowable synchronization pattern bit errors and maximum allowable number of bit slips for the current synchronization mode in order to be counted as an input frame.
Drops	Count of dropouts (Lock to Search transitions).
Sync Errs	This status reports the total number of bit errors in the synchronization patterns of the frames recognized by the serial input. It is important to note that only frames with synchronization pattern bit errors within the tolerance are reported: frames that exceed the error tolerance were never recognized as frames and cannot be counted. (This field is not currently implemented).
Bit Slips	This status reports how many of the frames recognized were slipped either long or short. It is important to note that only frames with slip bit errors within the tolerance are reported: frames that exceed the error tolerance were never recognized as frames and cannot be counted.
CRC Errs	Count of frames received in the Lock or Flywheel state that have CRC errors. (MONARCH or MONARCH-E only)
Long Frms	Count of long frames as seen by the Reed-Solomon Decoder. (MONARCH-E only)
Short Frms	Count of short frames as seen by the Reed-Solomon Decoder. (MONARCH-E only)
Irq Thread	Indicates if this Serial Input Module is currently Enabled or Disabled.
Irq Restart	Indicates if the Serial Input Module is automatically restarting (Enable) because of an error. Should be Disable if no errors occur.
BoardsInSys	Count of AT-HSIO2 or MONARCH boards in system as recorded in the Windows NT registry
Board Open	Displays if the board is currently Opened (Enabled) or

Parameter	Description
	Closed (Disabled).
Preload	Count of bytes that are preloaded into the AV_FSTS driver by the Serial Input Module.
Que Oflows	Count of overflows within the Serial Input thread. Should be zero for normal operation.
Irq Restarts	Count of Serial Input thread Restarts after overrun errors. A nonzero value indicates an error condition.
Irq Count	Count of successful attempts to read a buffer of data from the AV_FSTS driver.
Count	Count of frames read from the AV_FSTS driver.

The status values in the R/S UNCorrectable Errors sub-menu are reported by the MONARCH-E Reed-Solomon Decoder Chip.

Table 4-9: Status AVTEC Serial Input R/S UNcorrectable Errors Parameters

Parameter	Description
Codeword	Count of uncorrectable codewords (>16 byte errors)
Headers	Count of uncorrectable headers (> 2 nibble errors)
Frames	Count of frames with uncorrectable codewords

The status values in the R/S Correctable Errors sub-menu are reported by the MONARCH-E Reed-Solomon Decoder Chip.

Table 4-10: Status AVTEC Serial Input R/S Correctable Errors Parameters

Parameter	Description
Bits	Count of corrected bit errors
Hdr Errs	Count of corrected header errors - RS(10,6) code
Symbols	Count of corrected symbol errors
Codeword	Count of corrected code words
Headers	Count of corrected headers
Frames	Count of corrected frames

Table 4-11: Status AVTEC Serial Input Earth Receive Time (ERT)

Parameter	Description
First Frame	Time first frame received after Lock (Day of year: Hours: Minutes: Seconds : Milliseconds : Microseconds)
Last Frame	Time most recent frame received (Day of year: Hours: Minutes: Seconds : Milliseconds : Microseconds)

The MONARCH and MONARCH-E can append status information to every frame. The appended status is 48 bytes long and includes Time Tag, Frame Synchronizer Status, and RS Decoder Quality Annotation (MONARCH-E). The appended status is available to other modules to create headers or trailers with specific data quality and time information. The appended status can also be recorded to disk along with the frame data using the File Recorder Module.

The appended status information is outlined in the tables below.

Table 4-12: MONARCH and MONARCH-E 48-byte Appended Status Format

Parameter	Size	Description
Application_Flags	4 bytes	Currently not used.
Time Stamp	8 bytes	MONARCH time stamp tick counter which is extended to 64-bits by the Serial Input Module. This value is the number of ticks since the MONARCH timer was reset by the Serial Input Module (on enable). The Serial Input Module stores the absolute time when the board is enabled. The Serial Input Module computes the absolute time stamp for each frame by adding the stored start time to the relative offset appended to the frame. This absolute time stamp is passed to other modules in PB-4 format.
Hardware Flags	4 bytes	Bit mapped hardware status generated by the MONARCH Frame Synchronizer. Details are provided in the table below.
RS Annotation	32 bytes	Quality Annotation generated by the Reed-Solomon Error Correction chip. Details are provided in the table below. (All zeroes for MONARCH)

The bit definitions for the 32-bit Hardware Flags field in the Appended Status is outlined in **Table 4-13**.

Table 4-13: Appended Status Hardware Flags

Bit(s)	Name	Description
31:30	SICQ<1:0>	This is a 2 bit field that indicates the state of the serial input channel control logic after the frame was received. Values are defined as follows: SICQ<1:0> Input Channel State

Bit(s)	Name	Description
		00 search (a dropout occurred) 10 flywheel 11 lock
29	APC_FLG	This is the Autopolarity Correct Flag. When it is high (1), it indicates that the serial input control logic locked on an inverted synchronization pattern and that the Autopolarity Correct logic is active.
28	CRC_ERR_FLG	This is the CRC Error Flag. It is set high (1), if a CRC error occurs on that frame. A CRC error indicates that the Serial Input calculated CRC did not match the CRC value in the frame field. (This flag is not currently implemented)
27	SLIP_FLG	This is the Slip Flag. When it is high (1), it indicates that the previous frame is either too long or too short meaning that the Sync Pattern for the following frame arrived too early or too late.
26:24	SDIS<2:0>	This is the 2 bit Slip Distance. This field indicates the magnitude and direction of the current frame's slip. To calculate the slip, use the allowable slip window (Serial Input Config Synchronizer Window) The slip value is given by SDIS <2:0> - allowable slip window . For example, if the allowable slip window is 1 and SDIS <2:0> is 1, then no slip occurred. If the allowable slip window is 1 and SDIS <2:0> = 0, then the sync pattern occurred one bit later than expected.
23	TMSV	This is the Time Stamp Overflow. If set high (1), this bit indicates that the Time Stamp Counter has overflowed. This bit is not used because the Serial Input Module extends the Time Stamp Counter to 64-bits.
22	UNLOCK_FLG	This is the Unlock Flag. If this bit is high (1), it indicates that the serial input channel logic has entered the Search state from a Lock or a Flywheel state. It indicates loss of frame synchronization. This flag is not currently implemented. The same condition is indicated by a 5/CQ <1:0> value of 00.
21	LOCK_FLG	This is the Lock Flag. If high (1), it indicates that the Serial Input Logic has entered the Lock state from either Search or Check state. Will only be set for the first frame received after entering the Lock state.
20:16	PATERR	This is the 4 bit Synchronization Pattern Error. This binary number represents the number of synchronization pattern errors in this frame's synchronization pattern. This field is not currently implemented.
15:0	X	Not used.

The definition of the fields in the 32-byte Reed-Solomon Decoder Quality Annotation is given in **Table 4-14** below. These fields are only valid for the MONARCH-E board.

Table 4-14: Reed-Solomon Decoder Chip Annotation Format

Parameter	Size	Description
User Annotation [1]	4 bytes	32-bit user defined annotation from Serial Input Config Extended Options (User 1)
User Annotation [2]	4 bytes	32-bit user defined annotation from Serial Input Config Extended Options (User 2)
Frame Counter	4 bytes	32-bit count which is incremented for each frame passed through the RS decoder chip. Starting value is zero on enable.
Routing Data	1 byte	Not Used.
Terse Quality	1 byte	Contains high level information about the current frame. Defined in Table 4-15 below.
Errors Corrected	1 byte	Number of RS (255,223) errors corrected
Header Errors Corrected	1 byte	Number of RS (10,6) header errors corrected
Errored Codewords	1 byte	Number of codewords with errors in the current frame. An uncorrectable codeword is also errored, and this is counted in both fields.
Uncorrectable Codewords	1 bytes	Number of uncorrectable codewords in the current frame.
Error Bitfield	2 bytes	16 bit field that indicates which codewords were errored. Bit 0 refers to the first codeword, bit 1 refers to the second, etc. As with the previous quality annotation fields, if the uncorrectable bit is set for a certain codeword, the same bit is set in the errored bit field.
Uncorrectable Bitfield	2 bytes	16 bit field that indicates which codewords were uncorrectable. Bit 0 refers to the first codeword, bit 1 refers to the second, etc.
Bit Errors	2 bytes	Total number of bit errors found and corrected by the RS (255,223) decoder. The block decoder can correct up to 16 symbol errors per codeword. Because each symbol is a byte, the decoder could possibly correct 128 bits in a single codeword. This field reports the total number of bits corrected by the block decoder.
Errors Per Codeword	8 bytes	16 nibble fields; each nibble indicates the total number of errors per codeword. The first nibble refers to codeword 16, and the last nibble refers to codeword 1.

The Reed-Solomon Decoder Terse Quality field is defined in **Table 4-15**.

Table 4-15: Reed-Solomon Decoder Chip Terse Quality Definition

Bit #	Description
7	Long frame
6	Short frame
5	Unrouteable frame
4	Uncorrectable frame
3	RS (255,223) uncorrectable frame
2	RS (10,6) uncorrectable header
1	RS (255,223) found errors in frame
0	RS (10, 6) found errors in header

Section 3: AVTEC Serial Output Module

The Serial Output module supports the Telemetry Simulator portion of the AT-HSIO2 and the Reed-Solomon Encoder/ Telemetry Simulator portion of the MONARCH-E. The Serial Output module accepts buffers of data from other modules and writes the data to the I/O board for serial output. The Serial Output module can post an event each time it completes writing a buffer to the board.

This section provides an overview of the serial output control and status functions. Please refer to the AT-HSIO2 or MONARCH-E Functional Descriptions (References #1 and 4) for more detailed information on a particular parameter.

The Serial Output Module communicates with the I/O board using the AV_FSTS Frame Synchronizer/Telemetry Simulator driver for Windows NT. The AV_FSTS driver refers to the boards by channel number (Channel 0, 1, etc.). The mapping between AV_FSTS channel number and physical I/O board is defined in the Windows NT registry using the FSTS_CONFIG.EXE utility.

Figure 4-11 shows the AVTEC Serial Output window that appears after loading an instance of the Serial Output Module.



Figure 4-11: AVTEC Serial Output Module Window

When you click the configure button, a Config AVTEC Serial Output window will appear, as shown in **Figure 4-12**. The configuration parameters shown in **Figure 4-12** apply to both the AT-HSIO2 and the MONARCH-E.

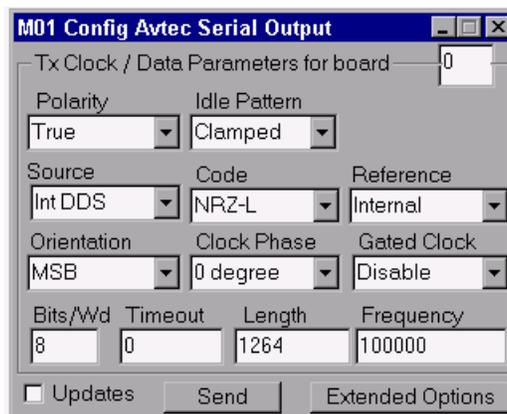


Figure 4-12: Config AVTEC Serial Output Window

Table 4-16 defines the fields in the Config AVTEC Serial Output window.

The entry after 'Tx Clock/Data Board Parameters for board' defines which AV_FSTS channel, and thus which I/O board, is assigned to this module. In **Figure 4-12**, this value is 0, which means that no board is assigned to the module. **The board number should be set to the desired AV_FSTS channel number plus one. For example, to assign the AV_FSTS channel 0 transmitter to the serial output module, set the board number to 1.**

- Tx Clock/Data Parameters for Board X sub-menu

Table 4-16: Config AVTEC Serial Output Tx Clock/Data Parameters for Board X Parameters

Parameter	Description
Polarity	Data Polarity: True or Inverted
Idle Pattern	Specifies idle transmission, all zeros (Clamped) or alternating 0/1 pattern (Toggle)
Source	Internal DDS or PLL: External TTL or RS-422
Code	NRZ-L, M, S, and BI0-L, M, S
Reference	Reference for the DDS: Internal, External TTL or RS-422
Orientation	MSB first, or LSB first
Clock Phase	0 degree, data transitions on the rising edge of the clock; or 180 degree, data transitions on the falling edge of the clock
Gated Clock	Enable (Shuts off clock at underrun) or Disable (Continuous clock)
Bits/Wd	Number of bits per word, 4 through 16, default 8
Timeout	msecs of no data until timeout (0 = no timeout)
Length	Frame Length in words (or bytes if bits per word = 8)
Frequency	Data Rate in Hz

When you click the Extended Options button in the Config AVTEC Serial Output window, the following window will appear, as shown in **Figure 4-13**. **The Serial Output Extended Options only apply to the MONARCH-E PCM Simulator with Reed-Solomon Encoder.**

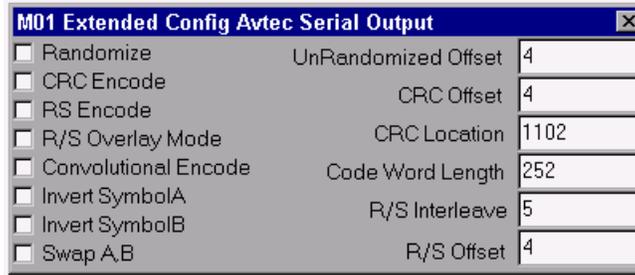


Figure 4-13: Config AVTEC Serial Output Extended Options Window

Table 4-17 defines the fields in the Config AVTEC Serial Output Extended Options window.

Table 4-17: Config AVTEC Serial Output Extended Options Parameters

Parameter	Description
Randomize	Controls the pseudo-randomizer. Checked is enabled.
CRC Encode	Controls the CRC encoder. Checked is enabled.
RS Encode	Controls the RS encoder. Checked is enabled.
R/S Overlay Mode	The MONARCH-E RS encoder can append the RS check symbols (32 per codeword) to the output frame (not checked) or overwrite part of the output frame with the RS check symbols (checked)
Convolutional Encode	Controls the MONARCH-E rate 1/2, constraint length 7 convolutional encoder. Checked enables the convolutional encoder. Note the transmit clock rate corresponds to the symbol rate (or 1/2 the bit rate) when the convolutional encoder is enabled.
Invert SymbolA	Controls the inversion of the convolutional encoder SymbolA (or G1 = 1111001) output. Checked inverts G1 before output.
Invert SymbolB	Controls the inversion of the convolutional encoder SymbolB (or G2 = 1011011) output. Checked inverts G2 before output.
Swap A,B	Controls the order the convolutional encoder symbols are transmitted. Unchecked, SymbolA (G1) is transmitted before SymbolB (G2). Checked, SymbolB (G2) is transmitted before SymbolA (G1).
UnRandomized Offset	Defines the randomizer offset from the start of the frame. To exclude a 32-bit sync pattern from randomization, this value should be set to 4.
CRC Offset	Defines the starting point for CRC calculations. To include the sync pattern, set this value to 0. To exclude a 32-bit sync pattern, set this value to 4.
CRC Location	Defines the location where the 16-bit CRC remainder should be written in the frame
Code Word Length	Defines the Reed-Solomon code word length from 33 to 255.
R/S Interleave	Defines the RS interleave depth from 1 to 8.
R/S Offset	Defines the starting point of the RS codeblock within the frame.

The MONARCH-E supports CCSDS Reed-Solomon codes with virtual fill and interleave depths up to 8. The following equation shows the relationship between frame length, Code Word Length, Code Word Length, interleave depth (R/S Interleave), and unencoded offset (R/S Offset):

$$\text{Frame Length} = \text{R/S Offset} + (\text{Code Word Length} * \text{R/S Interleave})$$

The amount of virtual fill in a frame is given by:

$$\text{Virtual Fill} = (255 - \text{Code Word Length}) * \text{R/S Interleave}$$

When you click the Status button in the AVTEC Serial Output window, a Status AVTEC Serial Output window appears, as shown in **Figure 4-14**.

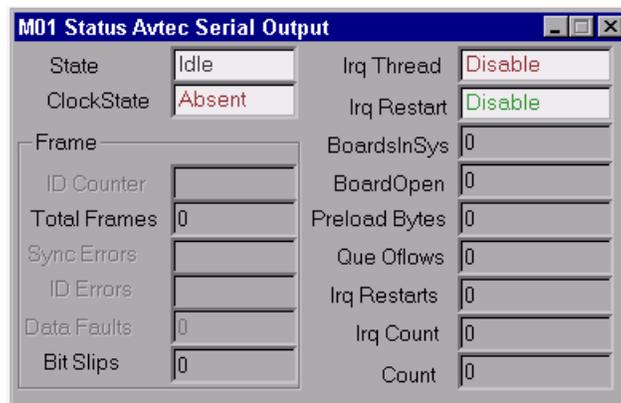


Figure 4-14: Status AVTEC Serial Output Window

Table 4-18 and **Table 4-19** define the fields in the Status AVTEC Serial Output window.

Table 4-18: Status AVTEC Serial Output Parameters

Parameter	Description
State	Idle or Transmit
ClockState	(future)
Irq Thread	Displays Serial Output Thread State
Irq Restart	Displays Automatic Restart State
BoardsInSys	Displays number of AT-HSIO2 or MONARCH-E boards in system as recorded in registry
BoardOpen	Specifies board in use
Preload Bytes	Number of bytes that are preloaded in the AV_FSTS driver
Que Oflows	Number of Queue Overflows in the Serial Output thread
Irq Restarts	Count of Serial Output thread restarts
Irq Count	Count of interrupts
Count	Count of frames written to the AV_FSTS driver

- Frame sub-menu

Table 4-19: Status AVTEC Serial Output Frame Parameters

Parameter	Description
ID Counter	(Future)
Total Frames	Count of frames transmitted
Sync Errors	(Future)
ID Errors	(Future)
Data Faults	(Future)
Bit Slips	Number of bit slips (not implemented)

Section 4: Com Port Transceiver Module

The Com Port Transceiver Module is used to send and receive buffers of data via the PC's Com Ports.

Figure 4-15 shows the Com Port Transceiver Module.



Figure 4-15: Com Port Transceiver Module Window

When you click the Configure button, a Config Com Port Transceiver Module window will appear, as shown in **Figure 4-16**.



Figure 4-16: Config ComPort Transceiver Window

Table 4-20 defines the fields in the Config Com Port Transceiver Module window.

Table 4-20: Config Com Port Transceiver Parameters

Parameter	Description
Port	Select Com Port 1, 2, 3, or 4
Baud Rate	Set data rate for Com Port (110 to 57,600 bps)
Handshake	Enable or Disable Handshaking for sending and receiving data
Phone Num	Future
FrameSync	Displays Frame Sync pattern (future)
Len	Displays Frame Sync length (future)
FrameSize	Specify Frame Size for incoming or outgoing data

When you click the Command button, a Command Com Port Transceiver Module window will appear, as shown in **Figure 4-17**.



Figure 4-17: Command Com Port Transceiver Window

Table 4-21 defines the fields in the Command Com Port Transceiver Module window.

Table 4-21: Command Com Port Transceiver Parameters

Parameter	Description
No Command Selected	No command selected
Dial	Future

When you click the Status button, a Status Com Port Transceiver Module window will appear, as shown in **Figure 4-18**.



Figure 4-18: Status Com Port Transceiver Window

Table 4-22 defines the fields in the Status Com Port Transceiver Module window.

Table 4-22: Status Com Port Transceiver Window Parameters

Parameter	Description
Tx Count	Displays count of frames transmitted
Rx Count	Displays count of frames received
Tx Que Oflows	Displays count of transmitter queue overflows

Section 5: CPU Timer Module

The CPU Timer generates events for other modules based on a user-specified interval. The CPU Timer has an event output but no data output.

The CPU Timer can be used to simulate end-of-frame interrupts when no serial I/O hardware is available. For example, the CPU Timer event could be connected to the Virtual Channel Simulator to generate frames at a particular rate for transmission over the network.

Figure 4-19 shows the CPU Timer module window.



Figure 4-19: CPU Timer Module Window

When you click the configure button, a Config CPU Timer window will appear, as shown in **Figure 4-20**.



Figure 4-20: Config CPU Timer Window

Table 4-23 defines the fields in the Config CPU Timer window.

Table 4-23: Config CPU Timer Parameters

Parameter	Description
Bit Rate	Specifies the simulated bit rate in bps
Frame Len	Specifies the simulated frame length
Sleep Time	Displays the period between events in milliseconds

The CPU Timer generates events based on the Bit Rate and Frame Length parameters. The relationship between the event interval, the simulated bit rate, and the frame length is given below:

$$\text{Interval} = \text{Frame Len} * 8 / \text{Bit Rate}$$

When you click the Status button on the CPU Timer module window, a Status CPU Timer window will appear, as shown in **Figure 4-21**.



Figure 4-21: Status CPU Timer Window

Table 4-24 defines the fields in the Status CPU Timer window.

Table 4-24: Status CPU Timer Parameters

Parameter	Description
Interrupt Events	Displays count of events generated
Sleep Time	Displays time between events in msec

Section 6: Dump to Line Printer Module

The Dump to Line Printer (DumpToLP) Module accepts blocks, frames, commands, network traffic, or any other inter-module data and outputs a formatted listing to a line printer (default) or to a file in octal, hexadecimal, or decimal format. User specified fields include number of bytes per line and number of lines per page. Current time and module from which the data came are listed with each frame/block/command. There is also an option to immediately print (character-at-a-time printer) and to spool full pages (laser printers).

Figure 4-22 shows the DumpToLP Module window.



Figure 4-22: Dump to Line Printer Module

Figure 4-23 shows the DumpToLP Module Config window.

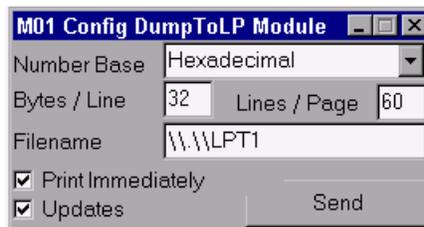


Figure 4-23: Config DumpToLP Window

Table 4-25 defines the Config DumpToLP Module window parameters.

Table 4-25: DumpToLP Config Parameters

Parameter	Description
Number Base	Format to output data: Hexadecimal, Decimal, or Octal
Byte / Line	Number of bytes to output per line (defines page width)
Lines / Page	Number of lines per page (defines page length)
Filename	Output location: Default=LPT1 (parallel port for printer); user can input any filename with path to output data

Figure 4-24 shows the DumpToLP Module Status window.

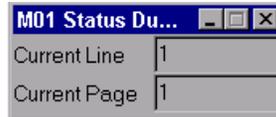


Figure 4-24: DumpToLP Status Window

Table 4-26 defines the DumpToLP Module Status window parameters.

Table 4-26: DumpToLP Status Parameters

Parameter	Description
Current Line	Current line number being output
Current Page	Current page number being output

Section 7: File Playback Module

The Playback Module supports reading binary data from hard disk or CD-ROM and transferring it to another module. The Playback Module supports five operating modes. In Fixed Mode, the Playback Module reads data until the end of file and then stops. In Loop Mode, the Playback Module reads until the end of file and then seeks back to the start of the file until disabled. In Segment Mode, the Playback module reads a segment of a file starting at a user-defined start location until a user-defined stop location. Sequence Mode is a combination of Segment and Loop Mode – the selected segment of a file is played in a continuous loop until disabled. Spooled mode allows the File Playback Module to be used in conjunction with the File Recorder to transmit frame data from a file while frames are still being written to the file. For this mode, the File Playback Module must be loaded after the File Recorder Module on the Desktop, with identical file names. As frames are requested from the File Playback, it will pull them from the binary file as the File Recorder appends new frames.

The File Playback module requires an event input. It reads a buffer from the file on each event. The File Playback has a single data output, the data read from the file.

Figure 4-25 shows the File Playback module window.

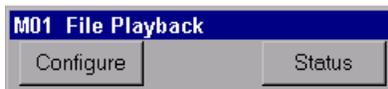


Figure 4-25: File Playback Module Window

When you click the configure button, a Config File Playback window will appear, as shown in **Figure 4-26**.

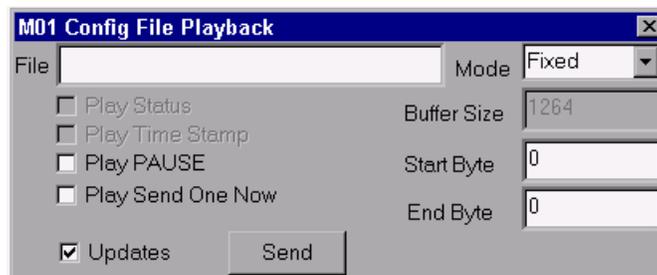


Figure 4-26: Config File Playback Window

Table 4-27 defines the fields in the Config File Playback window.

Table 4-27: Config File Playback Parameters

Parameter	Description
File	Specifies filename to replay
Play Status	future
Play Time Stamp	future
Play PAUSE	Stops playback, leaving file pointer in position
Play Send One Now	Sends one frame at a time (only works in PAUSE mode)
Mode	Choice of Fixed, Segment, Loop, Sequence Fixed – starts at beginning of file, and plays to the end Segment – Using StartByte and EndByte, the portion between is played Loop – Plays entire file over and over again, until manually terminated Sequence – Plays from StartByte to StopByte over and over again until manually terminated Spooled – Used in conjunction with File Recorder, frame data can be spooled from a file that is currently open by the File Recorder for writing
Buffer Size	Buffer size to send in bytes
Start Byte	Start byte in file (used when mode is Segment or Sequence)
End Byte	End byte in file (used when mode is Segment or Sequence)

When you click the Status button on the File Playback module window, a Status File Playback window will appear, as shown in **Figure 4-27**.



Figure 4-27: Status File Playback Window

Table 4-28 defines the fields in the Status File Playback window.

Table 4-28: Status File Playback Parameters

Parameter	Description
File	Displays file name being played
Play	Displays mode description for Fixed, Segment, Loop, Sequence, or Spooled modes
State	Displays “Waiting”, ”Running”, ”Paused”, or “Done”
Byte Position	Displays location in file
File Size	Displays byte count in file
Loop	Displays number of times file has been played when in Loop or Sequence mode

Section 8: File Recorder Module

The File Recorder Module supports logging binary data to hard disk or to a piped device. *Note: If buffer size value is set to a number that is modulo-4096, data cannot be logged to a piped device.* The File Recorder can log the output from any module. For example, the File Recorder can be used to log frame data from the Serial Input Module, data buffers from a Network Sockets Module, or packets from a Packet Processor Module. It can log raw frame data or frame data with time stamp and quality annotation. The File Recorder configuration parameters, status display, and file format are defined below.

The Recorder supports three record modes. In fixed mode, the recorder logs data until the end of file is reached and then closes the file. In dynamic mode, the recorder logs data until disabled or out of disk space. In circular mode, the recorder logs data until the end of file is reached and then loops back to the start of the file. The recorder can log raw frame data or frame data with time stamp and quality annotation.

Figure 4-28 shows the File Recorder module window.



Figure 4-28: File Recorder Module Window

When you click the configure button, a Config File Recorder window will appear, as shown in Figure 4-29.

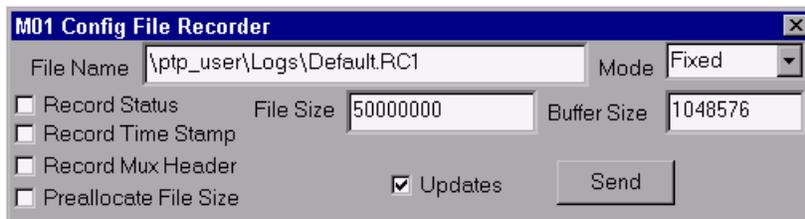


Figure 4-29: Config File Recorder Window

Table 4-29 defines the fields in the Config File Recorder window.

Table 4-29: Config File Recorder Parameters

Parameter	Description
File Name	Defines the log file name
Mode	Defines the recording mode: Fixed, Dynamic, Circular <u>Fixed</u> : To use this mode, the operator defines a FILE SIZE. Based on this file size, the File Recorder module saves data to file until the end of file is

	<p>reached, and then closes the file. If the data input is more than the file size can hold, the end of the data is lost.</p> <p><u>Dynamic:</u> In this mode, a file size is not predetermined. Therefore, the recorder logs data to file until either the end of the data flow input, until it is disabled, or until it runs out of disk space.</p> <p><u>Circular:</u> The Circular mode also uses the defined FILE SIZE. However, in this mode, data is logged to file until either the data flow ends or the recorder is disabled. If the end of the file is reached before the end of the data flow, then the recorder returns to the file start and begins to overwrite data. Therefore, data at the start of the data flow is lost if the file size is too small for the data flow.</p>
Record Status	Controls the logging of quality annotation and time stamp. If checked, the File Recorder writes the appended status from the Serial Input to the file before each buffer of data. The MUX Header is also written to the file before each buffer of data when Record Status is checked.
File Size	Size of file in bytes
Buffer Size	Defines the buffers size in bytes for file write operations. For best performance, the value should be a multiple of 4096. For best results , the buffer size should also reflect the data flow rate: the slower the rate, the smaller the buffer size should be.
Record Time Stamp	If checked, the File Recorder writes the buffer Time Stamp in PB-4 format to the file before the buffer of data. The MUX Header is also written to the file before each buffer of data when Record Time Stamp is checked.
Record Mux Header	If checked, the File Recorder writes the Multiplex (MUX) header to the file before each buffer of data. The MUX header defines which module sent the buffer to the File Recorder. It also defines which other headers (e.g. PB-4 or Appended Status) are present. Useful when logging multiple streams to a single file.
Preallocate File Size	This is an enable/disable field. If enabled, when the desktop is enabled the first thing the enable does is to reserve disk space for the size of the file specified in FILE SIZE field.

When you click the Command button on the File Recorder module window, a Command File Recorder window will appear, as shown in **Figure 4-30**.



Figure 4-30: Command File Recorder Window

Table 4-30 defines the field in the Command File Recorder window.

Table 4-30: Command File Recorder Parameter

Parameter	Description
Close File	Instructs recording to close. If recording mode is fixed, the file is truncated to the current size
Home File	Instructs the Recorder to seek the start of the file

When you click the Status button on the File Recorder module window, a Status File Recorder window will appear, as shown in **Figure 4-31**.

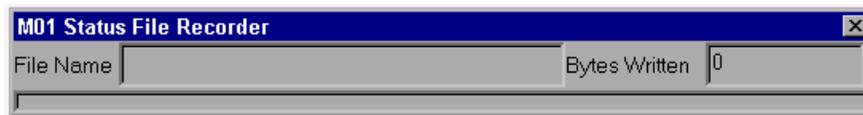


Figure 4-31: Status File Recorder Window

Table 4-31 defines the fields in the Status File Recorder window.

Table 4-31: Status File Recorder Parameters

Parameter	Description
File Name	Displays filename in use
Bytes Written	Displays bytes written into the file
Progress Indicator	Displays a visual cue of the file recording rate and status

File Recorder File Format

If no headers are selected, then the File Recorder logs the buffer of data to the file as it is received. If logging frame data from the Serial Input, note that the buffer size for each frame depends on the frame length and the word size. PCM Word sizes greater than 8 will be right justified in 2-byte words. The binary data is written in Intel little endian format (Low Byte, High Byte). MSB precedes the LSB for each byte. For use with Sun, HP, or SGI workstations, binary data must be byte re-ordered in software prior to processing.

If Record Status, Record Time Stamp, or Record Mux Header are selected then the File Recorder prepends a header to the input buffer and then writes it to the file as shown in Figures 1,2,3,4.

MUX (8 bytes)	Data Field (depends on Frame Length and Word Size)
-------------------------	--

Figure 4-32: File Recorder Format with Record MUX Header enabled.

MUX (8 bytes)	PB-4 (6 bytes)	Data Field (depends on Frame Length and Word Size)
-------------------------	--------------------------	--

Figure 4-33: File Recorder Format with Record Time Stamp enabled
(MUX header is automatically inserted).

MUX (8 bytes)	Appended Status (48 bytes)	Data Field (depends on Frame Length and Word Size)
-------------------------	--------------------------------------	--

Figure 4-34: File Recorder Format with Record Status enabled
(MUX header is automatically inserted).

MUX (8 bytes)	PB-4 (6 bytes)	Appended Status (48 bytes)	Data Field (depends on Frame Length and Word Size)
-------------------------	--------------------------	--------------------------------------	--

Figure 4-35: File Recorder Format with Record Time Stamp and Record Status enabled
(MUX header is automatically inserted).

The MUX Header structure is given in Table 1 below.

Table 4-32: PTP MUX Header Format

Item	Field Name	Format & Size	Value
1	Header Synchronization	Unsigned integer (4 bytes)	30030330 hex
2	Source Module	Unsigned integer (1 byte)	Equal to the source module number minus 1. For example, if Module 3 sends a buffer to the file recorder, the Source Module field is equal to 2.
3	Header Types	Unsigned integer (1 byte)	Defines the other header types that follow the MUX Header. 0 = No additional headers 1 = PB-4 Time Code 2 = Serial Input Appended Status 3 = Both PB-4 and Appended Status
4	Next Header Offset	Unsigned integer	Offset to the next MUX header in the file relative

		(2 bytes)	to the end of the current MUX header. Note that this is a Little Endian representation, the least significant byte precedes the most significant byte in the file. For example if the frame length 128 then the Next Header Offset appears in the file as 80 00 hex.
	Total Length	8 bytes	

The PB-4 Time Code format is given in Table 2 below.

Table 4-33: NASA PB-4 Time Code Format

Item	Field Name	Format & Size	Value
1	Days of Year	Unsigned integer (11 bits)	Range 1 to 365
2	Milliseconds of Day	Unsigned integer (27 bits)	Range = 0 to 86399999
3	Microseconds of a Millisecond	Unsigned integer (10 bits)	Range = 0 to 999
	Total Length	6 bytes	

The Appended Status Format is defined in Avtec Serial Input Module Section.

Section 9: File Spooler Module

The File Spooler Module combines capabilities from both the File Recorder and File Playback Modules. The Spooler takes data from one module (i.e. Avtec Serial Input), spools the data into a file on the local hard drive or a piped device, and plays back the data to another module (i.e. Network Sockets) at a different (faster or slower) rate. Similar to the Playback Module, the Spooler Module requires an event on output to drive the playback. Currently, the output frame length of the File Spooler Module must be an integer multiple of the incoming frame size (i.e. 1x, 2x, etc.). The next release of this module will allow dynamic frame lengths on the input and output.

Multiple File Spooler Modules can be chained to create a prioritized playback. To do so, a chain of events must be setup. First, multiple File Spoolers are loaded, the number of which is determined by the number of channels required for playback. The highest priority File Spooler channel to playback (priority 1) requires an event connection from an output module (i.e. Avtec Serial Output or Network Sockets). The priority 2 Spooler then requires an event connection from the priority 1 Spooler, and so on from Spooler to Spooler. This chaining of events determines the priority levels of playback.

Note: If using a Network Sockets Module (Sockets or Sockets Version 2) as the output, only the Sockets Module (not the Sockets Version 2) can be used to send events to this module. When using the Sockets Module, if the network goes down, the File Spooler will continue spooling to the local disk, then dump the spooled data to the network when it returns.

Figure 4-36 shows the File Spooler Module.



Figure 4-36: File Spooler Module

Figure 4-37 shows the File Spooler Module's Command window. It is identical to the File Recorder Module's Command window.



Figure 4-37: File Spooler Command Window

Table 4-34 defines the parameters in the File Spooler Command window.

Table 4-34: File Spooler Command Window Parameters

Parameter	Description
Close File	Instructs recording to close.
Home File	Instructs the Spooler to seek the start of the file.

Figure 4-38 shows the File Spooler Configuration window.



Figure 4-38: File Spooler Configuration Window

Table 4-35 defines the parameters in the File Spooler Configuration window.

Table 4-35: File Spooler Configuration Parameters

Parameter	Description
File Name	Defines the log file name.
Mode	Dynamic only due to nature of module.
Record Status	Not available in Spooler.
Record Time Stamp	Not available in Spooler.
Record MUX Header	Not available in Spooler.
Preallocate File Size	Reserves file space for the size of the file specified.
File Size	Size of file in bytes.
Buffer Size	Defines the buffer size for file write operations (should be multiple of 4096 for best performance).

Figure 4-39 shows the File Spooler Status window.

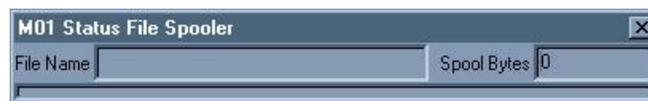


Figure 4-39: File Spooler Status Window

Table 4-36 defines the fields in the File Spooler Status window.

Table 4-36: File Spooler Status Fields

Parameter	Description
File Name	Displays filename in use.
Spool Bytes	Displays bytes written to file.
Progress Indicator	Displays a visual cue of the spooling rate and status.

Section 10: Network Sockets Module

The Network Sockets Module supports data transfer to/from a network interface. The Sockets Module supports connection oriented data transfer using TCP/IP sockets. The Sockets Module supports connectionless data transfer using UDP/IP sockets. The Sockets Module also support IP multicast so that data can be transferred to multiple hosts simultaneously without multiple transmissions on the network. Note: That IP multicast may not be supported by all routers.

Figure 4-40 shows the Sockets module window.



Figure 4-40: Sockets Module Window

When you click the configure button, a Config Sockets window appears, shown in **Figure 4-41**.

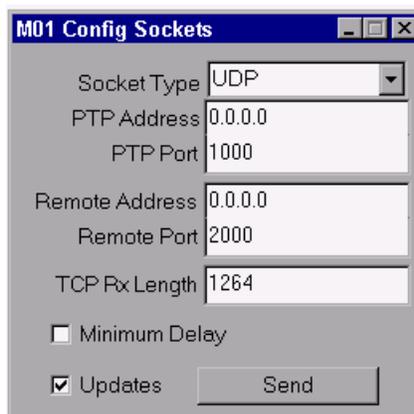


Figure 4-41: Config Sockets Window

Table 4-37 defines the fields in the Config Sockets window.

Table 4-37: Config Sockets Parameters

Parameter	Description
Socket Type	Choice of UDP, TCP Server, TCP Client, or Multicast
PTP Address	PTP IP address (0.0.0.0 specifies the default machine IP address)
PTP Port	Socket number (use for TCP Server or UDP Receiver)

Parameter	Description
Remote Address	Remote IP address (only used for TCP client or UDP transmit)
Remote Port	Remote Socket number (only used for TCP client or UDP transmit)
TCP Rx Length	TCP Receive Length in bytes
Minimum Delay	Sets buffer size to 512 kB

When you click the Status button on the Socket module window, a Status Sockets window will appear, as shown in **Figure 4-42**.

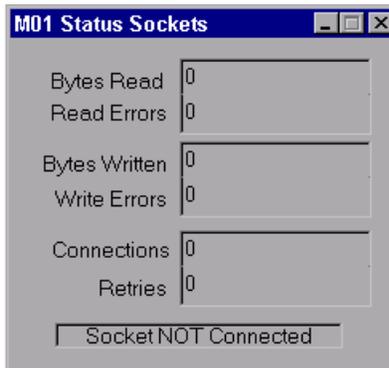


Figure 4-42: Status Sockets Window

Table 4-38 defines the fields in the Status Sockets window.

Table 4-38: Status Sockets Parameters

Parameter	Description
Bytes Read	Count of bytes read
Read Errors	Count of read errors
Bytes Written	Count of bytes written
Write Errors	Count of write errors
Connections	Number of connections
Retries	Number of connection retries

Section 11: Network Sockets Ver 2 Module

The Network Sockets Version 2 Module, similar to the Network Sockets Module, supports data transfer to/from a network interface. As opposed to the Sockets Module, the Sockets Version 2 Module only supports these transfers in a TCP Server mode. However, the TCP Server mode for the Sockets Version 2 supports multiple clients (the Sockets Module supports one TCP client). This module allows multiple clients to request an identical data stream from a single Server with the reliability of a TCP connection.

Figure 4-43 shows the Sockets Version 2 Module.



Figure 4-43: Sockets Version 2 Module

Figure 4-44 shows the Sockets Version 2 Config window.

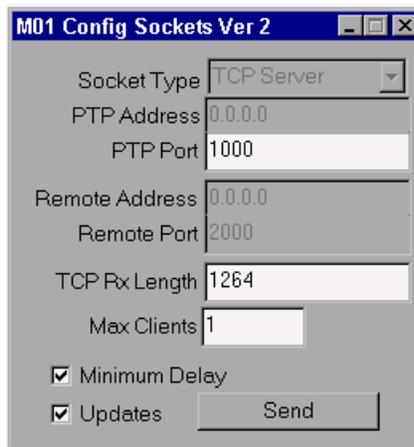


Figure 4-44: Sockets Version 2 Config Window

Table 4-39 defines the parameters in the Sockets Version 2 Config window.

Table 4-39: Sockets Version 2 Config Parameters

Parameter	Description
Socket Type	Set to TCP Server mode. Not user-changeable.
PTP Address	Defaults to IP address of local machine. Not user-changeable.
PTP Port	User selectable socket port to be used for TCP transfers.
Remote Address	Not used
Remote Port	Not used

Parameter	Description
TCP Rx Length	TCP Receive Length in bytes
Max Clients	Sets maximum TCP Clients to allow
Minimum Delay	Disables TCP nagle algorithm and sets socket priority to highest

Figure 4-45 shows the Sockets Version 2 Status window.

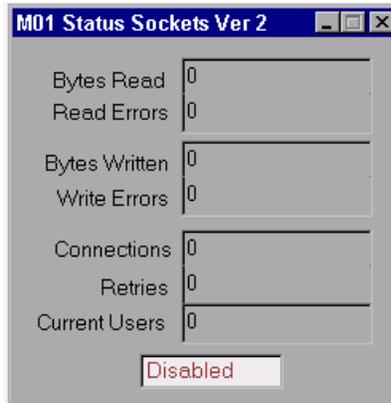


Figure 4-45: Sockets Version 2 Status Window

Table 4-40 defines the parameters in the Sockets Version 2 Status window.

Table 4-40: Sockets Version 2 Status Parameters

Parameter	Description
Bytes Read	Not Used
Read Errors	Not Used
Bytes Written	Count of bytes written
Write Errors	Count of write errors
Connections	Number of connections
Retries	Number of connection retries
Current Users	Number of clients connected to server

AUXILLIARY I/O MODULES

Section 12: Apogee PSK Modulator Module (ApogeePSK)

The Apogee PSK Module (ApogeePSK) supports the Apogee ISA-PSK BPSK Subcarrier Generator. This module has no data inputs or outputs and cannot be “connected” to any other modules. All connections to and from this device are external, hardwired connections. This board accepts serial command data at various rates from an external source in NRZ-L format. This data is used to coherently modulate a 16 kHz subcarrier

Figure 4-46 shows the ApogeePsk Module window.



Figure 4-46: Apogee PSK Modulator Module Window

When you click the Configure button, a Config ApogeePsk Module window will appear, as shown in **Figure 4-47**.

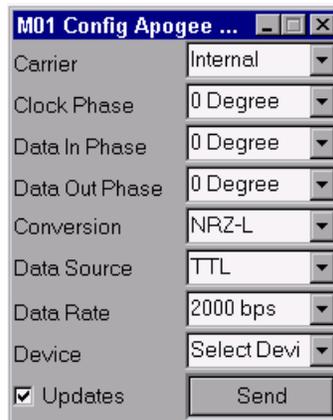


Figure 4-47: Config Apogee PSK Modulator Window

Table 4-41 defines the fields in the Config ApogeePsk Module window.

Table 4-41: Config Apogee PSK Modulator Parameters

Parameter	Description
Carrier	Selects subcarrier source: Internal, External TTL, or External 422
Clock Phase	0 or 180 ⁰
Data In Phase	Input Data Phase: 0 or 180 ⁰
Data Out Phase	Output Data Phase: 0 or 180 ⁰
Conversion	Code Conversion: NRZ-L, NRZ-M, NRZ-S
Data Source	Select Data Source: TTL or 422
Data Rate	Select Data Rate in bps: Off, 125, 250, 500, 1000, 2000
Device	Selects which Apogee PSK Modulator is being configured

Section 13: Apogee Time Code Generator Module (Time)

The Apogee Time Code Generator (Time) module supports the Apogee ISA-STG2 Synchronized Time Generator card. The Time module does not have any data inputs or outputs or event outputs and cannot be “connected” to another module.

Figure 4-48 shows the Time Module window.



Figure 4-48: Time Module Window

When you click the configure button, a Config Time Module window will appear, as shown in **Figure 4-49**.

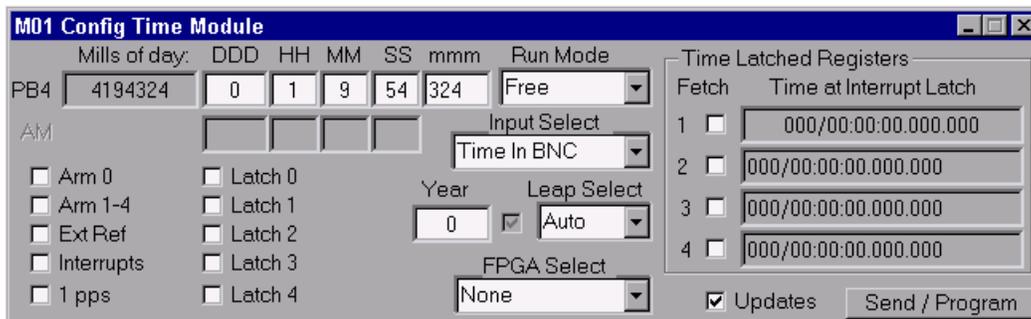


Figure 4-49: Config Time Window

The Config Time Module dialog controls the configuration of the ISA-STG2 Synchronized Time Generator.

Table 4-42 and **Table 4-43** define the fields in the Config Time Module window.

Table 4-42: Config Time Parameters

Parameter	Description
PB4 (Mills of day:)	Displays msec of day
DDD	Sets/Displays day of year
HH	Sets/Displays hour of day
MM	Sets/Displays minute of hour
SS	Sets/Displays seconds of minute
mmm	Sets/Displays msec
Run Mode	Free or Synchronized/Generate
AM	(future)
Input Select	Choice of Time Input from BNC, 1pps in from BNC, Freeze Input on BNC, External Reference, Auxiliary Input A, Auxiliary Input B, Auxiliary Input C, or Auxiliary Input D
Arm 0	Arms interrupt 0 (Future)
Arm 1-4	Arms interrupt 1 - 4 (Future)
Ext Ref	Set Ext Ref flag in R0 (Future)
Interrupts	Enable Interrupts (Future)
1 pps	Enables 1 pps signal (Future)
Latch 0-4	Specifies latching on interrupt 0 through 4 (Future)
Year	Specifies year (Future)
Leap Select	Specifies leap/not-leap year, or allows automatic detection based on 2 LSBs of year
FPGA Select	Selects 1 of 6 FPGA configurations to load: SGEN_36 - NASA-36 synchronized generator TCRD_36 - NASA-36 tape reader AMGEN36 - NASA-36 generator with AM output SGEN_B - IRIG-B synchronized generator TCRD_B - IRIG-B tape reader AMGENB - IRIG-B generator with AM output These files are provided with a PTP NT delivery

- Time Latched Registers sub-menu

Table 4-43: Config Time Module Time Latched Registers Parameters

Parameter	Description
Fetch	Performs a random fetch of the time in this mailbox; releases latch for next interrupt
Time at Interrupt Latch	Time at last latch in microseconds (μ S)

The Command button on the Time module window is not yet operable.

When you click the Status button on the Time module window, a Status Time Module window will appear, as shown in **Figure 4-50**.



Figure 4-50: Status Time Window

Table 4-44 defines the fields in the Status Time Module window.

Table 4-44: Status Time Parameters

Parameter	Description
MilliSeconds of Day	Displays milliseconds of day
PB4	Displays formatted PB4 time
AM	(Future)
Carrier	Ok or None
Code	Ok or Flywheel

Section 14: Aydin Bit Sync Module (AydinBS)

The Aydin Bit Sync Module (AydinBS) is used to configure and control the Aydin PC335 Bit Synchronizer board. The AydinBS module does not have any data inputs or outputs or event outputs and cannot be “connected” to another module

Figure 4-51 shows the Aydin Bit Sync Module window.



Figure 4-51: Aydin Bit Sync Module Window

When you click the configure button, a Config Aydin Bit Sync Module window will appear, as shown in Figure 4-52.

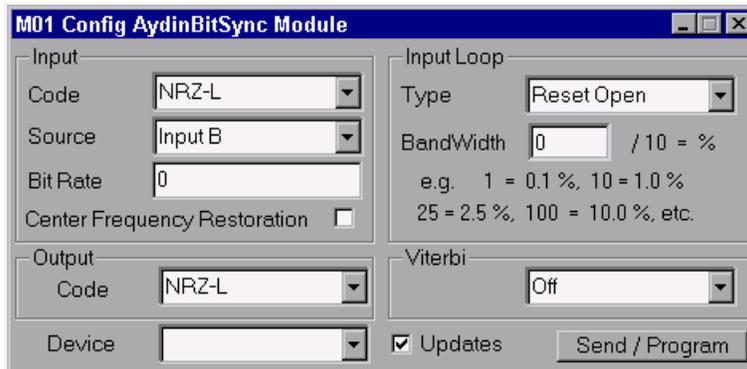


Figure 4-52: Config Aydin Bit Sync Window

Table 4-45 through Table 4-49 define the fields in the Config Aydin Bit Sync Module window.

- Input sub-menu

Table 4-45: Config Aydin Bit Sync Input Parameters

Parameter	Description
Code	Data code, NRZ-L, M, S; BIO-L, M, S; INRZ-L; RNRZ-L; IRNRZ-L; RNRZ-L(Rev); IRNRZ-L(Rev); IBIO-L; DM-M, S; and MDM-M, S
Source	Input A or B
Bit Rate	Expected Bit Rate
Center Frequency Restoration	Enables/Disables CFR

- Input Loop sub-menu

Table 4-46: Config Aydin Bit Sync Input Loop Parameters

Parameter	Description
Type	Specifies Open/Closed, Normal or Reset Loop
BandWidth (/10=%)	Specifies allowable bandwidth, in 0.1% resolution

- Output sub-menu

Table 4-47: Config Aydin Bit Sync Output Parameters

Parameter	Description
Code	Data code, NRZ-L, M, S; BI0-L, M, S; RNRZ-L

- Viterbi sub-menu

Table 4-48: Config Aydin Bit Sync Viterbi Parameters

Parameter	Description
Off/On	Choice of Viterbi Off, Encoder On, Decoder On, or Both On

The Command button on the Aydin Bit Sync Module window is not operable.

When you click the Status button on the Aydin Bit Sync Module window, a Status Aydin Bit Sync Module window will appear, as shown in **Figure 4-53**.



Figure 4-53: Status Aydin Bit Sync Window

Table 4-49 defines the fields in the Status Aydin Bit Sync window.

Table 4-49: Status Aydin Bit Sync Parameters

Parameter	Description
Deviation	Displays Loop Deviation as a percentage and a raw count
Signal	Displays Signal status as reported by the board
Status	Displays Lock status

Section 15: Aydin PSK Demodulator Module (Aydin PSKDemod)

The Aydin PSK Demodulator Module (Aydin PSKDemod) supports the Aydin PSK Demodulator card. The PSK Demodulator module does not have any data inputs or outputs or event outputs and cannot be “connected” to another module. The PSK Demodulator board performs subcarrier detection and synchronous demodulation of an input PSK modulation channel contaminated by wide band noise and embedded within a multiplex of other PSK and FM channel.

Figure 4-54 shows the PSKDemod Module window.



Figure 4-54: PSKDemod Module Window

When you click the configure button, a Config PSKDemod Module window will appear, as shown in **Figure 4-55**.

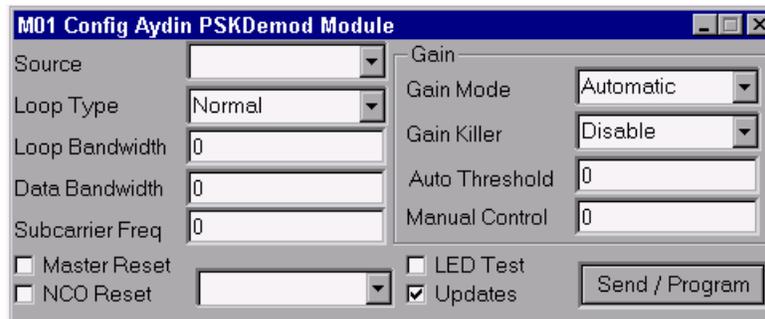


Figure 4-55: Config PSKDemod Window

Table 4-50 and **Table 4-51** define the fields in the Config PSKDemod Module window.

Table 4-50: Config PSKDemod Parameters

Parameter	Description
Source	Selects between source 1 or 2, also Low or Hi Z (Impedance)
Loop Type	Permits Normal or Open Loop
Loop Bandwidth	Specifies loop bandwidth in Hz
Data Bandwidth	Specifies data bandwidth in Hz
Subcarrier Freq	Specifies expected subcarrier frequency in Hz

Master Reset	Instructs board to perform a master reset and re-programming
NCO Reset	Instructs board to perform an NCO reset and re-sweep
LED Test	Turns on all LEDs on the PSKDemod board for a visual inspection

- Gain sub-menu

Table 4-51: Config PSKDemod Gain Parameters

Parameter	Description
Gain Mode	Choice of Automatic or Manual Gain control
Gain Killer	Enable or Disable
Auto Threshold	Automatic Gain Threshold; default = 80 (decimal)
Manual Control	Manual Gain Control Threshold Setting

When you click the Command button on the PSKDemod Module window, a Command PSKDemod Module window will appear, as shown in **Figure 4-56**.



Figure 4-56: Command PSKDemod Window

Table 4-52 defines the field in the Command PSKDemod Module window.

Table 4-52: Command PSKDemod Parameter

Parameter	Description
No Command Selected	No command selected
Master Reset	Initiates a Master Reset of the board
NCO Reset	Initiates a NCO Reset of the board
Master and NCO Reset	Initiates both a Master and NCO Reset of the board

When you click the Status button on the PSKDemod Module window, a Status PSKDemod Module Window will appear, as shown in **Figure 4-57**.



Figure 4-57: Status PSKDemod Window

Table 4-53 defines the fields in the Status PSKDemod Module window.

Table 4-53: Status PSKDemod Parameters

Parameter	Description
Signal Strength	Direct raw value readout from the board, with its validity indicator, followed by a calibrated Signal Strength in Volts peak-to-peak
Deviation	Direct raw value readout from the board, with its validity indicator, followed by a calibrated deviation read out in LoopBandWidths
Status	Displays Unknown, Subcarrier Sync, Signal LOS, or Overload

Section 16: GDP PSK Modulator Module (GDP PSKMod)

The PSK Modulator supports the GDP PSK004 PSK Modulator card. The PSK Modulator module does not have any data inputs or outputs or event outputs and cannot be “connected” to another module. The PSK Modulator board generates a modulated subcarrier according to user inputs,

Figure 4-58 shows the PSKMod Module window.



Figure 4-58: PSKMod Module Window

When you click the configure button, a Config PSKMod Module window will appear, as shown in **Figure 4-59**.

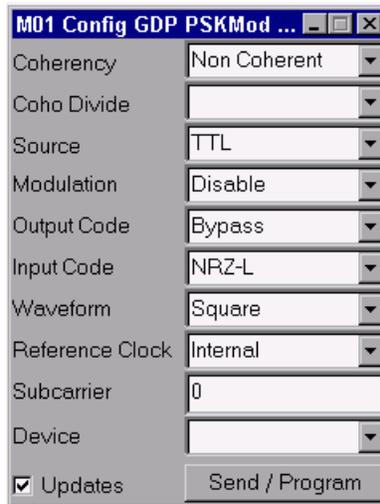


Figure 4-59: Config PSKMod Window

Table 4-54 defines the fields in the Config PSKMod Module window.

Table 4-54: Config PSKMod Parameters

Parameter	Description
Coherency	Choice of Coherent or Non-coherent operation
Coho Divide	Coherent Divisor; choice of 2, 4, 8, 16, 32, 64, 128
Source	Allows either TTL or 422 input

Parameter	Description
Modulation	Choice of Enable or Disable
Output Code	Choice of NRZ-L, M, S, or Bypass
Input Code	Choice of NRZ-L, M, S, or BI0-L, M, S, or Bypass
Waveform	Choice of Sinewave or Squarewave
Reference Clock	Choice of Internal, or External 5 or 10 MHz
Subcarrier	Subcarrier Frequency in Hz

Figure 4-60 shows the GDP PSK Modulator Module's Command window.



Figure 4-60: Command GDP PSK Modulator Window

Table 4-55 defines the GDP PSK Modulator's Command window parameters.

Table 4-55: Command GDP PSKMod Parameters

Parameter	Description
No Command Selected	No command selected
Disable Carrier	Disables the subcarrier by setting frequency to 0
Enable Carrier	Enable the subcarrier by setting frequency to programmed value

The Status button on the PSKMod Module window is not operable.

Section 17: Microdyne PCR-2000 Receiver/Demodulator Module (PCR2000)

The PCR2000 module supports the Microdyne PCR-2000 S-Band Receiver/Demodulator. This module has no data inputs or outputs and cannot be “connected” to any other modules. The Microdyne PCR-2000 is a two-board sandwich, which occupies one ISA slot. It has a dual conversion superheterodyne receiver, and operates in the 2200-2300 MHz (S-Band) range. The PTP does not validate this range, since boards with other ranges are available.

Figure 4-61 shows the PCR2000 Module window.



Figure 4-61: PCR2000 Module Window

Click on the *Configure* button on the PCR2000 Module window, and the Config PCR2000 Module window will open, as shown in Figure 4-62.

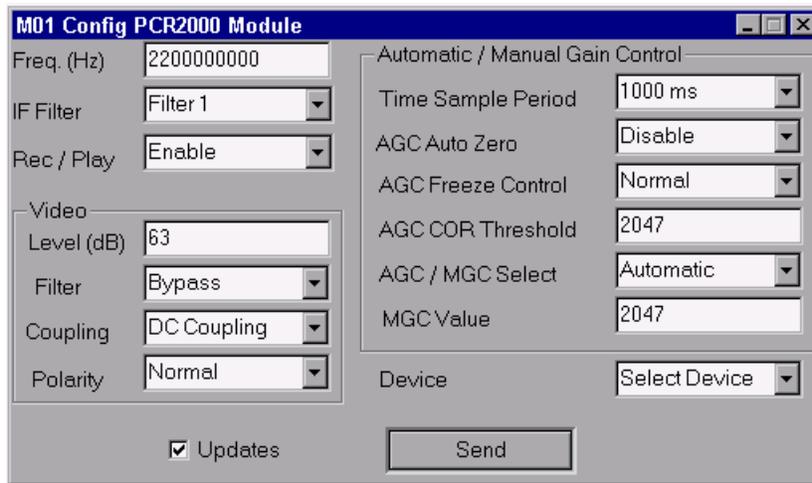


Figure 4-62: Config PCR2000 Module Window

Table 4-56 through Table 4-58 define the parameters in the PCR2000 Module’s Config window.

Table 4-56: Config PCR2000 Module Parameters

Parameter	Description
Freq. (Hz)	Input signal frequency; default is 220000000 (22 GHz)
IF Filter	Select IF filter to use, Filter 1, 2, 3, or 4
Rec / Play	Enables or Disable data to record line
Device	Selects which PCR2000 is being configured

Table 4-57: Config PCR2000 Module Video Parameters

Parameter	Description
Level (dB)	Input video signal level
Filter	Bypass/Enable (Filter 1, 2, or 3) video filter
Coupling	DC or AC coupling
Polarity	Selects signal polarity (Normal or Inverted)

Table 4-58: Config PCR2000 Module Gain Control Parameters

Parameter	Description
Time Sample Period	Selects Time Sample Period (0.1, 1.0, 10, 100, or 1000 ms)
AGC Freeze Control	Normal or Freeze
AGC COR Threshold	Input AGC COR Threshold value (0 to 4095)
AGC / MGC Select	Selects Automatic or Manual Gain Control
MGC Value	If MGC enabled, input value between 0 and 4095

Figure 4-63 shows the PCR2000 Module's Command window.

**Figure 4-63: Command PCR2000 Module Window**

Table 4-59 defines the parameters in the PCR2000 Module's Command window.

Table 4-59: Command PCR2000 Module Parameters

Parameter	Description
No Command Request	No command
Power On Test	Initiate power on test

Figure 4-64 shows the PCR2000 Module's Status window.

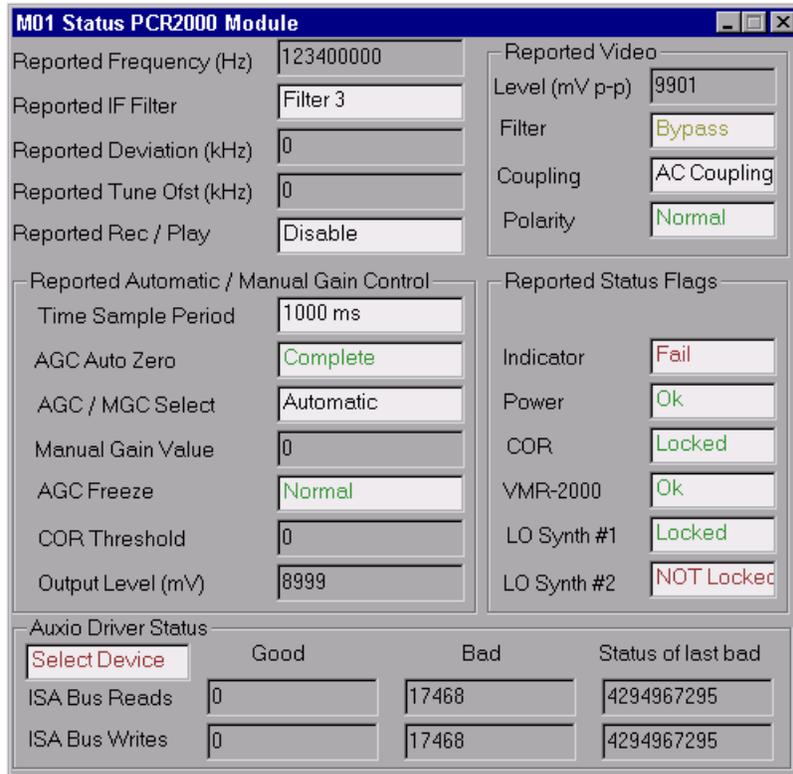


Figure 4-64: Status PCR2000 Module Window

Table 4-60 through Table 4-63 define the parameters in the PCR2000 Module's Status window.

Table 4-60: PCR2000 Module Status Parameters

Parameter	Description
Reported Frequency (Hz)	Displays input frequency
Reported IF Filter	Displays IF filter being used
Reported Deviation (kHz)	Displays deviation from input frequency
Reported Tune Ofst (kHz)	Displays tuneing offset
Reported Rec / Play	Displays if Play/Rec is enabled or disabled
Time Sample Period	Displays time sample period
AGC auto Zero	Displays status of AGC Auto Zero
AGC / MGC Select	Displays if automatic or manual gain control is being used
Manual Gain Value	Displays manual gain value
AGC Freeze	Displays status of AGC freeze
COR Threshold	Displays COR threshold value
Output Level (mV)	Displays output signal level

Table 4-61: PCR2000 Reported Video Status Parameters

Parameter	Description
Level (mV p-p)	Displays video level
Filter	Displays filter in use
Coupling	Displays AC or DC coupling in use
Polarity	Displays video polarity

Table 4-62: PCR2000 Reported Status Flags Parameters

Parameter	Description
Indicator	Displays general board status
Power	Displays Power status
COR	Displays COR (Command Output Ready) status (Locked or Not Locked)
VMR-2000	Displays VMR-2000 status (OK/Fail)
LO Synth #1	Displays status of LO Synth #1 (Locked or Not Locked)
LO Synth #2	Displays status of LO Synth #2 (Locked or Not Locked)

Table 4-63: PCR2000 AUXIO Driver Status Parameters

Parameter	Description
ISA Bus Reads	Displays number of good and bad ISA Bus Reads
ISA Bus Writes	Displays number of good and bad ISA Bus Writes

Section 18: National Instruments GPIB Interface Module (GPIB)

The GPIB Module supports the National Instruments General Purpose Interface Bus (GPIB) Board. GPIB is a standard method used to control instruments and takes measurements remotely. The GPIB module provides a generic interface to test the remote control capability of other instruments. This module can take user input and send it to a specific device, and await and accept a response or data from the instrument. This module is not tailored toward any specific instrument, rather it can address all instruments. Note: The Module Status for the GPIB Module is displayed in the Configuration window.

Figure 4-65 shows the GPIB Module.

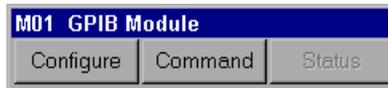


Figure 4-65: GPIB Module

Figure 4-66 shows the GPIB Module Config window.

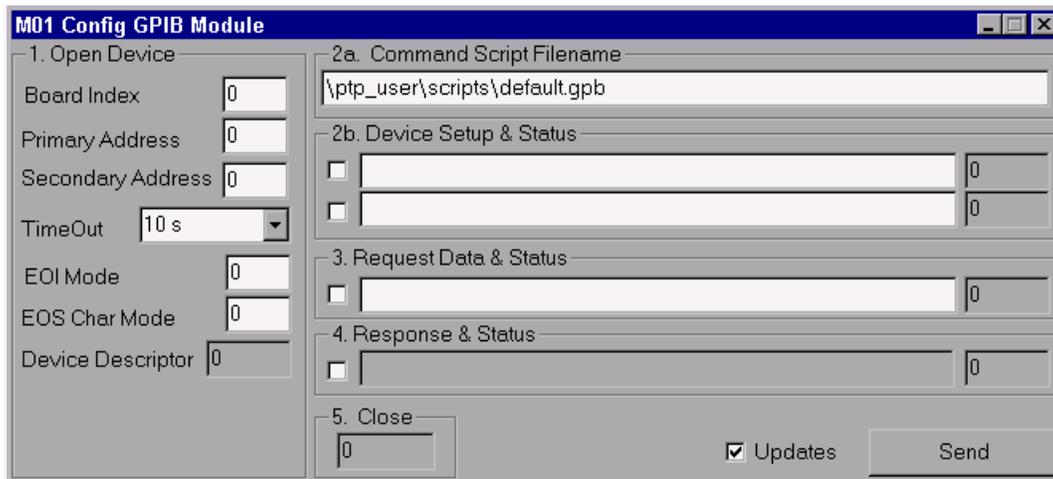


Figure 4-66: Config GPIB Module Window

Table 4-64 and **Table 4-65** define the parameters in the Config GPIB Module window.

Table 4-64: Config GPIB Module Open Device Parameters

Parameter	Description
Board Index	Board number of GPIB
Primary Address	Address of GPIB Device that is to be configured via the GPIB board selected by the Board Index parameter
Secondary Address	Secondary Address of GPIB Device (not used by all GPIB Devices)
TimeOut	Read/Write timeout for Device; User-selectable from 10 us to 1000 s
EOI Mode	End of Input Mode – defines how to terminate commands sent from GPIB Board to GPIB Device; Default terminator = 0
EOS Char Mode	End of String Character Mode – defines how to terminate strings sent from GPIB Board to GPIB Device; Default terminator = 0

Table 4-65: Config GPIB Command and Status Parameters

Parameter	Description
Commands Script Filename	Filename of text file that contains commands to be sent to GPIB device. The text file can be up to 30 lines long; each line is of the form: <i>COMMAND [name of command]=[string]</i> . These commands will be listed in the Command GPIB Module drop-down list.
Device Setup & Status	Commands to be sent to GPIB device. To send command, check checkbox and click Send in the Configuration window. Grayed edit box to right of command displays status.
Request Data & Status	Status command to be sent to GPIB device. To send request, check checkbox and click Send in the Configuration window. Grayed edit box to right of command displays status.
Response & Status	Displays responses to any commands and requests sent.
Close	Close status for GPIB device.

Figure 4-67 shows the GPIB Module's Command window.



Figure 4-67: Command GPIB Module Window

Table 4-66 defines the parameters in the Command GPIB Module window.

Table 4-66: Command GPIB Module Parameters

Parameter	Description
Command	Displays available commands to send to GPIB device. Commands are based on commands listed in file referenced by <i>Commands Script Filename</i> in the Config GPIB Module window.

Section 19: Odetics Time Module

The Odetics Time Module supports the Odetics GPSync-ISA Time Board. The Odetics Time Module does not have any data inputs or outputs or event outputs and cannot be “connected” to another module.

Figure 4-68 shows the Odetics Time Module



Figure 4-68: Odetics Time Module

When you click the configure button, a Config Odetics Time Module window will appear, as shown in Figure 4-69.

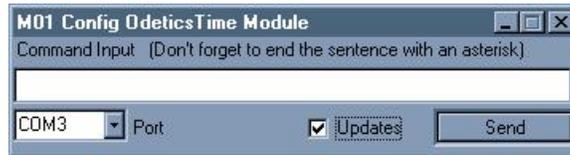


Figure 4-69: Config Odetics Time Window

Table 4-67 defines the parameters in the Config Odetics Time Module window.

Table 4-67: Config Odetics Time Module Parameters

Parameter	Description
Command Input	Edit box for inputting command strings to be sent to Odetics Time Board. All commands must end with an asterisk (*). Press Send to send the command to the board once the command has been input.
Port	Select which port the Odetics Time Board is using.

Click the Command button, and the Command window for the Odetics Time Module, shown in Figure 4-70, opens.



Figure 4-70: Command Odetics Time Board Window

Table 4-68 defines the commands available in the Odetics Time Command window.

Table 4-68: Odetics Time Commands

Parameter	Description
Recent Error	Retrieves the most recent error recorded by the board.
Warm Start	Causes the board to warm start. No positioning data is lost.
Cold Start	Causes the board to cold start. User must enter in latitude & longitude, or the board will take 6 hours to determine this information.
Clear All Tags	Clears all time tags.
Tag Buffer Wrap	Choose time buffers to wrap at end of last buffer.
Time Recov Dynamic	Tells board it is on a moving platform.
Time Recov PositionAvg	Board is in 6 hour latitude & longitude determination mode.
Time Recov Known	Latitude & longitude confirmed.
Set System Time	Sets PC time from the board.

Click the Status button to open the Odetics Time Status window, which is shown in **Figure 4-71**.

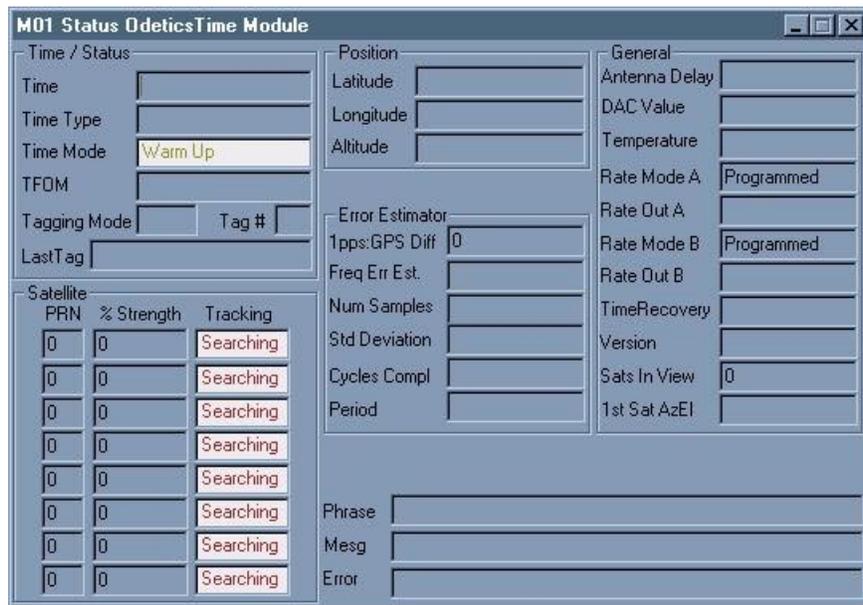


Figure 4-71: Odetics Time Status Window

Table 4-69 through **Table 4-74** define the parameters available in the Odetics Time Status window.

Table 4-69: Odetics Time Board Time/Status Parameters

Parameter	Description
Time	Current time as received from board.
Time Type	Local, UTC, or GPS (default = GPS).
Time Mode	Warmup, Time Lock, Coasting, Recovering, or Manual.
TFOM	Accuracy: $100\text{ns} \leq \text{ETE} \leq 1\text{ns}$, $1\text{ns} \leq \text{ETE} \leq 10\text{us}$, $10\text{us} \leq \text{ETE} \leq 100\text{us}$, $100\text{us} \leq \text{ETE} \leq 1\text{ms}$, $1\text{ms} \leq \text{ETE} \leq 10\text{ms}$, $10\text{ms} \leq \text{ETE}$.
Tagging Mode	Burst or Wrap
Tag #	1 – 100
Last Tag	(time) (to 100ns)

Table 4-70: Odetics Time Board Satellite Status Parameters

Parameter	Description
PRN	Satellite number
% Strength	0 – 100% Signal strength from satellite
Tracking	Searching, Acquiring, or Using this satellite

Table 4-71: Odetics Time Board Position Status Parameters

Parameter	Description
Latitude	Antenna Latitude
Longitude	Antenna Longitude
Altitude	Antenna Altitude

Table 4-72: Odetics Time Board Error Estimator Status Parameters

Parameter	Description
1pps:GPS Diff	1 pulse per second vs. GPS difference as reported from board
Freq Err Est	Most recent frequency error estimate (from board)
Num Samples	Current estimator cycle (from board)
Std Deviation	Most recently calculated standard deviation of estimator error (from board)
Cycles Compl	Number of estimator cycles completed (from board)
Period	Estimator period (from board)

Table 4-73: Odetics Time Board General Status Parameters

Parameter	Description
Antenna Delay	Antenna echo delay echoed as input by user (ANTD)
DAC Value	DAC value reading from board
Temperature	Temperature on board (degrees Celsius)
Rate Mode A	Number of pulses per second according to Mode table under ROUT directive in GPSync Board Manual 365-8005.
Rate Out A	Rate is microseconds
Rate Mode B	Number of pulses per second according to Mode table under ROUT directive in GPSync Board Manual 365-8005.
Rate Out B	Rate is microseconds
Time Recovery	Known, Dynamic, or Position Average (from board)
Version	Software version on board
Sats in View	Number of satellites in view of antenna (from board)
1 st Sat AxEl	Azimuth and elevation of first satellite (from board)

Table 4-74: Odetics Time Board Status Parameters

Parameter	Description
Phrase	Command input in ASCII
Mesg	Last message from board
Error	Last error from board

Section 20: Veda Bit Synchronizer Module (VedaBitSync)

The Veda Bit Synchronizer Module (VedaBitSync) supports both the Veda Bit Synchronizer board and the Veda Viterbi Decoder board. This module has no data inputs or outputs and cannot be “connected” to any other modules.

Figure 4-72 shows the VedaBitSync Module window.



Figure 4-72: VedaBitSync Module Window

When you click the configure button, a Config VedaBitSync Module window appears, as shown in Figure 4-73.

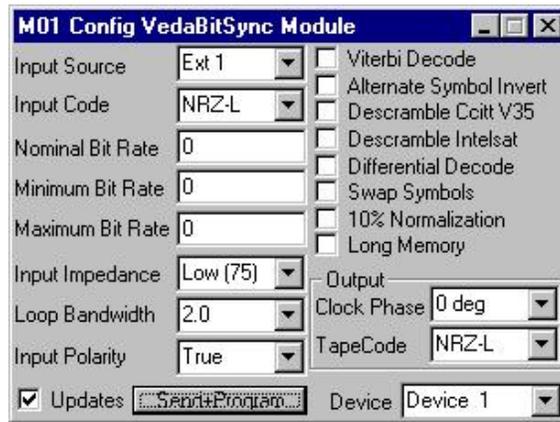


Figure 4-73: Config VedaBitSync Window

Table 4-75 through Table 4-76 define the fields in the Config VedaBitSync Module window.

Table 4-75: Config VedaBitSync Parameters

Parameter	Description
Input Source	Specifies input. Internal or 4 External inputs are available.
Input Code	NRZ-L, M, S; Bi0-L, M, S; DMMm DMS; RZ, RNRZ9-F, RNRZ11-F, RNRZ15-F, RNRZ9-R, RNRZ11-R, RNRZ15-R
Nominal Bit Rate	Specifies expected bit rate
Minimum Bit Rate	Calculated using Nominal bit rate and bandwidth
Maximum Bit Rate	Calculated using Nominal bit rate and bandwidth
Input Impedance	Choice of Low (50 ohms) or Hi (10K ohms)

Loop Bandwidth	Choice of Loop bandwidth; 0.1, 0.5, 1.0, 2.0
Input Polarity	Chooses input polarity, True or Inverted
Viterbi Decode	Enables/Disables Viterbi Decode, requires Convolutional Decoder board
Alternate Symbol Invert	Enables/Disables Alternate Symbol Inversion
Descramble CCITT V.35	Enables/Disables Descrambling
Descramble Intelsat	Enables/Disables Descrambling
Differential Decode	Enables/Disables Differential Decoding
Swap Symbols	Enables/Disables Symbol Swap

- Output sub-menu

Table 4-76: Config VedaBitSync Output Parameters

Parameter	Description
Clock Phase	Choice of 0, 90, 180, 270 degree
TapeCode	NRZ-L, M, S; Bi0-L, M, S; DMMm DMS; RZ, RNRZ9-F, RNRZ11-F, RNRZ15-F, RNRZ9-R, RNRZ11-R, RNRZ15-R

The Command button on the VedaBitSync Module window is not operable.

When you click the Status button on the VedaBitSync Module window, a Status VedaBitSync Module window appears, as shown in **Figure 4-74**.

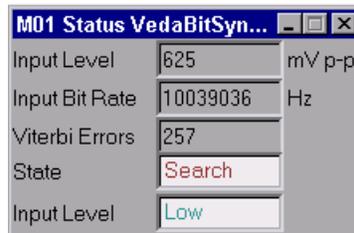


Figure 4-74: Status VedaBitSync Window

Table 4-77 defines the fields in the Status VedaBitSync Module window.

Table 4-77: Status VedaBitSync Parameters

Parameter	Description
Input Level (mV p-p)	Displays input line level as measured by the board in millivolts peak-to-peak
Input Bit Rate (Hz)	Displays actual input bit rate as measured by the board
Viterbi Errors	Displays Viterbi errors as reported by the board (not implemented)

State	Displays lock state as reported by the board
Input Level	Displays input level (ok or low) as reported by the board

DATA PROCESSING MODULES

Section 21: 8 Bit Sorter Module

This Sorter Module receives one stream from its input. Based on an 8 bit or smaller value in a specified byte location of each frame or block, the 8 bit Sorter will range test 4 user-specified ranges of values and send the frame or block out of the corresponding output port of the module. The 8 bit value that is picked out of the data may be bit-order reversed, One's complemented, shifted, and masked as needed. The frames or blocks can also be recorded to separate disk files with a user specified name.

Figure 4-75 shows the 8 Bit Sorter Module window.



Figure 4-75: 8 Bit Sorter Module

Figure 4-76 shows the 8 Bit Sorter Module's Config window.

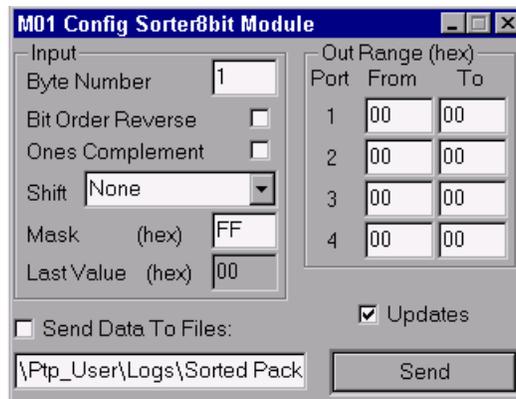


Figure 4-76: 8 Bit Sorter Config Window

Table 4-78 through **Table 4-80** define the parameters in the 8 Bit Sorter Config window.

Table 4-78: 8 Bit Sorter Config Input Parameters

Parameter	Description
Byte Number	Byte number to sort on
Bit Order Reverse	Enable Bit Order Reversing
Ones Complement	Ones Complement Data
Shift	Shift data left or right, 1 to 7 bits

Mask	Mask pattern
Last Value	Last value (in hex) output from Sorter module

Table 4-79: 8 Bit Sorter Config Output Range Parameters

Parameter	Description
Port 1-4	For each output port, define the range of values to be output

Table 4-80: 8 Bit Sorter Config Data Logging Parameters

Parameter	Description
Send Data To Files:	Sorted data can be logged to disk directly from the Sorter module. Check the checkbox to enable recording to disk. The file name is user defined, the file extension is automatically set as the value of the 8 bit or smaller field in the data.

Figure 4-77 shows the 8 Bit Sorter Status window.

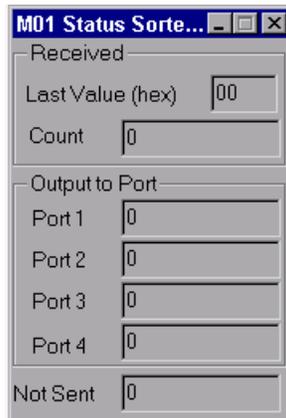


Figure 4-77: 8 Bit Sorter Status Window

Table 4-81 through Table 4-83 define the parameters in the 8 Bit Sorter Status window.

Table 4-81: 8 Bit Sorter Receive Status Parameters

Parameter	Description
Last Value (hex)	Value of last data received (in hex)
Count	Total number of frames/blocks received by Sorter

Table 4-82: 8 Bit Sorter Output Status Parameters

Parameter	Description
Port 1-4	Total number of frames/blocks sent out each Data Port

Table 4-83: 8 Bit Sorter Status Parameters

Parameter	Description
Not Sent	Total number of frames/blocks not sent

Section 22: 16 Bit Sorter Module

This 16 bit Sorter Module receives one stream from its input. Based on an 16 bit or smaller value in a specified byte location (and the following byte) of each frame or block, the 16 bit Sorter will range test 4 user-specified ranges of values and send the frame or block out of the corresponding output port of the module. The 16 bit value that is picked out of the data may be bit-order reversed, One's complemented, shifted, and masked as needed. The frames or blocks can also be recorded to separate disk files with a user specified name.

Figure 4-78 shows the 16 Bit Sorter Module.



Figure 4-78: 16 Bit Sorter Module

Figure 4-79 shows the 16 Bit Sorter Config window.

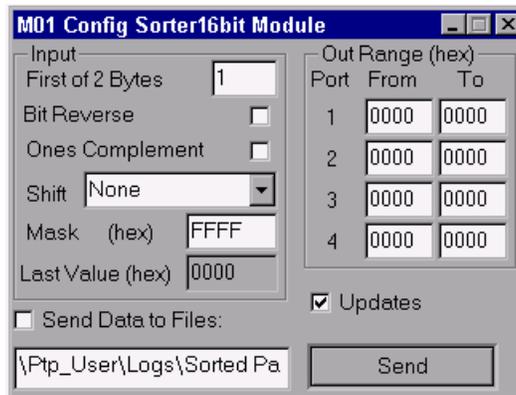


Figure 4-79: 16 Bit Sorter Config Window

Table 4-84 through Table 4-86 define the parameters in the 16 Bit Sorter Config window.

Table 4-84: Config 16 Bit Sorter Input Parameters

Parameter	Description
First of 2 Bytes	Byte number to sort on (first of two bytes)
Bit Reverse	Enables Bit Order Reversing
Ones Complement	Ones Complement Data
Shift	Shift data left or right, 1 to 7 bits
Mask (hex)	Mask pattern
Last Value (hex)	Last value (in hex) output from Sorter module

Table 4-85: Config 16 Bit Sorter Output Range Parameters

Parameter	Description
Port 1-4	For each output port, define the range of values to be output

Table 4-86: Config 16 Bit Sorter Data Logging Parameters

Parameter	Description
Send Data To Files:	Sorted data can be logged to disk directly from the Sorter module. Check the checkbox to enable recording to disk. The file name is user defined, the file extension is automatically set as the value of the 8 bit or smaller field in the data.

Figure 4-80 shows the 16 Bit Sorter Status window.

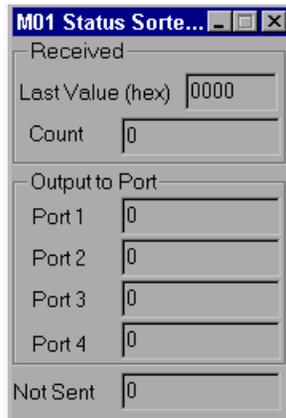


Figure 4-80: 16 Bit Sorter Status Window

Table 4-87 through Table 4-89 define the parameters in the 16 Bit Sorter Status window.

Table 4-87: 16 Bit Sorter Receive Status Parameters

Parameter	Description
Last Value (hex)	Value of last data received (in hex)
Count	Total number of frames/blocks received by Sorter

Table 4-88: 16 Bit Sorter Output Status Parameters

Parameter	Description
Port 1-4	Total number of frames/blocks sent out each Data Port

Table 4-89: 16 Bit Sorter Status Parameters

Parameter	Description
Not Sent	Total number of frames/blocks not sent

Section 23: Best Source Select Module (BSS)

The Best Source Select (BSS) Module is designed for applications where the user has multiple, identical data streams and needs to compare each stream so that only the “best” stream is accepted. The BSS Module accepts multiple input streams, and counts how many frames are received from each during a predefined Sample Period. The BSS Module will switch from the current stream to another stream if the other stream’s frame count exceeds the current streams frame count by the programmable Error Threshold.

Figure 4-81 shows the BSS Module window.



Figure 4-81: BSS Module

When you click the Configure button, a Config BSS Module window appears, as shown in **Figure 4-82**.

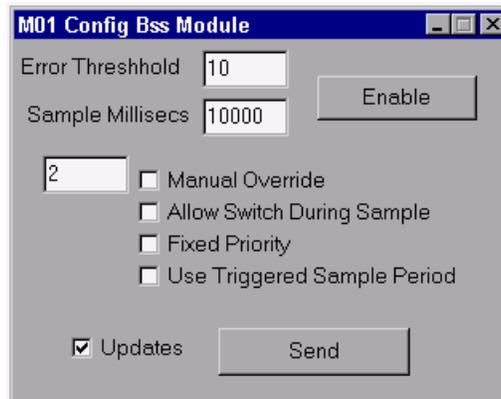


Figure 4-82: Config BSS Module Window

Table 4-90 defines the Config BSS Module window parameters.

Table 4-90: Config BSS Parameters

Parameter	Description
Error Threshold	Defines frame count value for switching from one stream to another
Sample Millisecs	Defines sample period in milliseconds to count number of frames received from each stream
Manual Override	Allows user to manually select output stream (user

			selected stream is input in edit box beside Override checkbox)
Allow Sample	Switch	During	As soon as Error Threshold is reached, the BSS Module will switch streams (by default, the BSS Module will only switch its output source at Sample Period intervals)
Fixed Priority			Fixed priority – starts checking from Stream 1 to check for “good” streams; Rotational – starts checking from current stream to check for “good” stream
Use Period	Triggered	Sample	Enable – synchronizes Sample Period timer to first detected error

When you click the Command button, a Command BSS Module window appears, as shown in **Figure 4-83**.



Figure 4-83: Command BSS Module Window

Table 4-91 defines the Command BSS Module parameters.

Table 4-91: Command BSS Module Parameters

Parameter	Description
No Command Selected	No command selected
Switch Now	Forces the BSS Module to switch streams. The stream that the BSS module will switch is the stream number entered in the Config BSS Module window in the edit box beside the Manual Override checkbox.

When you click the Status button, a Status BSS Module window appears, as shown in **Figure 4-84**.



Figure 4-84: BSS Status Window

Table 4-92 defines BSS Module Status window parameters.

Table 4-92: BSS Status Parameters

Parameter	Description
Active Stream	Displays the current active data stream

Section 24: Bit Density Correction Module

The Bit Density Correction Module is designed to guarantee signal transitions in a bit stream. This is accomplished by XORing an incoming data with a 2047 psuedo-random pattern, regardless of frame size. Encoding is removed from the frame by passing through second instance of module on receiving end.

Figure 4-85 shows the Bit Density Correction Module window.



Figure 4-85: Bit Density Correction Module

There are no other windows associated with this module.

Section 25: Bit Error Rate Tester Module

The Bit Error Rate Tester (BERT) provides the capability to perform end-to-end system testing using pseudo-random number sequences. The BERT uses the IRIG and NASA standard 2047 bit pseudo-random sequence.

The BERT transmitter requires an event input to generate PN data output. The BERT receiver is driven by data input from another module. A typical application uses three modules: BERT, Serial Output, and Serial Input.

The BERT data output is connected to the Serial Output module, the Serial Output module's event output is tied back to the BERT, and the Serial Input module's data output is connected to the BERT module. The BERT module repeatedly generates a 2047 bit pseudo-random sequence and passes it to the serial output module for transmission. The serial input modules receive the pseudo-random data from the communications line and pass it back to the BERT module. The BERT module performs a bit-by-bit comparison between the received data and the original sequence to verify that no bits are in error. The BERT module status indicates how many blocks of PN data have been transmitted and received with and without errors.

Figure 4-86 shows the BERT module window.



Figure 4-86: Bit Error Rate Tester Module Window

When you click the configure button, a Config Bit Error Rate Tester window appears, as shown in **Figure 4-87**.

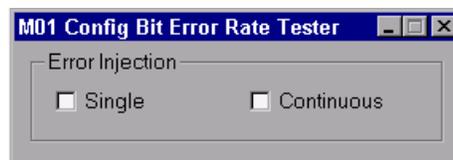


Figure 4-87: Config Bit Error Rate Tester Window

Table 4-93 defines the fields in the Config Bit Error Rate Tester window.

- Error Injection sub-menu

Table 4-93: Config Bit Error Rate Tester Error Injection Parameters

Parameter	Description
Single	Inject a single error into the next frame
Continuous	Inject an error into every frame

When you click the Status button on the BERT module window, a Status Bit Error Rate Tester window will appear, as shown in **Figure 4-88**.

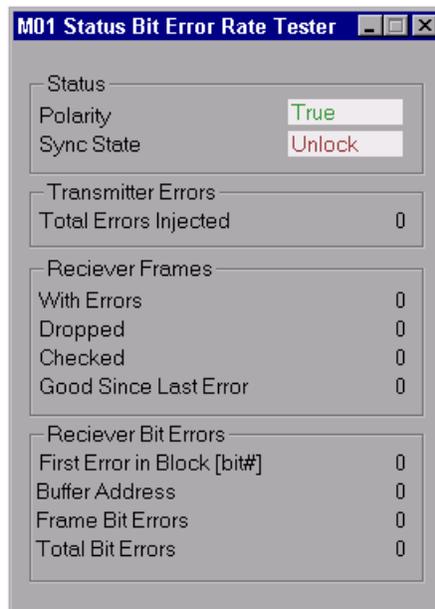


Figure 4-88: Status Bit Error Rate Tester Window

Table 4-94 through **Table 4-97** define the fields in the Status Bit Error Rate Tester window.

- Status sub-menu

Table 4-94: Status Bit Error Rate Tester Status Parameters

Parameter	Description
Polarity	Clock Polarity (True or Inverted)
Sync State	Synchronizer State (Lock or Unlock)

- Transmitter Errors sub-menu

Table 4-95: Status Bit Error Rate Tester Transmitter Errors Parameters

Parameter	Description
Total Errors Injected	Displays Injected Error Count

- Receiver Frames sub-menu

Table 4-96: Status Bit Error Rate Tester Receiver Frames Parameters

Parameter	Description
With Errors	Frames received with errors
Dropped	Frames dropped in the sync process
Checked	Frames checked
Good Since Last Error	Number of good frames received since last error

- Receiver Bit Errors sub-menu

Table 4-97: Status Bit Error Rate Tester Receiver Bit Errors Parameters

Parameter	Description
First Error in Block [bit#]	Displays the bit number of the first bit error
Buffer Address	Address of bad buffer
Frame Bit Errors	Number of frames with bit errors
Total Bit Errors	Total number of bit errors

Section 26: CCSDS Virtual Channel Processor Module

The CCSDS Virtual Channel Processor Module (VCP) receives frame data with quality annotation from another module. The VCP also provides a software CRC decoder and Reed-Solomon decoder. It filters frames based on virtual channel ID and data quality and outputs accepted frames to downstream modules. The VCP supports both Version 1 Transfer Frames and Version 2 Channel Access Data Units. The VCP maintains per virtual channel counts of frames received, frames with RS errors, frames with uncorrectable RS errors, frames with CRC errors, and frames with sequence errors.

The VCP provides multiple data output ports. Each VCP output port can be configured to pass a particular subset of VCDU data. For example, VCP output port 1 could be configured to pass only VCID 0 and VCP output port 2 could be configured to pass VCID 1 and 2.

Figure 4-89 shows the CCSDS VC Processor module window.

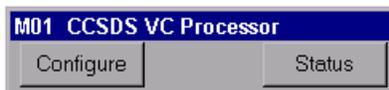


Figure 4-89: CCSDS VC Processor Module Window

When you click the configure button, a Config CCSDS VC Processor window will appear, as shown in **Figure 4-90**.

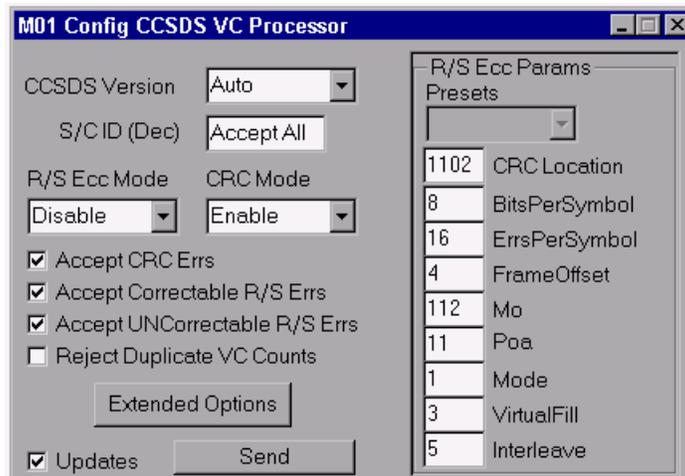


Figure 4-90: Config CCSDS VC Processor Window

Table 4-97 and **Table 4-98** define the fields in the Config CCSDS VC Processor window.

Table 4-98: Config CCSDS VC Processor Parameters

Parameter	Description
CCSDS Version	Defines the expected transfer frame version: Auto - detect version and accept either: Version 1 - accept only Version 1; or Version 2 - accept only version 2
S/C ID (Dec)	Defines the expected Spacecraft ID. Set to Accept All to accept any S/C ID. Set to a particular value to accept only that S/C ID.
R/S Ecc Mode	Controls the software Reed-Solomon decoder. Disable - disables the software RS decoder; Check - enables the software RS decoder for error detection; Correct - enables the software RS decoder for error detection and correction.
CRC Mode	Controls the Software CRC Decoder: Disable – disables software CRC; Enable – enables software CRC; Pre Filter R/S – checks 16-bit CRC, if valid, does not check Reed-Solomon, if CRC not valid, checks Reed-Solomon
Accept CRC Errs	Controls the filtering of frames with CRC errors. Checked, frames with CRC errors are accepted. Unchecked, frames with CRC errors are discarded.
Accept Correctable R/S Errs	Controls the filtering of frames with correctable RS errors. Checked, frames with correctable RS errors are accepted. Unchecked, frames with correctable RS errors are discarded.
Accept UNCorrectable R/S Errs	Controls the filtering of frames with uncorrectable RS errors. Checked, frames with uncorrectable RS errors are accepted. Unchecked, frames with uncorrectable RS errors are discarded.
Reject Duplicate VC counts	Controls the filtering of frames with duplicate VC sequence counts. Checked, discard frames with duplicate VC sequence counts. Unchecked, accept frames with duplicate VC sequence counts.

The R/S Ecc Params sub-menu controls the operation of the software Reed-Solomon decoder and software CRC decoder.

- R/S Ecc Params sub-menu

Table 4-99: Config CCSDS VC Processor R/S Ecc Params Parameters

Parameter	Description
Presets	(Future)
CRC Location	Specifies CRC location
BitsPerSymbol	Specifies Bits Per Symbol for the RS code
ErrsPerSymbol	Specifies Errors Per Symbol for the RS code
FrameOffset	Specifies Frame Offset for the RS code block. Set to 4 to skip the attached sync marker.

Parameter	Description
Mo	Fixed Value; Specifies Mo
Poa	Fixed Value; Specifies Poa
Mode	Fixed Value; Specifies Mode
VirtualFill	Specifies Virtual Fill per codeword
Interleave	Specifies Interleave depth from 1 to 8

When you click the Extended Options button in the Config CCSDS VC Processor window, the window shown in **Figure 4-91** appears.

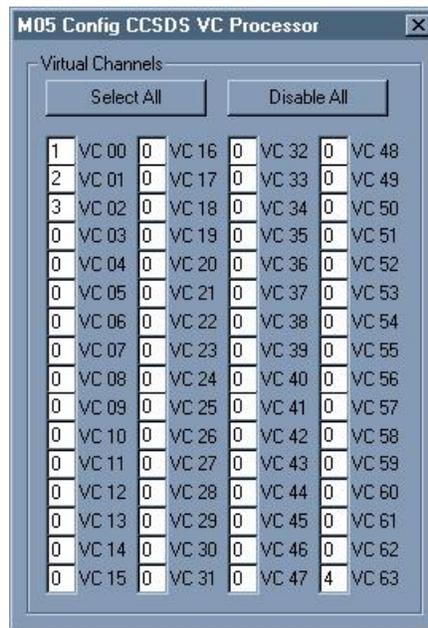


Figure 4-91: Config CCSDS VC Processor Extended Options Window

The CCSDS VC Processor Extended Options controls the routing of transfer frames based on VCID. In the **Figure 4-91** example, frames with VCID 0 will be sent to the module's data port 1, VC 1 will be sent to the module's data port 2, VC 2 will be sent to the module's data port 3, and VC 63 will be sent to the module's data port 4. Frames with any other VCID will be discarded.

Table 4-100 defines the fields in the Config CCSDS VC Processor Extended Options window.

Table 4-100: Config CCSDS VC Processor Extended Options Parameters

Parameter	Description
Select All	Configures all VCs to be output on module data port 1
Disable All	Disables all VC output (sets all data port values to 0)
VC 00 through VC 63	Selects which module data port this VC is to be output. 0 indicates that the data should be discarded.

When you click the Status button on the CCSDS VC Processor module window, a Status CCSDS VC Processor window appears, as shown in **Figure 4-92**.

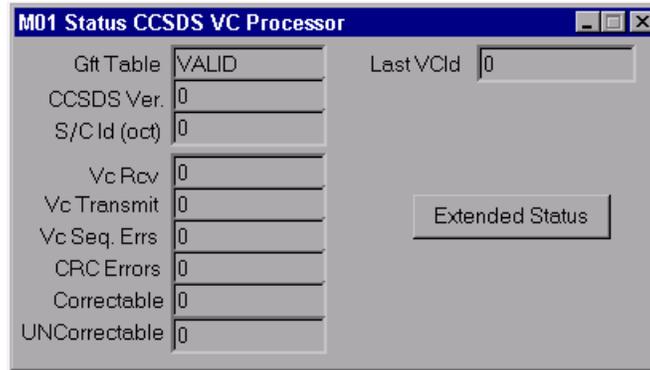


Figure 4-92: Status CCSDS VC Processor Window

Table 4-101 defines the fields in the Status CCSDS VC Processor Window.

Table 4-101: Status CCSDS VC Processor Parameters

Parameter	Description
Gft Table	Valid/Not Valid
CCSDS Ver.	Displays the detected Transfer Frame Version number
S/C Id (oct)	Displays the detected Spacecraft ID
Vc Rcv	Count of total transfer frames received
Vc Transmit	Count of total transfer frames accepted and transmitted
Vc Seq. Errs	Count of total number of frames with VC sequence errors
CRC Errors	Count of total number of frames with CRC errors
Correctable	Count of total number of frames with correctable RS errors
UNCorrectable	Count of total number of frames with uncorrectable RS errors
Last VCID	Displays the VCID of the last frame received

When you click the Extended Status button in the Status CCSDS VC Processor window, the window shown in **Figure 4-93** appears.

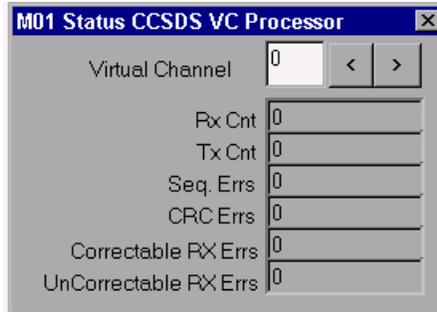


Figure 4-93: Status CCSDS VC Processor Extended Status Window

The VCP Extended Status displays the status information for each VCID. **Table 4-102** defines the fields in the Status CCSDS VCP Extended Status Window.

Table 4-102: Status CCSDS VC Processor Extended Status Parameters

Parameter	Description
Virtual Channel	Selects which VC status is viewed
Rx Cnt	Number of transfer frames received for the selected VC
Tx Cnt	Number of transfer frames accepted and transmitted for the selected VC
Seq. Errs	Sequence Error Count for the selected VC
CRC Errs	CRC Error Count for the selected VC
Correctable RX Errs	Correctable RS Errors for the selected VC
UnCorrectable RX Errs	Uncorrectable RS Errors for the selected VC

Section 27: CCSDS Virtual Channel Simulator Module

The CCSDS Virtual Channel Simulator Module generates Version 1 or Version 2 Transfer Frames. The Virtual Channel Simulator creates Transfer Frames with the 32-bit Attached Sync Marker (1ACFFC1D), a primary header, and fill data in the VCDU data zone. The Virtual Channel Simulator cycles through a preprogrammed virtual channel sequence and generates the sequence count for each frame. The Virtual Channel Simulator also provides software support for CRC encoding and RS encoding.

The Virtual Channel Simulator requires an event input to drive the generation of frames. The Virtual Channel Simulator can be connected to the Serial Output Module to simulate a CCSDS data stream.

Figure 4-94 shows the CCSDS VC Simulator module window.

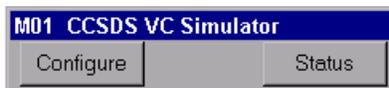


Figure 4-94: CCSDS VC Simulator Module Window

When you click the configure button, a Config CCSDS VC Simulator window appears, as shown in **Figure 4-95**.

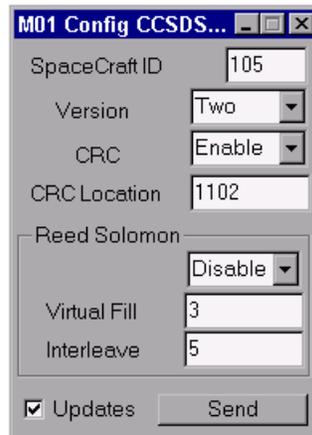


Figure 4-95: Config CCSDS VC Simulator Window

Table 4-103 and **Table 4-104** define the fields in the Config CCSDS VC Simulator Module window.

Table 4-103: Config CCSDS VC Simulator Parameters

Parameter	Description
SpaceCraft ID	Defines the Spacecraft ID in the frame primary header
Version	<p>Selects CCSDS Version:</p> <p>One – Generates Transfer Frames 0, 1, 2, 7</p> <p>Two – Generates Transfer Frames 0, 1, 2, 63</p> <p>1-Fill – Version 1 with VCID inserted as fill pattern across individual VCs</p> <p>2-Fill – Version 1 with VCID inserted as fill pattern across individual VCs</p> <p>GPB – Version 1 with Transfer Frames 0, 1, 2, 3</p> <p>1 BERT – Version 1 with 2047 BERT pattern across all VCs in a single contiguous stream</p> <p>2 BERT – Version 2 with 2047 BERT pattern across all VCs in a single contiguous stream</p> <p>1 VBERT – Version 1 with 2047 BERT pattern across individual VCs in a single contiguous stream</p> <p>2 VBERT – Version 2 with 2047 BERT pattern across individual VCs in a single contiguous stream</p> <p>1 SPBERT – Version 1, generates 1 packet per VC with size equal to entire packet insert zone. Puts a 2047 BERT pattern across individual packets in a single contiguous stream</p> <p>2 SPBERT – Version 2, generates 1 packet per VC with size equal to entire packet insert zone. Puts a 2047 BERT pattern across individual packets in a single contiguous stream</p> <p>1 MPBERT – Version 1 with multiple packets per VC. The size of the first packet in a VC is 200, the next packets increase in size by 4 bytes until the minimum remainder packet of 100 is inserted. Packet ID is derived from VCID and the packet order. Puts a 2047 BERT pattern across individual packets in a single contiguous stream</p> <p>2 MPBERT – Version 2 with multiple packets per VC. The size of the first packet in a VC is 200, the next packets increase in size by 4 bytes until the minimum remainder packet of 100 is inserted. Packet ID is derived from VCID and the packet order. Puts a 2047 BERT pattern across individual packets in a single contiguous stream</p> <p>1 VExt – Version 1 with interleaving of one external data channel/source per virtual channel. Module number of external channel to be interleaved is mapped as the virtual channel number of the CCSDS encapsulated data stream</p> <p>2 VExt – Version 2 with interleaving of one external data channel/source per virtual channel. Module number of external channel to be interleaved is mapped as the virtual</p>

Parameter	Description
	channel number of the CCSDS encapsulated data stream 1 PExt – Version 1 with interleaving of multiple external data sources into a Virtual Channel. Encapsulates data from sources into packets and inserts packets into the transfer frame. Module number of data source is mapped as the APID of the Packet. 2 PExt – Version 2 with interleaving of multiple external data sources into a Virtual Channel. Encapsulates data from sources into packets and inserts packets into the transfer frame. Module number of data source is mapped as the APID of the Packet.
CRC	Enable/Disable software CRC generation
CRC Location	Defines the CRC location

- Reed-Solomon sub-menu

Table 4-104: Config CCSDS VC Simulator Reed-Solomon Parameters

Parameter	Description
Enable/Disable	Enable – enables software R-S encoder; Disable – disables software R-S encoder; Reserve – used when hardware R-S encoding is enabled on Avtec Serial Output, reserves space at end of generated frames for appending R-S codeword
Virtual Fill	Defines the amount of virtual fill per RS code word (0 corresponds to a codeword size of 255, 222 corresponds to a codeword size of 33)
Interleave	Defines the RS encoder interleave depth from 1 to 8

When you click the Status button on the CCSDS VC Simulator window, a Status CCSDS VC Simulator window appears, as shown in **Figure 4-96**.



Figure 4-96: CCSDS VC Simulator Window

Table 4-105 defines the fields in the Status CCSDS VC Simulator window.

Table 4-105: Status CCSDS VC Simulator Parameters

Parameter	Description
VC 0 through VC 63	Displays count of frames generated on each virtual channel ID for VCs 0-7 and 63
Last	Displays the VC ID for the last frame generated

Section 28: Error Inject Module

The Error Inject Module is used to induce errors in a data stream. These errors include throwing away transfer frames, introducing bit errors in transfer frames, or passing a limited number of frames and ignoring the remaining. The Error Inject Module is connected within the data path, i.e. it takes the data stream in on its input port, and transmits the data stream on its output port. If no commands are sent from the Module, the data stream is passed untouched.

Figure 4-97 shows the Error Inject Module.



Figure 4-97: Error Inject Module Window

Figure 4-98 shows the Error Inject Module's Command window.



Figure 4-98: Command Error Inject Window

Table 4-106 defines the fields available in the Command Error Inject Module. The Command function is the only one that is available for this Module.

Table 4-106: Command Error Inject Parameters

Parameter	Description
No Command Selected	No command selected
Skip 5	Throw away 5 consecutive transfer frames
Skip 6	Throw away 6 consecutive transfer frames
Skip 10	Throw away 10 consecutive transfer frames
Trash 5	Insert errors in Sync Pattern for 5 consecutive transfer frames
Trash 6	Insert errors in Sync Pattern for 6 consecutive transfer frames
Trash 10	Insert errors in Sync Pattern for 10 consecutive transfer frames
Accept 100	Accept only 100 transfer frames, then stop accepting frames
Accept 1000	Accept only 1,000 transfer frames, then stop accepting frames
Accept 10000	Accept only 10,000 transfer frames, then stop accepting frames

Section 29: Gather Scatter Module

The Gather Scatter Module is used to extract user-specified bytes from a telemetry frame/packet, discarding the remaining data, and passing only the extracted bytes.

Figure 4-99 shows the Gather Scatter Module window.



Figure 4-99: Gather Scatter Module

Figure 4-100 shows the Config Gather scatter Module window.

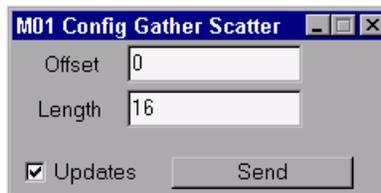


Figure 4-100: Gather Scatter Config Window

Table 4-107 defines the Config Gather Scatter Module window parameters.

Table 4-107: Gather Scatter Config Parameters

Parameter	Description
Offset	Location of first byte to be extracted within frame/packet
Length	Number of bytes from Offset to extract from frame/packet

Section 30: In-Line UNZip Module

The In-Line UNZip Module provides on-the-fly data decompression of data compressed using the In-Line Zip Module. Compressed data is input to the module, and uncompressed data is output to another module. There is no configuration or command window associated with this module.

Figure 4-101 shows the UNZip Module.



Figure 4-101: In-Line UNZip Module

Figure 4-102 shows the UNZip Module Status window.



Figure 4-102: In-Line UNZip Status Window

Table 4-108 defines the fields in the UNZip Module Status window.

Table 4-108: In-Line UNZip Status Fields

Parameter	Description
UnCompression	Displays current uncompression ratio.

Section 31: In-Line Zip Module

The In-Line Zip Module provides on-the-fly data compression. A serial stream is input to this module and a compressed stream is output to another module. There is no configuration or command window associated with this module.

Figure 4-103 shows the Zip Module.



Figure 4-103: In-Line Zip Module

Figure 4-104 shows the Zip Module Status window.



Figure 4-104: In-Line Zip Status Window

Table 4-109 defines the fields in the Zip Module Status window.

Table 4-109: In-Line Zip Status Fields

Parameter	Description
Compression	Displays current compression ratio.

Section 32: Null Transceiver Module

The Null Transceiver Module is essentially a pass-through module that is used to view buffers of data from other modules for debugging purposes. It can also be used to multiplex several data streams into a single data stream. It does not perform any other data processing.

Figure 4-105 shows the Null Transceiver Module Window.



Figure 4-105: Null Transceiver Module Window

When you click the configure button, a Config Null Transceiver window appears, as shown in **Figure 4-106**.

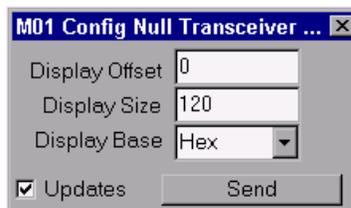


Figure 4-106: Config Null Transceiver Window

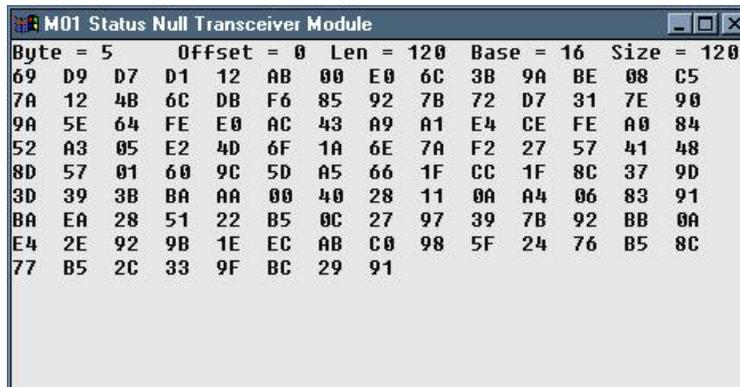
Table 4-110 defines the fields in the Config Null Transceiver window. The Null module configuration parameters define the subset of the buffer data that will be displayed in the Null Status display.

Table 4-110: Config Null Transceiver Parameters

Parameter	Description
Display Offset	Byte Offset into Buffer to be displayed
Display Size	Number of bytes to display
Display Base	Hex, Octal, Decimal, Binary

When you click the Status button on the Null Transceiver window, a Status Null Transceiver window appears, as shown in **Figure 4-107**. The Null Status window displays the raw data from the buffers that are sent to the Null module input port. The data is extracted from the buffer at the location defined in the Null Config parameters. The configuration parameters defined in the Config window are displayed along the top of the Status window. In addition, a *Byte =* field is

available. This field displays the byte position of the value that the mouse pointer is over. The Status window can also be resized.



The screenshot shows a window titled "M01 Status Null Transceiver Module" with a hex dump of data. The window includes a title bar with standard Windows controls (minimize, maximize, close). Below the title bar, the following information is displayed: "Byte = 5", "Offset = 0", "Len = 120", "Base = 16", and "Size = 120". The main area of the window contains a grid of hexadecimal values arranged in rows and columns. The first row contains 13 bytes: 69, 09, 07, 01, 12, AB, 00, E0, 6C, 3B, 9A, BE, 08, C5. The second row contains 13 bytes: 7A, 12, 4B, 6C, DB, F6, 85, 92, 7B, 72, 07, 31, 7E, 90. The third row contains 13 bytes: 9A, 5E, 64, FE, E0, AC, 43, A9, A1, E4, CE, FE, A0, 84. The fourth row contains 13 bytes: 52, A3, 05, E2, 4D, 6F, 1A, 6E, 7A, F2, 27, 57, 41, 48. The fifth row contains 13 bytes: 8D, 57, 01, 60, 9C, 5D, A5, 66, 1F, CC, 1F, 8C, 37, 9D. The sixth row contains 13 bytes: 3D, 39, 3B, BA, AA, 00, 40, 28, 11, 0A, A4, 06, 83, 91. The seventh row contains 13 bytes: BA, EA, 28, 51, 22, B5, 0C, 27, 97, 39, 7B, 92, BB, 0A. The eighth row contains 13 bytes: E4, 2E, 92, 9B, 1E, EC, AB, C0, 98, 5F, 24, 76, B5, 8C. The ninth row contains 13 bytes: 77, B5, 2C, 33, 9F, BC, 29, 91.

Byte	09	07	01	12	AB	00	E0	6C	3B	9A	BE	08	C5
69	09	07	01	12	AB	00	E0	6C	3B	9A	BE	08	C5
7A	12	4B	6C	DB	F6	85	92	7B	72	07	31	7E	90
9A	5E	64	FE	E0	AC	43	A9	A1	E4	CE	FE	A0	84
52	A3	05	E2	4D	6F	1A	6E	7A	F2	27	57	41	48
8D	57	01	60	9C	5D	A5	66	1F	CC	1F	8C	37	9D
3D	39	3B	BA	AA	00	40	28	11	0A	A4	06	83	91
BA	EA	28	51	22	B5	0C	27	97	39	7B	92	BB	0A
E4	2E	92	9B	1E	EC	AB	C0	98	5F	24	76	B5	8C
77	B5	2C	33	9F	BC	29	91						

Figure 4-107: Status Null Transceiver Window

Section 33: Packet Processor Module

The Packet Processor Module extracts packets from a single virtual channel of a CCSDS telemetry stream. It receives a stream of transfer frames and outputs a stream of packets. The user defines a list of Application IDs (APID) and the Packet Processor will strip packets with matching APIDs to its data output for another module. In general, the Packet Processor is preceded by the CSDS Virtual Channel Processor Module which sorts frames based on virtual Channel ID.

Figure 4-108 shows the Packet Processor Module window.



Figure 4-108: PacketProcessor Module Window

When you click the configure button, a Config PacketProcessor Module Window will appear, as shown in **Figure 4-109**.

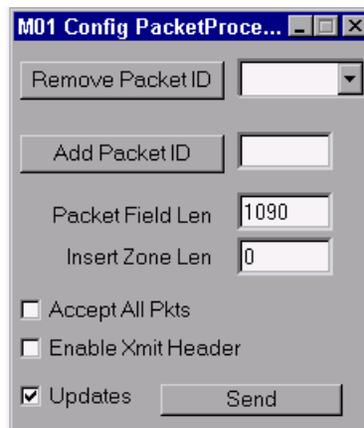


Figure 4-109: Config PacketProcessor Window

The Config Packet Processor module window is used to define the list of valid APIDs. The Packet Processor will discard all packets that have APIDs that are not in the list of valid APIDs.

Table 4-111 defines the fields in the Config PacketProcessor Module window.

Table 4-111: Config PacketProcessor Parameters

Parameter	Description
Remove Packet ID	Removes the selected packet ID from the filter list

Add Packet ID	Adds the entered packet ID to the filter list
Packet Field Len	Length of the Packet Data Field within the Transfer Frame
Insert Zone Len	Defines Secondary Transfer Frame Header Length
Accept All Pkts	Accepts all Packets IPs (APID)
Enable Xmit Header	Controls the output of the packet header. Checked enables the output of the packet header. Unchecked, output packet data only.

When you click the Command button on the PacketProcessor module window, a Command PacketProcessor window will appear, as shown in **Figure 4-110**.



Figure 4-110: Command PacketProcessor Window

Table 4-112 defines the field in the Command PacketProcessor Module window.

Table 4-112: Command PacketProcessor Parameter

Parameter	Description
Command	(Not implemented)

When you click the Status button on the PacketProcessor module window, a Status PacketProcessor window will appear, as shown in **Figure 4-111**.

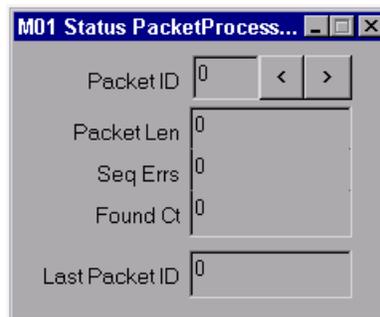


Figure 4-111: Status PacketProcessor Window

Table 4-113 defines the fields in the Status PacketProcessor Module window.

Table 4-113: Status PacketProcessor Parameters

Parameter	Description
Packet ID	Selects which APID is to be viewed
Packet Len	Displays the packet length of the last packet with the specified APID
Seq Errs	Displays the count of sequence errors of the packet specified above
Found Ct	Displays the count of packets with specified APID
Last Packet ID	Last packet ID received

Section 34: TDM Simulator Module (TDMSim)

The TDM Simulator Module (Time Division Multiplex Simulator) simulates a user-defined TDM data stream. It is an event-driven module, and therefore requires a CPU Time Module or Serial Output Module to generate an event for each frame of data. The user can configure multiple ID counters and Fill Patterns for the simulated stream. An option to interleave a second stream from the Serial Input module is also available. A CRC codeword can also be added to the stream.

Figure 4-112 shows the TDM Simulator Module.



Figure 4-112: TDM Simulator Module

Figure 4-113 shows the TDM Simulator Module's Config window.

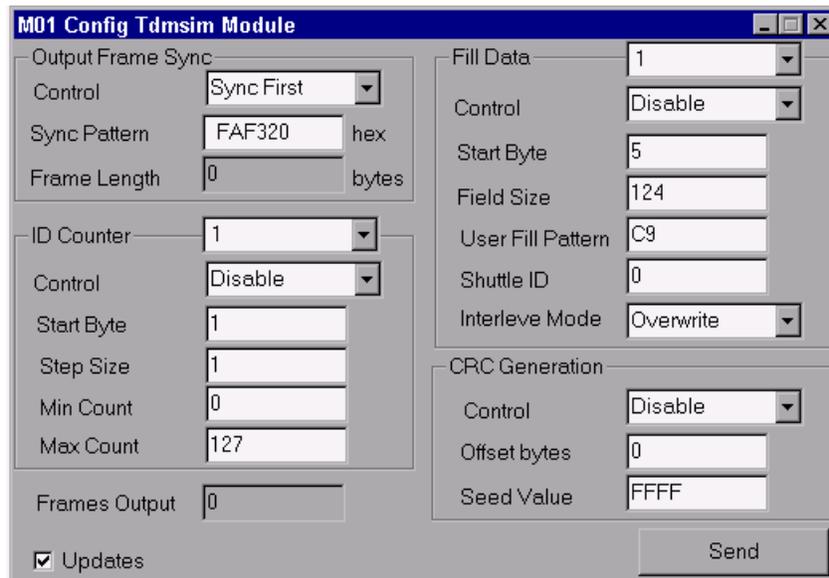


Figure 4-113: Config TDM Simulator Window

Table 4-114 through Table 4-117 define the parameters in the TDM Simulator Module's Config window.

Table 4-114: Config TDMSim Output Frame Sync Parameters

Parameter	Description
-----------	-------------

Control	Places sync at beginning (first) or end (last) of frame
Sync Pattern	Configure Sync Pattern (up to 64 bits long)
Frame Length	Configure Frame Length

Table 4-115: Config TDMSim ID Counter Parameters

Parameter	Description
ID Counter	Up to 30 ID counters can be inserted in frame – this parameter selects which ID Counter to configure
Control	Enable (select type of counter) or Disable selected ID Counter. Types of counters available include:
Start Byte	Select byte in frame to insert start of counter
Step Size	Value to increment counter
Min Count	Minimum value of counter (start value)
Max Count	Maximum value of counter (end value)

Table 4-116: Config TDMSim Fill Data Parameters

Parameter	Description
Fill Data	Up to 30 fill patterns can be inserted into frame – this parameter selects which fill pattern to configure
Control	Enable (select type of fill pattern) or Disable selected fill pattern. Types of fill data include: Dynamic (pseudo-random pattern), Byte Position (inserts the byte position as the fill pattern), ID Counter, User Specified, Normal I/L (allows a second stream obtained from the Serial Input Module to be interleaved, byte by byte, with the User Fill Pattern), or Shuttle I/L (same as Normal I/L, but also inserts PB4 time in front of the data)
Start Byte	Select byte in frame to insert start of fill pattern
Field Size	Size of fill pattern in bytes
User Fill Pattern	Defines Fill Pattern to be used when Control parameter is set to either User Specified, Normal I/L, or Shuttle I/L.
Shuttle ID	Shuttle ID value
Interleave Mode	Overwrite or XOR: Defines method of interleaving Serial Input stream and User Fill Pattern for Normal I/L and Shuttle I/L modes.

Table 4-117: Config TDMSim CRC Generation Parameters

Parameter	Description
Control	Disables CRC or uses Normal or Inverted CRC codeword
Offset bytes	Byte Offset to insert CRC codeword
Seed Value	Seed value used for CRC generation

Figure 4-114 shows the TDM Simulator Module's Status window.



Figure 4-114: TDM Simulator Status Window

Table 4-118 defines the parameters in the TDM Simulator Module's Status window.

Table 4-118: TDM Simulator Status Parameters

Parameter	Description
Frame Output	Displays number of frames that have been transmitted

Section 35: Unknown Module

Figure 4-115 shows the Unknown Module window. You can actually load the Unknown module, but it is not designed as a functional module. The Unknown module is invoked by the Console task when a server is connected which has a (probably) new module on the server, one that is not located on the Client Console disk. This should be considered a system management problem and the appropriate personnel should be contacted if the Unknown module is displayed.



Figure 4-115: Unknown Module Window

ENCAPSULATION/EXTRACTION MODULES

Section 36: ACE SFDU Formatter Module

The ACE SFDU Formatter Module supports encapsulation of telemetry frames with a Standard Formatted Data Unit header. The header format is based on the definitions as needed for the Advanced Composition Explorer. Frame synchronization states are recorded within the SFDU header only if a serial input module using the MONARCH-E board is loaded as a source of the telemetry stream. In addition the Reed Solomon decoder and Extended Status must be enabled.

Note: See DSN Document 820-13, Rev. A – TLM-3-27 *DSN Telemetry Interface with Advanced Composition Explorer (ACE)* for a detailed overview of the ACE SFDU format.

Figure 4-116 shows the ACE SFDU Formatter Module window.



Figure 4-116: ACE SFDU Module

Figure 4-117 shows the Config ACE SFDU Module window.

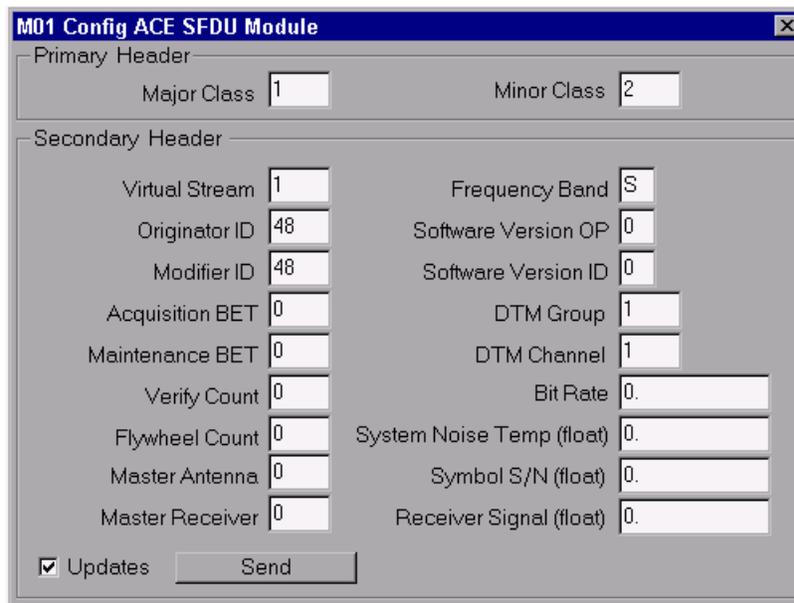


Figure 4-117: ACE SFDU Config Window

Table 4-119 and **Table 4-120** defines the Config ACE SFDU Formatter Module window parameters.

Table 4-119: ACE SFDU Primary Header Config Parameters

Parameter	Description
Major Class	Major Data Class; Default = 1 (Spacecraft telemetry data)
Minor Class	Minor Data Class. Indicates configuration of DSCC (Deep Space Communications Complex) frame sync and Reed-Solomon decoding processes. Valid values = 0, 1, 2

Table 4-120: ACE SFDU Secondary Header Config Parameters

Parameter	Description
Virtual Stream	Virtual Stream Identifier; value = 1, 2, 64 (i.e. virtual channel 1)
Originator ID	Indicates message origin; 48=originated from DSN (Deep Space Network)
Modifier ID	Indicates where message was last modified; 48=last modified by DSN
Acquisition BET	Acquisition Bit Error Tolerance – defines number of allowable bit errors in sync pattern in search and verify modes; values=0 – 15
Maintenance BET	Defines number of allowable bit errors in sync pattern in lock and flywheel modes; values=0 – 15
Verify Count	Number of frames required for frame sync logic to transition from verify to lock state; values=0 – 15
Flywheel Count	Number of frames required for frame sync logic to transition from flywheel to search state; values=0 – 15
Master Antenna	Indicates which DSS antenna data was received from.
Master Receiver	Indicates which receiver was used with Master Antenna
Frequency Band	Indicates frequency band of data; values = S, X, K
Software Version OP	Software Version Operational Level; assigned by DSN Operations
Software Version ID	Software Version Identifier; assigned by DTM CDE
DTM Group	DSCC Telemetry Subsystem Group number
DTM Channel	DTM Channel number
Bit Rate	Bit Rate in bits per second
System Noise Temp (float)	System Noise Temperature in Kelvin
Symbol S/N (float)	Estimated signal-to-noise ratio (dB) in symbol domain (prior to Viterbi decoding)
Receiver Signal (float)	Receiver signal level (dBm) for master receiver in array

Figure 4-118 shows the ACE SFDU Module Status window.

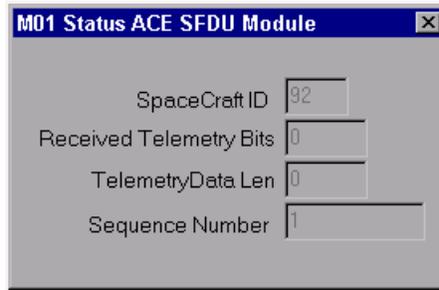


Figure 4-118: ACE SFDU Status Window

Table 4-121 defines the ACE SFDU Module Status window parameters.

Table 4-121: ACE SFDU Status Parameters

Parameter	Description
SpaceCraft ID	Displays Spacecraft ID number
Received Telemetry Bits	Count of received bits
TelemetryData Len	Telemetry data length
Sequence Number	Displays VC Sequence Number

Section 37: Ames Command Encapsulator Module

The Ames Command Encapsulator Module is used to encapsulate command packets for serial output. The Command Encapsulator receives a command packet and prepends a Preamble and a BarkerCode and appends a Postamble. The Command Encapsulator outputs the new packet.

Figure 4-119 shows the Ames Command Encapsulator module window.



Figure 4-119: Ames Command Encapsulator Module Window

When you click the configure button, a Config Ames Command Encapsulator window appears, as shown in Figure 4-120.

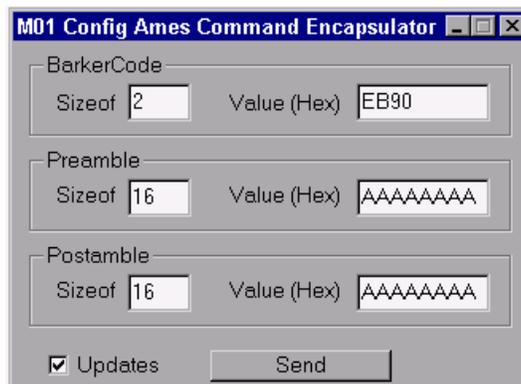


Figure 4-120: Config Ames Command Encapsulator Window

Table 4-122 through Error! Reference source not found. define the fields in the Config Ames Command Encapsulator window.

- BarkerCode sub-menu

Table 4-122: Config Ames Command Encapsulator BarkerCode Parameters

Parameter	Description
Sizeof	Size of the Barker Code in bytes
Value (Hex)	Value of barker code

- Preamble sub-menu

Table 4-123: Config Ames Command Encapsulator Preamble Parameters

Parameter	Description
Sizeof	Size of the Preamble in bytes
Value (Hex)	Value of preamble

- Postamble sub-menu

Table 4-124: Config Ames Command Encapsulator Postamble Parameters

Parameter	Description
Sizeof	Size of the Postamble in bytes
Value (Hex)	Value of postamble

When you click the Command button on the Ames Command Encapsulator window, a Command Ames Command Encapsulator window will appear, as shown in **Figure 4-121**. No commands are currently defined for the Ames Command Encapsulator.



Figure 4-121: Command Ames Command Encapsulator Window

The Status button on the Ames Command Encapsulator module window is not yet operable.

Section 38: AXAF SFDU Formatter Module

The AXAF SFDU Formatter Module supports encapsulation of telemetry frames with a Standard Formatted Data Unit header. The header format is based on the definitions as needed for the Advanced X-Ray Astrophysics Facility (AXAF). Frame synchronization states are recorded within the SFDU header only if a serial input module using the Monarch-E board is loaded as a source of the telemetry stream. In addition the Reed Solomon decoder and Extended Status must be enabled.

Note: See DSN Document 820-13, Rev. A – TLM-3-26 *DSN Telemetry Interface with MSFC for the Advanced X-ray Astrophysics Facility – Imaging (AXAF-I) Project* for a detailed overview of the AXAF SFDU format.

Figure 4-122 shows the AXAF SFDU Module window.



Figure 4-122: AXAF SFDU Module

Figure 4-123 shows the Config AXAF SFDU Module window.

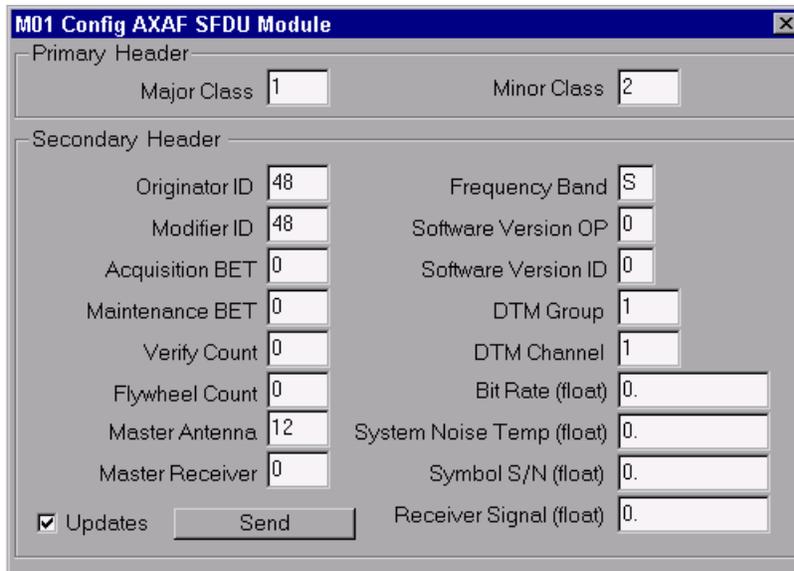


Figure 4-123: AXAF SFDU Config Window

Table 4-125 and Table 4-126 defines the Config AXAF SFDU Module window parameters.

Table 4-125: AXAF SFDU Primary Header Config Parameters

Parameter	Description
Major Class	Major Data Class; Default = 1 (Spacecraft telemetry data)
Minor Class	Minor Data Class. Indicates configuration of DSCC (Deep Space Communications Complex) frame sync and Reed-Solomon decoding processes. Valid values = 0, 1, 2; AXAF-I, value=2

Table 4-126: AXAF SFDU Secondary Header Config Parameters

Parameter	Description
Originator ID	Indicates message origin; 48=originated from DSN (Deep Space Network)
Modifier ID	Indicates where message was last modified; 48=last modified by DSN
Acquisition BET	Acquisition Bit Error Tolerance – defines number of allowable bit errors in sync pattern in search and verify modes; values=0 – 15
Maintenance BET	Defines number of allowable bit errors in sync pattern in lock and flywheel modes; values=0 – 15
Verify Count	Number of frames required for frame sync logic to transition from verify to lock state; values=0 – 15
Flywheel Count	Number of frames required for frame sync logic to transition from flywheel to search state; values=0 – 15
Master Antenna	Indicates which DSS antenna data was received from.
Master Receiver	Indicates which receiver was used with Master Antenna
Frequency Band	Indicates frequency band of data; values = S, X, K
Software Version OP	Software Version Operational Level; assigned by DSN Operations
Software Version ID	Software Version Identifier; assigned by DTM CDE
DTM Group	DSCC Telemetry Subsystem Group number
DTM Channel	DTM Channel number
Bit Rate (float)	Measured bit rate of received telemetry data in bits per second
System Noise Temp (float)	System Noise Temperature in Kelvin
Symbol S/N (float)	Estimated signal-to-noise ratio (dB) in symbol domain (prior to Viterbi decoding)
Receiver Signal (float)	Receiver signal level (dBm) for master receiver in array

Figure 4-124 shows the AXAF SFDU Module Status window.

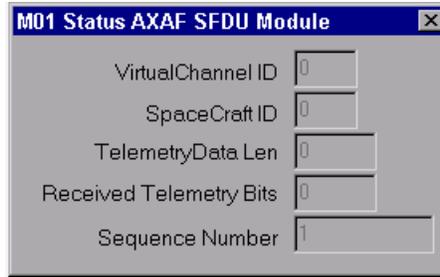


Figure 4-124: AXAF SFDU Status Window

Table 4-127 defines the AXAF SFDU Module Status window parameters.

Table 4-127: AXAF SFDU Status Parameters

Parameter	Description
VirtualChannel ID	Displays VCID number
SpaceCraft ID	Displays Spacecraft ID
TelemetryData Len	Displays telemetry data length
Received Telemetry Bits	Count of received bits
Sequence Number	Displays sequence number

Section 39: DST SFDU Formatter Module

The DST SFDU Formatter module supports encapsulation of telemetry frames with a Standard Formatted Data Unit header. The header format is based on the definitions as needed for the Deep Space Terminal. Frame synchronization states are recorded within the SFDU header only if a serial input module using the MONARCH-E board is loaded as a source of the telemetry stream. In addition the Reed Solomon decoder and Extended Status must be enabled.

Note: See *SFOC Software Interface Specification for the Mars Observer Spacecraft Telemetry Data SFDU*, Document SFOC-1-GIF-DSN-MOGCFT1m for a detailed overview of the DST SFDU format.

Figure 4-125 shows the DST SFDU Module window.



Figure 4-125: DST SFDU Module

Figure 4-126 shows the Config DST SFDU Module window.

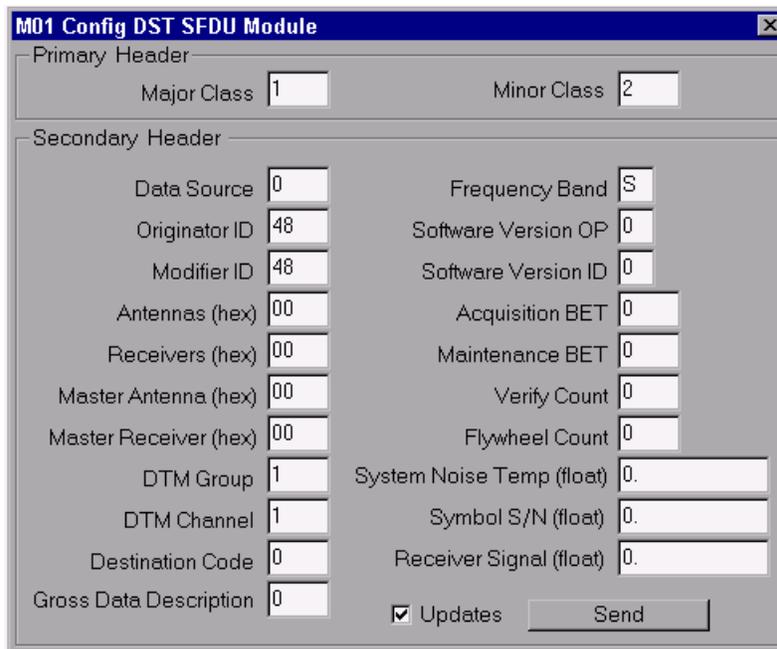


Figure 4-126: DST SFDU Config Window

Table 4-129 and Table 4-130 displays the Config DST SFDU Module window parameters.

Table 4-128: DST SFDU Primary Header Config Parameters

Parameter	Description
Major Class	Major Data Class; Default = 1 (Spacecraft telemetry data)
Minor Class	Minor Data Class. Indicates configuration of DSCC (Deep Space Communications Complex) frame sync and Reed-Solomon decoding processes. Valid values = 0, 1, 2

Table 4-129: DST SFDU Secondary Header Config Parameters

Parameter	Description
Data Source	Data Source number
Originator ID	Indicates message origin; 48=originated from DSN (Deep Space Network)
Modifier ID	Indicates where message was last modified; 48=last modified by DSN
Antennas (hex)	Antennas in use
Receivers (hex)	Receivers in use
Master Antenna (hex)	Indicates which DSS antenna data was received from
Master Receiver (hex)	Indicates which receiver was used with Master Antenna
DTM Group	DSCC Telemetry Subsystem Group number
DTM Channel	DTM Channel number
Destination Code	Destination Code
Gross Data Description	
Frequency Band	Indicates frequency band of data; values = S, X, K
Software Version OP	Software Version Operational Level; assigned by DSN Operations
Software Version ID	Software Version Identifier; assigned by DTM CDE
Acquisition BET	Acquisition Bit Error Tolerance – defines number of allowable bit errors in sync pattern in search and verify modes; values=0 – 15
Maintenance BET	Defines number of allowable bit errors in sync pattern in lock and flywheel modes; values=0 – 15
Verify Count	Number of frames required for frame sync logic to transition from verify to lock state; values=0 – 15
Flywheel Count	Number of frames required for frame sync logic to transition from flywheel to search state; values=0 – 15
System Noise Tem (float)	System Noise Temperature in Kelvin
Symbol S/N (float)	Estimated signal-to-noise ratio (dB) in symbol domain (prior to Viterbi decoding)
Receiver Signal (float)	Receiver signal level (dBm) for master receiver in array

Figure 4-127 show the DST SFDU Module Status window.

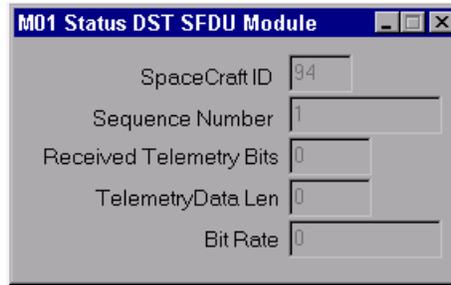


Figure 4-127: DST SFDU Status Window

Table 4-130 defines the DST SFDU Module status window parameters.

Table 4-130: DST SFDU Status Parameters

Parameter	Description
SpaceCraft ID	Displays Spacecraft ID
Sequence Number	Displays sequence number
Received Telemetry Bits	Count of received bits
TelemetryData Len	Displays telemetry data length
Bit Rate	Bit Rate in bits per second

Section 40: EDOS Service Header Module

The EDOS Service Header Module is a packet or frame service header module. This module accepts packets from the Packet Processor Module or from formatted packet files, or VCDUs or CLCWs from the Scatter Gather Module. The module then creates a different header for each data unit that contains data statistics, including number of VC sequence errors, and number of packet sequence errors. The output of this module is the original packet with the EDOS Service Header prepended.

Note: See Appendix B for a detailed overview of the EDOS Service Header.

Figure 4-128 is the EDOS Module.



Figure 4-128: EDOS Service Header Module

Figure 4-129 shows the EDOS Module Configuration window.

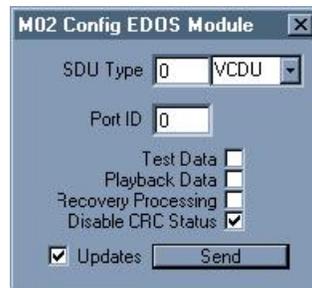


Figure 4-129: EDOS Service Header Config Window

Table 4-131 defines the parameters in the EDOS configuration window.

Table 4-131: EDOS Service Header Configuration Parameters

Parameter	Description
SDU Type	Defines SDU (Service Data Unit) type, i.e. type of data being received by this module – 0 = VCDU, 1 = CCSDS packet, 2 = CLCW
Port ID	EDOS physical port identification; Range 0 – 63
Test Data	If enabled, checks logical bit to determine

Parameter	Description
	data type; 0 = live operational data, 1 = EDOS test data.
Playback Data	If enabled, checks logical bit to determine if VCDU contains playback data; 0 = non-playback data, 1 = playback data/data from on-board spacecraft recorder.
Recovery Processing	If enabled, checks logical bit to determine if data is from EDOS's data capture recovery processing; 0 = live data, 1 = data capture playback data.
Disable CRC Status	If enabled, does not check CRC status.

Figure 4-130 shows the EDOS Service Header Status window.



Figure 4-130: EDOS Service Header Status Window

Table 4-132 defines the parameters in the EDOS Status window.

Table 4-132: EDOS Service Header Status Parameters

Parameter	Description
Virtual Channel ID	Displays the VCID number
Spacecraft ID	Displays the Spacecraft ID number
ESH Version	Display the EDOS Service Header version number

Section 41: IPDU Formatter Module

The IPDU Formatter Module encapsulates data for transmission across the network. The IPDU Formatter accepts data buffers from other modules and prepends the IPDU header. The IPDU header format is defined in Appendix B. The IPDU Formatter outputs the IPDU packet (header and data).

The IPDU Formatter typically receives telemetry frames with quality annotation from the Serial Input Module or the VCP Module. The IPDU Formatter is typically connected to a Socket Module for network output.

The IPDU Formatter could also be used in an operations center to encapsulate commands for network transmissions.

Note: See Appendix B for a detailed overview of the IPDU format.

Figure 4-131 shows the IPDU Formatter Module window.



Figure 4-131: IPDU Formatter Module Window

When you click the configure button, a Config IPDU Formatter Module window will appear, as shown in **Figure 4-132**.

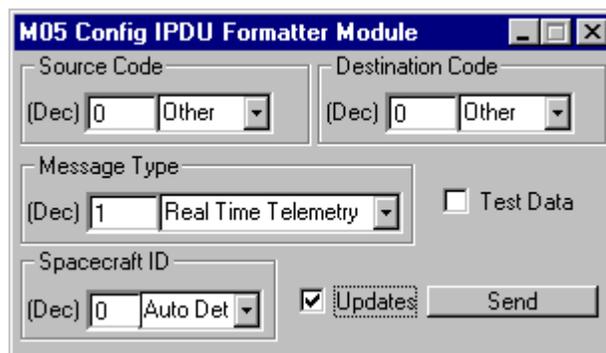


Figure 4-132: Config IPDU Formatter Window

The Config IPDU Formatter is used to set user-programmable fields within the IPDU header. The other fields in the IPDU header such as RS data quality and time stamp are filled with information taken from each buffer on a frame-by-frame basis.

Table 4-133 through **Table 4-136** define the fields in the Config IPDU Formatter Module window.

- Source Code sub-menu

Table 4-133: Config IPDU Formatter Source Code Parameters

Parameter	Description
Source Code	Choice of L7M, SFS, ASK, SPZ, WSC, WIL, VLF, VAF, WFF, or Other (with value specified in decimal)

- Destination Code sub-menu

Table 4-134: Config IPDU Formatter Destination Code Parameters

Parameter	Description
Destination Code	Choice of L7M, SFS, ASK, SPZ, WSC, WIL, VLF, VAF, WFF, or Other (with value specified in decimal)

- Message Type sub-menu

Table 4-135: Config IPDU Formatter Message Type Parameters

Parameter	Description
Message Type	Choice of RealTime Telemetry, Playback Telemetry, RealTime Command, Command Echo, Other (with value specified in decimal)

- Spacecraft ID sub-menu

Table 4-136: Config IPDU Formatter Spacecraft ID Parameters

Parameter	Description
Spacecraft ID	Choice of Auto Detect, LSAT7, XTE, or Other (with value specified in decimal)

There is no status window for the IPDU Formatter module.

Section 42: IPDU Receiver Module

The IPDU Receiver Module is a special type of Socket Module. It receives IPDU packets from a network socket connection. The IPDU header is defined in Appendix B. The IPDU Receiver detects the start of an IPDU packet using the IPDU header sync pattern. The IPDU Receiver filters the packet based on the IPDU header. The IPDU module has two data output ports. Port 1 outputs accepted packets with or without the IPDU header. Port 2 is used for command echo and outputs accepted packets with the source and destination code swapped for transmission back to the control center.

Note: See Appendix B for a detailed overview of the IPDU format. **Figure 4-133** shows the IPDU Receiver Module window.



Figure 4-133: IPDU Receiver Module Window

When you click the configure button, a Config IPDU Receiver Module window will appear, as shown in **Figure 4-134**.

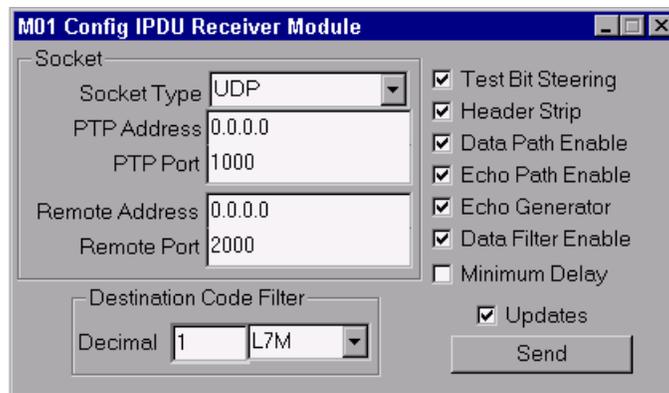


Figure 4-134: Config IPDU Receiver Window

Table 4-137 through **Table 4-139** define the fields in the Config IPDU Receiver Module window.

- Socket sub-menu

Table 4-137: Config IPDU Receiver Socket Parameters

Parameter	Description
Socket Type	Sets the socket type: Disabled, UDP, TCP Client, or TCP Server

Parameter	Description
PTP Address	Sets IP address (0.0.0.0 specifies default host address)
PTP Port	Socket number
Remote Address	Remote IP address
Remote Port	Remote Socket number

Table 4-138: Config IPDU Receiver Parameters

Parameter	Description
Test Bit Steering	Used to test echo path; Enable – checks for test bit, if test bit is high, sends echo and throws away data; Disable – does not check for test bit.
Header Strip	Enable – strips header on Data output; Disable – passes header and data
Data Path Enable	Opens/closes Data Port 1 (data path)
Echo Path Enable	Opens/closes Data Port 2 (echo path)
Echo Generator	Enable – swaps destination and source code in header and sends command echo
Data Filter Enable	Enable – passes only command telemetry out Data Port 1; Disable – passes command and telemetry out Data Port 1
Minimum Delay	Disables TCP nagle algorithm and sets socket priority to highest

- Destination Code Filter sub-menu

Table 4-139: Config IPDU Receiver Destination Code Filter Parameters

Parameter	Description
Destination Code Filter	Choice of L7M, SFS, ASK, SPZ, WSC, WIL, VLF, VAF, WFF, Disabled, or Other (with value specified in decimal)

When you click the Status button on the IPDU Receiver Module window, a Status IPDU Receiver Module window will appear, as shown in **Figure 4-135**.

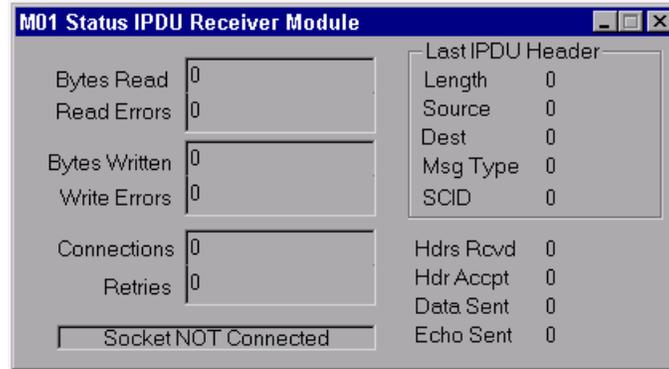


Figure 4-135: Status IPDU Receiver Window

Table 4-140 and Table 4-141 define the fields in the Status IPDU Receiver Module window.

Table 4-140: Status IPDU Receiver Parameters

Parameter	Description
Bytes Read	Count of bytes read
Read Errors	Count of read errors
Bytes Written	Count of bytes written
Write Errors	Count of write errors
Connections	Count of connections
Retries	Count of retried connections
Hdrs Rcvd	Count of headers received
Hdr Accpt	Count of headers accepted
Data Sent	Count of data units sent
Echo Sent	Count of echoes sent

- Last IPDU Header sub-menu

Table 4-141: Status IPDU Receiver Last IPDU Header Parameters

Parameter	Description
Length	Last received header data length
Source	Last received source code
Dest	Last received destination code
Msg Type	Last received message type
SCID	Last received Spacecraft ID

Section 43: LEO-T CDH Formatter Module

The LEO-T CDH (Command Delivery Header) Formatter Module encapsulates command data for transmission across the network. The LEO-T CDH Formatter accepts data buffers from other modules and prepends the CDH header. The LEO-T CDH Formatter outputs the CDH packet (header and data).

The LEO-T CDH Formatter typically receives command frames with quality annotation from the Serial Input Module or the VCP Module. The LEO-T CDH Formatter is typically connected to a Socket Module for network output.

Note: See Appendix B for a detailed overview of the LEO-T CDH format.

Figure 4-136 shows the LEO-T CDH Formatter Module.



Figure 4-136: LEO-T CDH Formatter Module

Figure 4-137 shows the Config LEO-T CDH Formatter Module window.

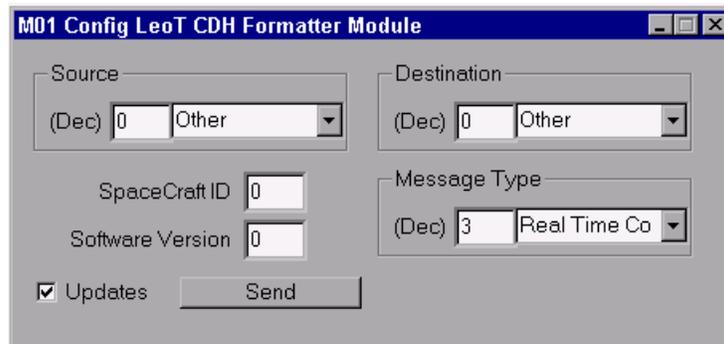


Figure 4-137: LEO-T CDH Formatter Config Window

Table 4-142 defines the Config LEO-T CDH Formatter Module window parameters.

Table 4-142: LEO-T CDH Formatter Config Parameters

Parameter	Description
Source (Dec)	Choice of L7M, SFS, ASK, SPZ, WSC, WIL, VLF, VAF, WFF, or Other (with value specified in decimal)
Destination (Dec)	Choice of L7M, SFS, ASK, SPZ, WSC,

	WIL, VLF, VAF, WFF, or Other (with value specified in decimal)
Message Type (Dec)	Choice of RealTime Telemetry, Playback Telemetry, RealTime Command, Command Echo, Other (with value specified in decimal)
SpaceCraft ID	Choice of Auto Detect, LSAT7, XTE, or Other (with value specified in decimal)
Software Version	EDOS Software version number.

Figure 4-138 shows the LEO-T CDH Formatter Status window.

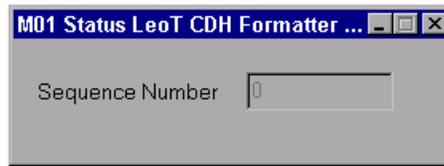


Figure 4-138: LEO-T CDH Formatter Status Window

Table 4-143 defines the LEO-T CDH Formatter Module Status window parameters.

Table 4-143: LEO-T CDH Formatter Status Parameters

Parameter	Description
Sequence Number	Displays current sequence number

Section 44: LEO-T CDH Receiver Module

The LEO-T CDH Receiver Module is a special type of Socket Module. It receives LeoT CDH packets from a network socket connection. The LEO-T CDH Receiver filters the packet based on the LEO-T CDH header. The LEO-T CDH module has two data output ports. Port 1 outputs accepted packets with or without the LEO-T CDH header. Port 2 is used for command echo and outputs accepted packets with the source and destination code swapped for transmission back to the control center.

Note: See Appendix B for a detailed overview of the LEO-T CDH format.

Figure 4-139 shows the LEO-T CDH Receiver Module.



Figure 4-139: LEO-T CDH Receiver Module

Figure 4-140 shows the Config LEO-T CDH Receiver Module window.

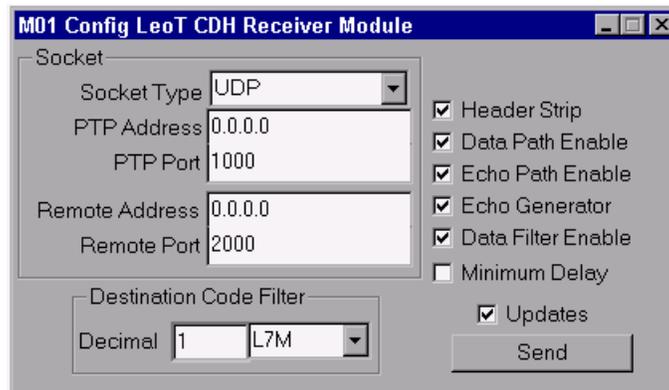


Figure 4-140: LEO-T CDH Receiver Config Window

Table 4-144 through Table 4-146 define the Config LEO-T CDH Receiver Module window parameters.

Table 4-144: LEO-T CDH Receiver Socket Config Parameters

Parameter	Description
Socket Type	Sets the socket type: Disabled, UDP, TCP Client, or TCP Server
PTP Address	Sets IP address (0.0.0.0 specifies default host address)
PTP Port	Socket number

Parameter	Description
Remote Address	Remote IP address
Remote Port	Remote Socket number

Table 4-145: LEO-T CDH Receiver Destination Code Config Parameters

Parameter	Description
Destination Code Filter	Choice of L7M, SFS, ASK, SPZ, WSC, WIL, VLF, VAF, WFF, Disabled, or Other (with value specified in decimal)

Table 4-146: LEO-T CDH Receiver Config Parameters

Parameter	Description
Header Strip	Enable – strips header and passes data; Disable – passes header and data
Data Path Enable	Opens/closes Data Port 1 (data path)
Echo Path Enable	Opens/closes Data Port 2 (echo path)
Echo Generator	Enable – swaps destination and source code in header and sends command echo
Data Filter Enable	Enable – passes only command telemetry out Data Port 1; Disable – passes command and telemetry out Data Port 1
Minimum Delay	Disables TCP nagle algorithm and sets socket priority to highest

Figure 4-141 shows the LEO-T CDH Receiver Module Status window.

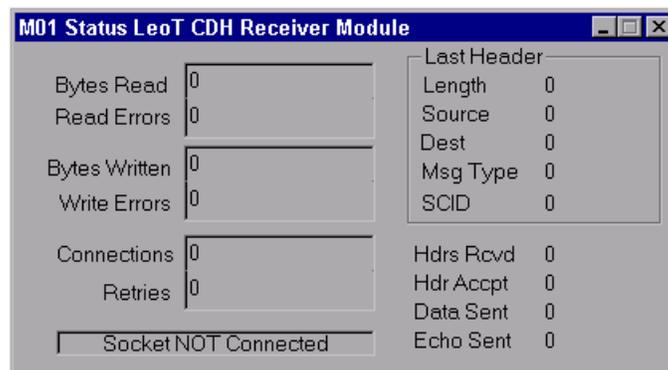


Figure 4-141: LEO-T CDH Receiver Status Window

Table 4-147 and Table 4-148 defines the LEO-T CDH Receiver Module Status window parameters.

Table 4-147: LEO-T CDH Receiver Status Parameters

Parameter	Description
Bytes Read	Count of bytes read
Read Errors	Count of read errors
Bytes Written	Count of bytes written
Write Errors	Count of write errors
Connections	Count of connections
Retries	Count of retried connections
Hdrs Rcvd	Count of headers received
Hdr Acpt	Count of headers accepted
Data Sent	Count of data units sent
Echo Sent	Count of echoes sent

Table 4-148: LEO-T CDH Receiver Last Header Status Parameters

Parameter	Description
Length	Last received header data length

Section 45: LEO-T TFDH Transmitter Module

The LEO-T TFDH (Telemetry Frame Delivery Header) Transmitter Module encapsulates data for transmission across the network. The LEO-T TFDH Transmitter accepts data buffers from other modules and prepends the TFDH header. The LEO-T TFDH Formatter outputs the TFDH packet (header and data).

The LEO-T TFDH Transmitter typically receives telemetry frames with quality annotation from the Serial Input Module or the VCP Module.

Note: See Appendix B for a detailed overview of the LEO-T TFDH format.

Figure 4-142 shows the LEO-T TFDH Transmitter Module window.



Figure 4-142: LEO-T TFDH Transmitter Module

Figure 4-143 shows the Config LEO-T TFDH Transmitter Module window.

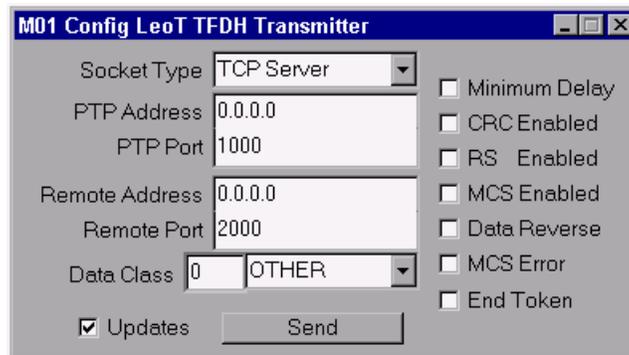


Figure 4-143: LEO-T TFDH Transmitter Config Window

Table 4-149 defines the Config LEO-T TFDU Transmitter Module window parameters.

Table 4-149: LEO-T TFDH Transmitter Config Parameters

Parameter	Description
Socket Type	Sets the socket type: Disabled, UDP, TCP Client, or TCP Server
PTP Address	Sets IP address (0.0.0.0 specifies default host address)
PTP Port	Socket number
Remote Address	Remote IP address

Parameter	Description
Remote Port	Remote Socket number
Data Class	Data Class; 1 = CCSDS Frame, 2 = CCSDS Packet, 3 = TDM Frame, 4 = Stripped TDM Frame
Minimum Delay	Disables TCP nagle algorithm and sets socket priority to highest
CRC Enabled	Enable/Disable; Enabled – module adds CRC state to header
RS Enabled	Enable/Disable; Enabled – module adds Reed-Solomon state to header
MCS Enabled	Enable/Disable; Enabled – module adds Master Channel Sequence check state to header
Data Reverse	Disable – data forward, Enable – data reverse
MCS Error	Enable – MCS error number increases by 2 or more, Disable – number increases monotonically.
End Token	Enable/Disable; Enabled – prior to closing socket, module will send 10 byte SMEX header without frame to destination.

Figure 4-144 shows the LEO-T TFDH Transmitter Module Status window.

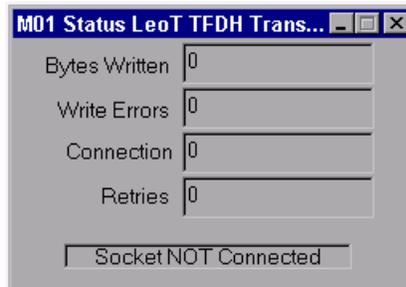


Figure 4-144: LEO-T TFDH Transmitter Status Window

Table 4-150 defines the LEO-T TFDH Transmitter Module Status window parameters.

Table 4-150: LEO-T TFDH Transmitter Status Parameters

Parameter	Description
Bytes Written	Count of bytes written
Write Errors	Count of write errors
Connection	Count of connections
Retries	Count of retried connections

Section 46: NASCOM Blocker Module

The NASCOM Blocker Module accepts a serial telemetry data stream and outputs NASCOM 4800 bit blocks. The NASCOM Block formats consist of multiple block types, each with its own unique header and data length. From the NASCOM Blocker Module, the user can select which Block Type is to be used and provide the information to be placed in the 4800BB header.

Note: NASA Document STDN-502.5 provides a definition of each NASCOM Block type.

Figure 4-145 shows the NASCOM Blocker Module.



Figure 4-145: NASCOM Blocker Module

Figure 4-146 shows the NASCOM Blocker Configuration window.

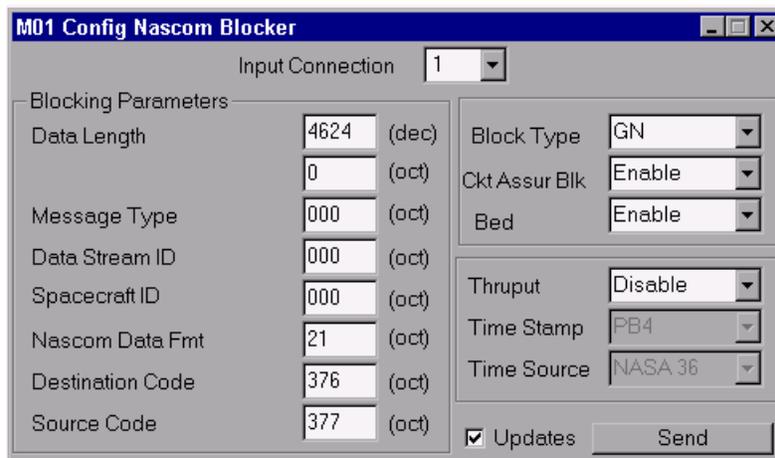


Figure 4-146: NASCOM Blocker Config Window

Table 4-151 through Table 4-154 define the parameters available in the NASCOM Blocker Configuration window.

Table 4-151: NASCOM Blocker Config Input Parameters

Parameter	Description
Input Connection	Specifies which Module (by number) is the data stream source

Table 4-152: NASCOM Blocker Config Blocking Parameters

Parameter	Description
Data Length	Length of data to be inserted into 4800BB. Default value is maximum data length for the selected Block Type; Required by all Block Types.
Message Type	Defines message type; Required by GN, DGIB, JSC, Shuttle, and Delta Block Types.
Message Subtype	Secondary Message Type field for Delta Block Type.
Message ID	Message identifier field for Delta Block Type.
Data Stream ID	Data Stream Identification; Required by GN, and DGIB Block Types.
Spacecraft ID	Defines Spacecraft ID; Required by GN, DGIB, and Delta Block Types.
Nascom Data Fmt	Defines NASCOM Data Format; Required by GN, DGIB, JSC, Shuttle, and Delta Block Types
Destination Code	Destination Code Field; Required by GN, DGIB, JSC, and Delta Block Types.
Destination Code 1	Defines primary destination code for Shuttle Block Type
Destination Code 2	Defines secondary destination code for Shuttle Block Type
Source Code	Source Code Field; Required by all Block Types.
Fixed Value	Fixed Value field for MDM Block Type
PDI Format	Payload Data Interleaver Format field for JSC Block Type
Payload Vehicle ID	Payload Vehicle identifier for JSC Block Type
Source Ckt ID	Source Circuit identifier for Shuttle Block Type

Table 4-153: NASCOM Blocker Config Block Type Parameters

Parameter	Description
Block Type	Defines which NASCOM Block Type format is to be used. Available Block Types are: GN – NASA Ground Network or DDPS2 or DDPS Thruput format; MDM – Johnson Space Center Mux-Demux or Shuttle Thruput format; DGIB – DSN to Ground Network Interface Block format; JSC – Johnson Space Center Telemetry Block format; Shuttle – Space Shuttle format; Delta – Modified DDPS Synchronous Blocks for Delta rocket.
Ckt Assur Blk	Circuit Assurance Blocks. When enabled, ensures that 1 message/sec is transmitted; if there is no data to send, transmits empty block
Bed	Block Error Detector – when enabled, generates PEP code and appends to end of NASCOM block.

Table 4-154: NASCOM Blocker Config Time Parameters

Parameter	Description
Thruput	When enabled, ensures that each block's data field is full based on user-defined data length before transmitting block. NASCOM Blocker will hold block until it is full, and then transmit.
Time Stamp	Appends PB4 Time Stamp to each block.
Time Source	Defines Time Source.

Figure 4-147 shows the NASCOM Blocker Status window.

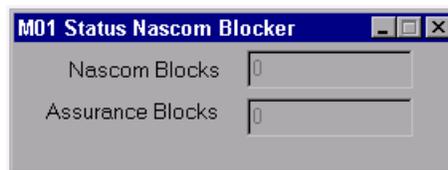
**Figure 4-147: NASCOM Blocker Status Window**

Table 4-156 defines the parameters in the NASCOM Blocker Status window.

Table 4-155: NASCOM Blocker Status Parameters

Parameter	Description
Nascom Blocks	Displays number of NASCOM Blocks transmitted.
Assurance Blocks	If Ckt Assur Blk is enabled, displays number of Assurance Blocks transmitted.

Section 47: NASCOM Deblocker Module

The NASCOM Deblocker Module accepts NASCOM 4800BB based on user-defined Block Header Filter and outputs a serial telemetry data stream.

Note: NASA Document STDN-502.5 provides a definition of each NASCOM Block type.

Figure 4-148 shows the NASCOM Deblocker Module window.



Figure 4-148: NASCOM Deblocker Module

Figure 4-149 shows the NASCOM Deblocker Configuration window.

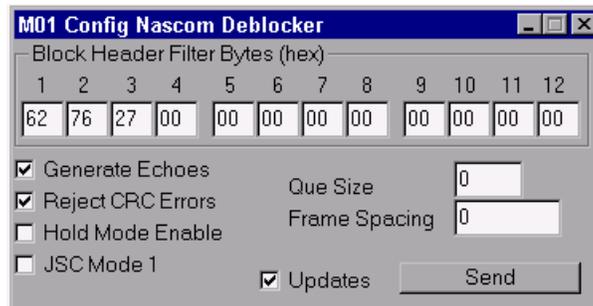


Figure 4-149: NASCOM Deblocker Config Window

Table 4-156 defines the NASCOM Deblocker Configuration window parameters.

Table 4-156: NASCOM Deblocker Config Parameters

Parameter	Description
Block Header Filter Bytes 1-12	Input Block Header of NASCOM Blocks to accepts (up to first 12 bytes)
Generate Echoes	Enabled – Generates and transmits command echo.
Reject CRC Errors	Enabled – Rejects blocks with CRC errors.
Hold Mode Enable	
JSC Mode 1	Enabled – Provides support for JSC Mode 1 Commanding. JSC Mode 1 Command Block header contains extra bytes (defined size); when JSC Mode 1 is enabled, the Deblocker Module grabs these extra bytes.
Que Size	Minimum number of buffers that Rate Adjust Module will

	try to maintain in queue
Frame Spacing	Minimum number of bytes (frames) to inject between command frames

Figure 4-150 shows the NASCOM Deblocker Module Status window.

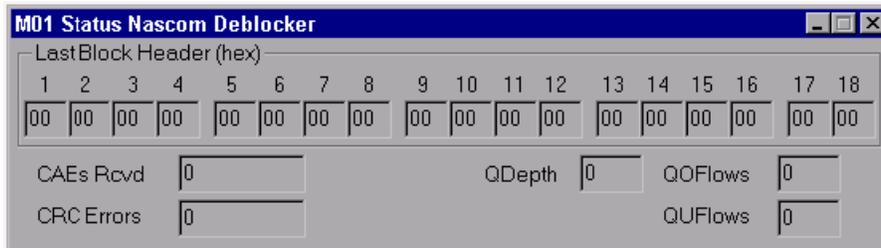


Figure 4-150: NASCOM Deblocker Status Window

Table 4-157 defines the NASCOM Deblocker Module Status window parameters.

Table 4-157: NASCOM Deblocker Status Parameters

Parameter	Description
Last Block Header 1-18	Displays first 18 bytes of the header of the last block received.
CABs Rcvd	Displays number of Circuit Assurance Blocks received.
CRC Errors	Displays number of CRC Errors received.
Qdepth	Displays current queue depth
QOFlows	Displays number of queue overflows.
QUFlows	Displays number of queue underflows.

Section 48: NASCOM RTP Formatter Module

The NASCOM RTP Formatter Module accepts a NASCOM 4800 bit block and outputs an IP encapsulated frame. The interface for the NASCOM RTP Formatter Module is similar to the Network Sockets Module. The user inputs a Multicast IP address as the destination of the encapsulated frames. The Module then transmits all encapsulated 4800BB to that Multicast address.

Note: See Appendix B for a detailed overview of the NASCOM RTP format.

Figure 4-151 shows the NASCOM RTP Formatter Module window.



Figure 4-151: RTP Format Window

Figure 4-152 shows the Config NASCOM RTP Formatter window.

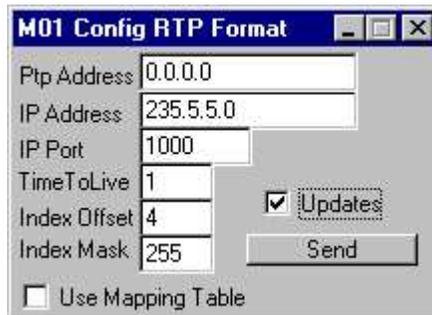


Figure 4-152: Config RTP Format Window

Table 4-158 defines Config NASCOM RTP Formatter window parameters.

Table 4-158: Config RTP Format Parameters

Parameter	Description
PTP Address	Address of local machine.
IP Address	Destination address of encapsulated 4800 BB. Must be a Multicast IP address.
IP Port	IP Port of receiving machines.
TimeToLive	Number of hops before IP message dies
Index Offset	Defines Destination Address location in header to be used for table mapping
Index Mask	Defines Mask for Index Offset

Use Mapping Table	Enables the use of the MSS Mapping Table defined below
-------------------	--

MSS Destination Code Table

The Message Switching System (MSS) destination code table stored in the file mss_table.dat in the root directory (i.e. c:\)

The MSS Table maps destination code of serial data block from/to MSS user to destination IP address for MSS CDs.

Fields:

Destination Code Table Version Number
 MSS Destination Code
 Destination IP Address
 Destination UDP Port

Description:

Destination Code Taable Version Number:

Contains the version number of the Destination Code Table. The version number corresponds to a baselined data format which must be known by and compatible with the CD and NMS processes responsible for interpreting the file. Version number will be in the form of two decimal digits separated by a period (e.g.: 1.0)

Values: Version numbers for the Destination Code Tables will be assigned and managed by Nascom

Destination Code:

Contains a valid MSS supported destination code.

Contains a valid MSS supported destination code.

Values: 1 to 377 octal. Destination codes are assigned and managed by Nascom

Destination IP Address:

Contains the multicast address of either a single host or group of hosts to which the data is destined. The format is the standard dotted decimal notation for IP addresses comprised of four decimal integers separated by decimal points, where each integer gives the value o one octet of the IP address.

Values: Destination IP addresses ae assigned and managed by Nascom

Destination UDP Port:

Contains the UDP port number that the data is to be sent to on the receiving host specified by the Destination IP Address. Possible values for AUDP port numbers can range from 1025 to 65,535 decimal.

Values: All Destination UDP Port numbers in the MSS Destination Code Table use 8001.

Table 4-159: MSS Destination Table Example

```
// MSS Front End Destination Code Table
//
// Version number
version 1.0
//
// Dest. Code      IP Address      UDP Port
      1            225.0.0.1      8001 // MILA Tracking
      2            225.0.0.2      8001 // MIL-71
      3            225.0.0.3      8001 // MIL-71 DSS-72
      4            225.0.0.4      8001 // Bermuda Tracking
      :
      :
      376          225.0.0.254    8001 // JPL-TOPEX
      377          225.0.0.255    8001 // WSC/ETGT
//
// end MSS Front End Dest Code Table
```

Figure 4-148 shows the NASCOM RTP Formatter Module Status window.

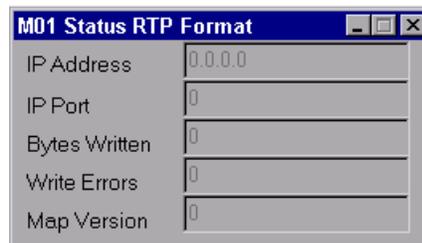


Figure 4-153: RTP Formatter Status Window

Table 4-160 defines the NASCOM RTP Formatter Module Status window parameters

Table 4-160: RTP Formatter Status Parameters

Parameter	Description
IP Address	Displays receiver's IP Address
IP Port	Displays receiver's IP Port
Bytes Written	Count of bytes written
Write Errors	Count of write errors
Map Version	

Section 49: NASCOM RTP Receiver Module

The NASCOM RTP Receiver Module accepts IP encapsulated 4800BB from a network interface and outputs 4800BB. This Module can accept up to 16 different network streams and outputs each 4800BB stream to an independent output port.

Note: See Appendix B for a detailed overview of the NASCOM RTP format.

Figure 4-154 shows the NASCOM RTP Receiver Module window.



Figure 4-154: RTP Receiver Module

Figure 4-155 shows the Config NASCOM RTP Receiver window.

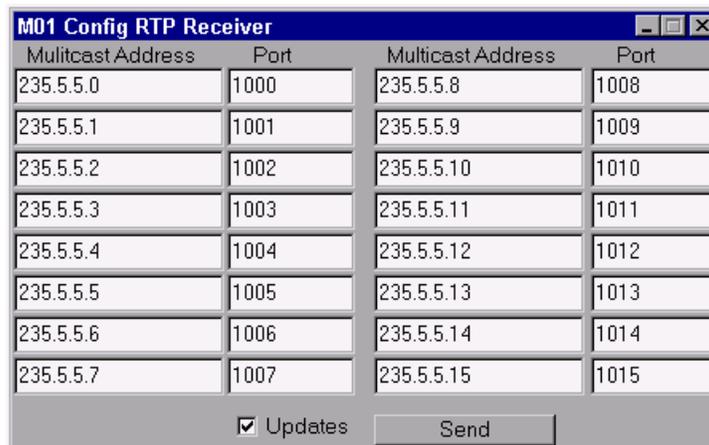


Figure 4-155: RTP Receiver Config Window

Table 4-161 defines the Config NASCOM RTP Receiver window parameters.

Table 4-161: RTP Receiver Config Parameters

Parameter	Description
Multicast Address	Input Multicast addresses to receive data streams
Port	Input Port for each Multicast address

Figure 4-156 shows the NASCOM RTP Receiver Status window.

MulticastAddress / Port	Bytes Read	Seq Errors
235.5.5.0 / 1000	0	0
235.5.5.1 / 1001	0	0
235.5.5.2 / 1002	0	0
235.5.5.3 / 1003	0	0
235.5.5.4 / 1004	0	0
235.5.5.5 / 1005	0	0
235.5.5.6 / 1006	0	0
235.5.5.7 / 1007	0	0
235.5.5.8 / 1008	0	0
235.5.5.9 / 1009	0	0
235.5.5.10 / 1010	0	0
235.5.5.11 / 1011	0	0
235.5.5.12 / 1012	0	0
235.5.5.13 / 1013	0	0
235.5.5.14 / 1014	0	0
235.5.5.15 / 1015	0	0

Figure 4-156: RTP Receiver Status Window

Table 4-162 defines the NASCOM RTP Receiver Status window parameters.

Table 4-162: RTP Receiver Status Parameters

Parameter	Description
Bytes Read	Count of bytes read per input channel
Seq Errors	Count of sequence errors per input channel

Chapter 5

PRACTICAL EXAMPLES

Section 1 - Introduction

This chapter contains examples designed to teach the reader the uses of the Programmable Telemetry Processor for Windows NT (PTP NT). Section 2 - Building Desktops, will cover creating, configuring, and testing desktops. Section 3 covers creating and running Script Files from the PTP NT Server window. These Script Files allow users to bypass the Console GUI and configure desktops by using the commands outlined in Appendix A.

Section 2 - Building Desktops

A. BERT Loopback Desktop

In this first example, we will see a simple application of how the PTP can be incorporated into the initial testing stages of system development as a Bit Error Rate Tester. We are going to loop three modules together in this exercise - Serial Input, Serial Output, and BERT. The BERT module has two portions - a Transmitter and a Receiver. Using the BERT Transmitter, we will generate and send a fixed pseudo-random pattern and send it to the Serial Output Module. The Serial Output module will take the data and transmit it to the Serial Input Module, which will loop it back to the BERT module. The BERT Receiver will then check the data stream for a pseudo-random pattern and attempt to lock on to it, while the Transmitter continues to buffer out a pseudo-random data stream. We will be using a single board (ATHSIO2 or MONARCH) as both our input and our output device.

Now, we have to set-up a BERT Loopback desktop. From the PTP Console window, click on the *Remote Desktop* menu, select *Add Module* and load the following modules:

1. M0202_AvtecSerialOutput.dll - Avtec Serial Output Module
2. M0201_AvtecSerialInput.dll - Avtec Serial Input Module
3. M1001_BitErrorRateTester.dll - BERT Module

As you load a module, a GUI box with several push-buttons representing the module will be loaded onto the Console and assigned a module number (M01 represents module 1), see **Figure 5-1**. After loading, we will need to configure the modules as follows:

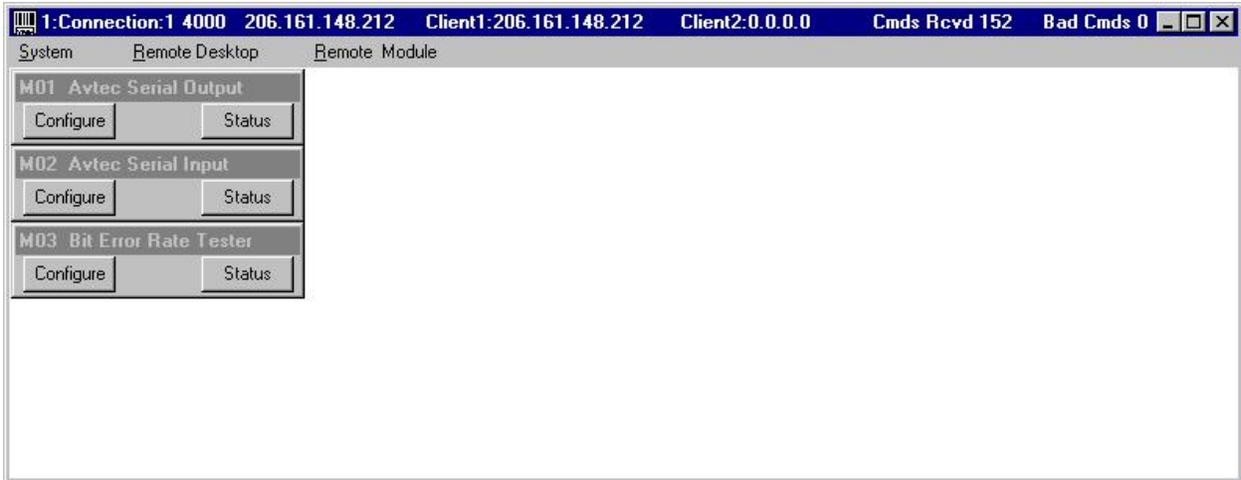


Figure 5-1:

Serial Output Module Configuration

Click on the *Configure* button to display the configuration window for this module. Configure the Output Module as displayed in Figure 1. The board parameter in the upper right corner of the configuration box tells the PTP which Serial I/O board (ATHSIO2 or MONARCH) should be used as the output module¹. In **Figure 5-2**, board 2 is an ATHSIO2 Frame Synchronizer. Complete the configuration of this module by setting it to output NRZ-L data of length 1264 bytes at a rate of 100 kHz.

When you click the mouse pointer in a box in the configuration window, you should notice the check next to the *Update* box disappear. When you are done making changes in the configuration window, you must click the *Send* button for the Console application to send the changes to the PTP NT Server application.

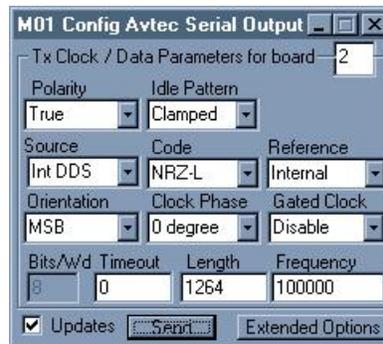


Figure 5-2: Avtec Serial Output Module Configuration

Serial Input Module Configuration

¹ Note: Selecting 0 (zero) for the board number refers to no board. All valid board (channel) numbers start with 1. Check your system configuration to determine the channel number assignment for each board. This applies to both the Serial Output and Serial Input modules.

Click on the *Configure* button to display the configuration window. Set the options so that it is identical to **Figure 5-3**. Here, the module is being configured to receive an NRZ-L data stream with a *Frame Length* of 1264 (which is what is being sent by the output module), again using board 2. Setting the *Source* to Loopback tells the Input device that the data stream will be coming from the Output of the same device. In addition, the *Synchronizer* must be disabled if the board is being used as a BERT. This is because a BERT stream does not contain a Sync pattern to lock onto. Disabling the Synchronizer tells the Input device to just take in the data stream. Click the *Send* button to have the PTP acknowledge the changes.

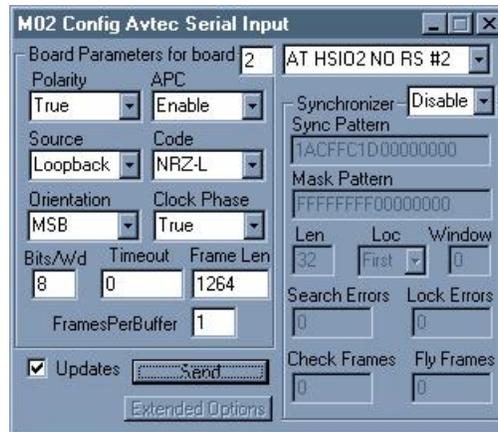


Figure 5-3: Avtec Serial Input Module Configuration

No configuration will be required for the BERT module at this time. We will discuss this module further on in this example.

Now that the necessary modules have been configured, we will need to make the appropriate connections between all the modules. Since we are using the same card as the input and output interface, the connection between the serial output and serial input already exists via hardware. In addition, software connections will be needed.

Select *Edit Module Connections* from the *Remote Desktop* menu. A window will open with two sets of checkbox grids - one for *Module DATA Outputs* and one for *Module EVENT Outputs* (**Figure 5-4**). The column numbers across the top of the two grids represent the module number you wish to connect to from the current module selected. Data Ports 1 through 4 represent different data channel outputs for the module selected. Event Port represents an event connection from the current module.²

Serial Output Module Connections

Make an *Event* connection to the **BERT Module** from the **Serial Output Module** by placing a check in the box in column 3 in the row *Event Port* (column 3 represents module 3, the BERT module), as shown in **Figure 5-4**.

² Please refer to Chapter 3, Section 1, *Connecting Modules* for a more detailed definition of Event and Data Connections.

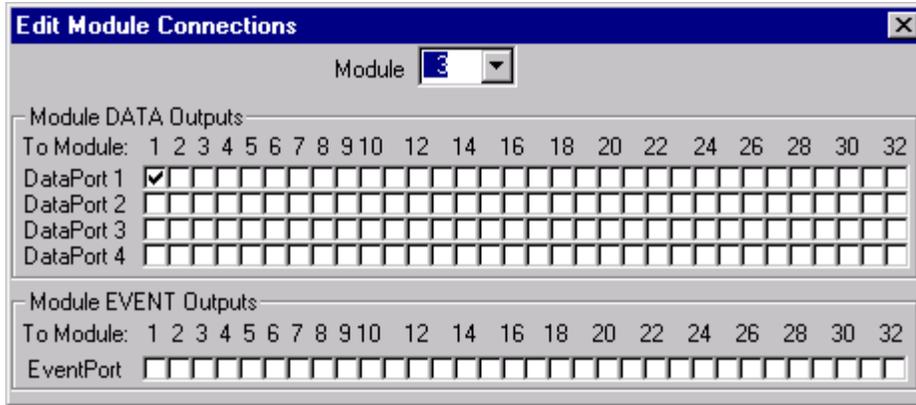


Figure 5-6: BERT Module Connections

Now that all the individual module connections have been made, we can close the *Edit Module Connections* window.

The desktop is now ready to be tested. To run the desktop, from the *Remote Desktop* menu, select *Enable All Streams...*, or, you can use the key combination <CTRL>+F1. To check the status of each module, simply click on the *Status* button in the main dialog box for that module.

Open the Status window for the Serial Output module. It should be similar to the one shown in **Figure 5-7**. This window displays information including the transmit state, number of bit slips, number of frames sent and the IRQ count. For more details and definitions on the various fields, see Chapter 4, Section 3 of this User's Manual.



Figure 5-7: Avtec Serial Output Status

Open the Status window for the Serial Input module. It should be similar to the one in **Figure 5-8**. This window displays information including the board and clock state, number of frames received and correctable and uncorrectable errors. For more details, see Chapter 4, Section 2 of this User's Manual.

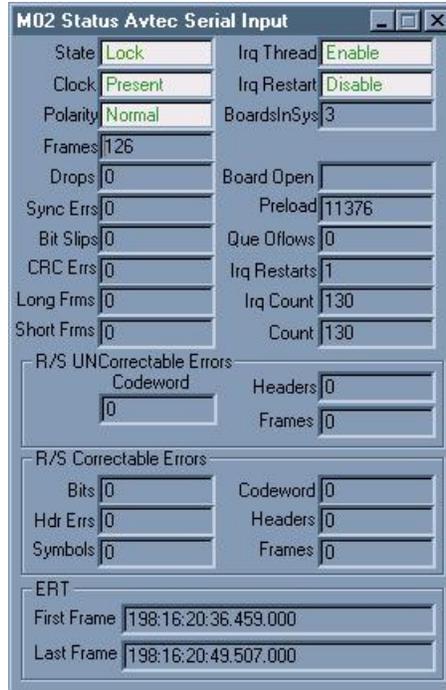


Figure 5-8: Avtec Serial Input Status

The BERT Status window, as shown in **Figure 5-9**, displays information for both the Transmitter and the Receiver functions of the BERT module. The Transmitter status displays the number of errors that have been injected by the user. The Receiver status displays frame data and bit error data. For more details on this module, see Chapter 4, Section 5 of this User's Manual.

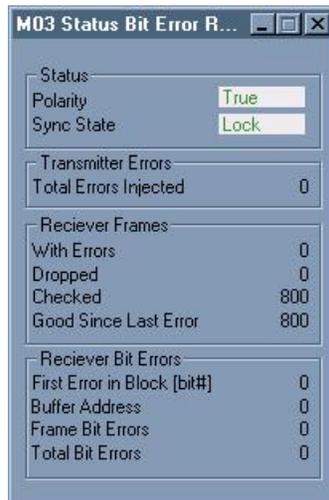


Figure 5-9: BERT Status

Open up the configuration window for the BERT module and place it beside the BERT status window. You will see two checkboxes allowing you to select/deselect the type of errors you would like inserted into the data - either a single bit error or a continuous stream of errors. Now,

insert a single error by clicking the checkbox beside the work *Single*, and you will see the *Total Errors Injected* count and the *Receiver Frames with Errors* count both increase in the status window. If the error caused a frame to drop, you will see that counter increase as well. If you select the *Continuous* checkbox, the BERT will send a steady stream of errors into the data buffers, and will continue to do so until the *Continuous* checkbox is disabled. Select the *Continuous* checkbox and see how the system reacts to a sudden rush of errors.

The BERT module in similar loops, i.e. between two separate input and output boards or any other type of interface, can be used to test the BER (Bit Error Rate) of components and systems.

B. Telemetry Acquisition and Processing Desktop

This next example will demonstrate a typical application of the PTP as a Telemetry Receiver. This will require the use of two PTP NT Servers, both of which can be run off one PTP NT machine. One PTP NT Server will be designated as the “*Simulator*”, and will be used for transmitting simulated telemetry. The other PTP NT Server will be designated as the “*Receiver*” and will be used to process and forward the data as necessary. The transfer of telemetry from the Simulator to the Receiver will be accomplished via a network sockets connection. Before going into the construction of the desktops, we will start with an overview of network protocols and introduce the Network Sockets Module.

Using the Network Sockets Module

The Network Sockets module allows data to be sent and received via a network socket connection. The module supports UDP (User Datagram Protocol), Multicasting, TCP (Transmission Control Protocol) Server, and TCP Client protocols.

User Datagram Protocol (UDP) is a connectionless transport protocol. It allows applications to send and receive encapsulated raw IP (Internet Protocol) data without having to establish a connection. This protocol is widely used when quick delivery of data is more important than accurate delivery, as in real-time applications.

IP Multicasting can be used when you want a specific data stream to be directed to multiple users on a specific network group. Multicasting is a subset of the UDP protocol. This means that it is a connectionless transport protocol and does not attempt to establish a connection. It simply sends data to a Multicast IP address. Multicast addresses are of the class D IP address format. Class D addresses fall in the range 224.0.0.0 through 239.255.255.255. Each class D address identifies a group of hosts and 28 bits are assigned for identifying groups. This means that over 250 million groups can exist at any one time. When a packet is sent to a multicast address, a best-efforts attempt is made to deliver that packet to all the members of the group that was addressed. However, there are no guarantees on delivery, thus some members of the group may not receive the packet.

Transmission Control Protocol (TCP) is a connection-oriented transport protocol. This means that a handshake, or an acknowledgment, takes place between the transmitter (server) and the receiver (client). It is designed to provide a reliable byte stream over unreliable networks and internetworks. It fragments an incoming data stream into discrete messages and sends them to a specified location. At the destination location, the receiving TCP process reassembles the received messages into its original sequence. The TCP process also handles flow control to prevent a fast sender from overloading a slow receiver with more messages than it can handle.

Configuring the Network Sockets module appropriately depends on whether the module will be used as a transmitter or a receiver and the *Socket Type* selected in the configuration window. After these requirements are determined, we can configure the *PTP Address*, *PTP Port*, *Remote Address*, *Remote Port*, and *TCP Rx Length*. The *TCP Rx Length* is usually set to the frame length.

First, we will start with configuring the Sockets module for transmitting via a TCP connection. The first thing to do is to set the *Socket Type* to *TCP Server*. The only other items that need to be configured are *PTP Address* and *PTP Port*. The default PTP Address is 0.0.0.0. If there is only one Ethernet board installed in your computer, this value does not need to be changed because an address of 0.0.0.0 refers to any available output address. This also holds true for multiple Ethernet boards within a computer. Each Ethernet board is assigned its own IP address, and if the PTP address is set to 0.0.0.0, the data will be sent out any available connection. However, if multiple Ethernet boards exist and only one is to be assigned for outputting a data stream, then setting the PTP address to that board's IP address will tell the PTP to send the specified data stream through that particular board. In addition to the PTP Address, the PTP Port must also be defined. In general, the PTP Port is used to define a virtual input/output channel on a local machine. The necessity of this can be seen when multiple Network Sockets modules are being used to transmit multiple data streams to different locations. Each Sockets module would setup an individual Port address. Thus, each Sockets module can use the same PTP Address because the Port Address would differentiate the streams. In addition, when a TCP connection is made, a sender and a receiver create end points called sockets. A socket consists of a host's IP address and a port number, which is local to the host. The port is necessary for TCP to make a connection. The default port address can be used, or any other address you wish to assign. The Remote Address and Remote Port are not defined for TCP Server. When the Network Sockets module is set as a TCP Server, the module is waiting for a TCP Client on a remote machine to attempt a connection and request information, it does not make the attempt itself.

This leads us into the next possible configuration for the Network Sockets module, receiving via a TCP connection. The first thing to do is to set the *Socket Type* to *TCP Client*. For the TCP Client, the only items that need to be configured are the *Remote Address* and the *Remote Port*. These values are the same as the *PTP Address* (the actual IP address, not 0.0.0.0) and *PTP Port* of the TCP Server Network Sockets module of the computer that the data stream will be coming from. The TCP Client will try to make a connection with the specified TCP Server, and once the connection is established, the Server will start sending data to the Client. Unless a system configuration has multiple Ethernet cards installed and a data stream needs to come in through a dedicated IP address, the *PTP Address* and *PTP Port* do not need to be changed from their

default values. Basically, the TCP Client is set up to ask a TCP Server to send information, while the TCP Server waits for such a request to be made and sends the data to the requesting TCP Client.

Next, we will go into configuring the Network Sockets module for transmitting via UDP. Again, the first thing to do is to set the *Socket Type* in the configuration window to *UDP*. Since UDP is a connectionless protocol, the only things that need to be configured are the *Remote Address* and *Remote Port*. These addresses must match the *PTP Address* and *PTP Port* of the remote system that the data stream is being directed to. Although UDP does not require a socket connection as in TCP, the Remote Port address is still necessary for the Network Sockets module to send the data stream to the appropriate channel within the remote system. The port is not used to establish a connection, merely to direct the data stream once it reaches the remote host.

On the receiving side of the UDP transmission, the Network Sockets module must also have the *Socket Type* set to *UDP* in the configuration window. The only other items that need to be configured on this end are the *PTP Address* and the *PTP Port*, which are the local address and local port. Again, since UDP is connectionless, the *Remote Address* and *Remote Port* do not need to be adjusted. The module configured in this fashion will receive any data stream that is addressed to it without wondering where it came from.

Configuring the Networks Sockets module to transmit and receive via Multicasting is similar to transmitting and receiving via UDP. There are a couple differences, however. First, the *Socket Type* for both transmitting and receiving in Multicast must be set to *Multicast*. Second, on the transmitting side, the *Remote Address* should be set to the class D Multicast IP address that represents the group of hosts that the data stream will be sent to. On the receiver side, the *Remote Address* (local) still remains as the actual IP address of the receiving machine, and the *PTP Address* must be Multicast address of the group of hosts.

System Setup - Simulator and Receiver Desktops

The first step is to load the appropriate PTP NT software on each system.³ The *Simulator* will only require a PTP NT Server to be loaded. Load a Server window by double clicking on, or executing the file, PTP NT.EXE in the PTP NT directory. The *Receiver* will also require a PTP NT Server, as well as two PTP NT Consoles. Multiple Consoles can be loaded by executing the command CONSOLE.EXE in the PTP NT directory more than once. Load one Server and two Console windows on the *Receiver*.

The reason for two Console windows is so that both PTP NT Servers (*Simulator* and *Recevier*) can be controlled from one system (in this example, the *Receiver*). Each Console will be connected, via network sockets, to a Server. After loading the Server and Console windows, connect one Console to the PTP NT Server running on the *Simulator* and the other Console to the Server on the local *Receiver*.

³ For this example, it is assumed that 2 PTP NT machines are being used, each running one instance of the PTP NT Server application (PTP NT.EXE). One machine is designated the *Simulator* and the other is designated the *Receiver*.

Building the Simulator Desktop

The *Simulator* Desktop will simulate a telemetry data stream and transmit it via a network connection. To accomplish this, we will be using CCSDS Virtual Channel Simulator module in conjunction with the CPU Timer module, the IPDU (Inter-project Data Unit) Formatter module, and the Network Sockets module.

As the name implies, the Virtual Channel Simulator generates and transmits simulated Virtual Channel Data Units (VCDU). The VC Simulator is mainly designed to be used for system testing purposes, and as such, does not generate the maximum 64 channels that the Virtual Channel Processor is designed to manage. There are four different CCSDS versions that the VC Simulator can be configured to use. *Version 1* transmits virtual channels 0, 1, 2, and 7 and fills these channels with their respective channel number. *Version 2* transmits virtual channels 0, 1, 2, and 63 and fills these channels with their respective channel number. *Version 1 Fill* only fills virtual channel 7, which is the fill channel, with the letter "A." *Version 2 Fill* only fills virtual channel 63, the fill channel, with the letter "A." The output of the VC Simulator can be directed to modules including the Virtual Channel Processor, Serial Output module, or the Network Sockets module.

The simulated stream will be sent to the IPDU Formatter. This module receives VCID frames and formats them for transfer over the network. For this configuration, the IPDU Formatter accepts 1264 byte frames from the Virtual Channel Simulator and prepends the 32-byte IPDU header.

The IPDU formatted data stream is then sent to the Network Sockets module. The Sockets module will be configured to open a UDP socket and send the formatted telemetry stream to a specified IP address.

The CPU Timer module is also required here to provide Events to the Virtual Channel Simulator. These Events will simulate a clock for the VC Simulator to synchronize to so that it can generate a telemetry stream.

Load the following modules onto the Console connected to the *Simulator*:

1. M1003_CCSDSVirtualChannelSimulator.dll - Virtual Channel Simulator
2. M100E_IPDUFormatter.dll - IPDU Formatter Module
3. M0401_NetworkSockets.dll - Network Sockets Module
4. M100C_CPUSimulator.dll - CPU Timer Module

Configure these modules as follows:

CCSDS Virtual Channel Simulator Configuration

Click on the *Configure* button. Configure the module as shown in **Figure 5-10**. There are several options available in this window for Reed-Solomon encoding. The *Code Size* specifies the length of the Reed-Solomon code block, in bytes, which is appended to the end of the frame. The *Virtual Fill* designates the fill data to be used in the simulated data streams. *Interleave* specifies the interleave depth for the Reed-Solomon encoding, which takes a value from 1 (no interleaving) to 7. When *CRC* is enabled, a 2 byte CRC code is appended immediately before the Reed-Solomon code block. For more details, see Chapter 4, Section 7 of this User's Manual. Click the *Send* button to accept the changes made and close the configuration window.

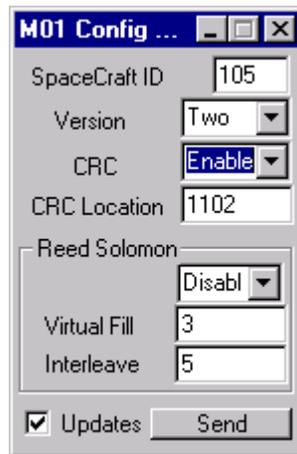


Figure 5-10: Virtual Channel Simulator Module Configuration

IPDU Formatter Module Configuration

Click on the *Configure* button. The parameters that can be selected in this window all comprise information that will be placed in the IPDU header. This information is used by the IPDU Receiver module to filter the data is being input. For this example, set the *Source Code* to *L7M* and the *Destination Code* to *WFF*. Also, the *Message Type* should be set to *Real Time Telemetry*. Once configured, the module should look like **Figure 5-11**. Click *Send* to accept the changes.

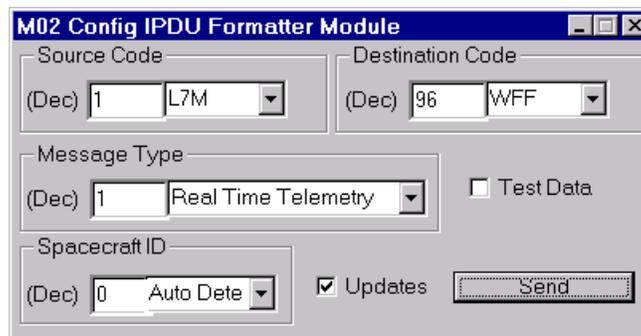


Figure 5-11: IPDU Formatter Module Configuration

Network Sockets Module Configuration

Click on the *Configure* button. Configure the module as shown in **Figure 5-12**. The *Remote Address* and *Remote Port* should be set to the address and port designated for the reader's *Receiver* machine. The *PTP Address* and *PTP Port* can be ignored here. Click *Send* to accept the changes.

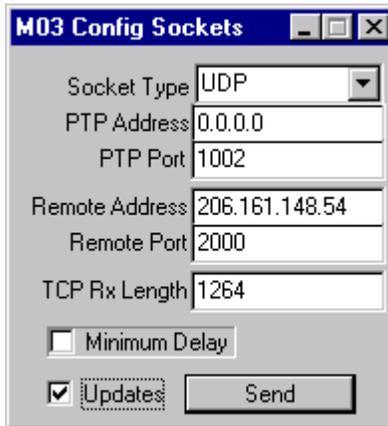


Figure 5-12: Network Sockets Module Configuration

CPU Timer Module Configuration

Click on the *Configure* button. Configure the module as shown in **Figure 5-13**. This module will provide a 100 kHz clock to the VC Simulator module for generation of the telemetry stream. Click *Send* to accept the changes.



Figure 5-13: CPU Timer Module Configuration

Next, the modules need to be connected. Select *Edit Module Connections* from the *Remote Desktop* menu.

CCSDS Virtual Channel Simulator Connections

Make a connection from *DataPort 1* to the IPDU Formatter module as shown in **Figure 5-14**.

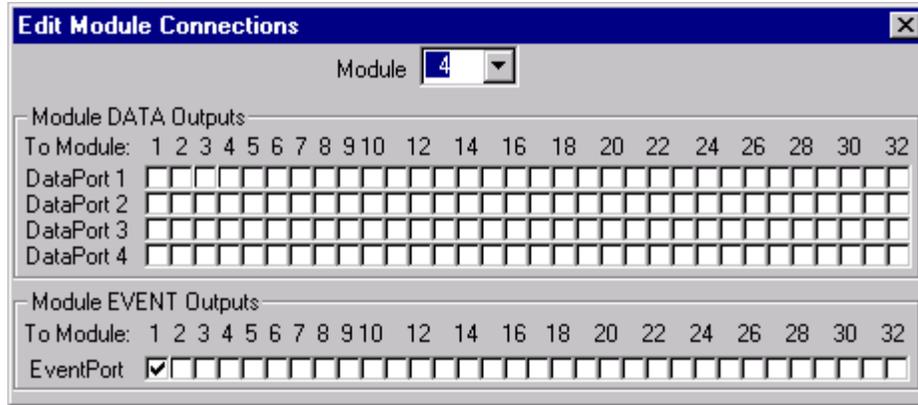


Figure 5-16: CPU Timer Module Connections

Close the *Edit Module Connections* window. The *Simulator* is now set up and configured properly to send a telemetry data stream.

Building the Receiver Desktop

The *Receiver* will be configured to take in the telemetry stream that is being generated by the *Simulator*, via a network socket, and process that stream.

An IPDU Receiver module will be used to take in the telemetry stream. This module has built-in socket functions, and thus a Network Sockets module will not be needed to make the connection to the remote host (*Simulator*).

The data stream that will be entering this system will have four virtual channels interleaved. In this example, we want to filter the individual virtual channels and perform different operations on each. The filtering of the virtual channels is accomplished using the CCSDS Virtual Channel Processor.

The CCSDS Virtual Channel Processor can take in a data stream, filter up to 64 Virtual Channels (as defined by CCSDS Recommendations) contained within the stream, and output each virtual channel to one of 4 output data ports. Configuring the data port connections is done using the *Edit Module Connections* window. The Virtual Channel Processor can also be configured to perform CRC and Reed-Solomon checking.

In this scenario, we want to view virtual channel 0 and virtual channel 1 as they enter our system and log them to one file on the local hard disk. This procedure will require two Null Transceiver modules and one File Recorder module. Furthermore, virtual channel 2 contains data that we do not need to examine immediately, so we are going to log that channel directly to a second file on the local hard disk. A second File Recorder module will be required since we wish this data to be in a separate file than the data from virtual channels 0 and 1. Finally, virtual channel 3 needs to be sent, via Network Sockets Module, to a remote host for further processing.

The *Receiver* desktop will therefore require the following modules:

1. M100D_IPDUReceiver.dll - IPDU Receiver Module
2. M1002_VirtualChannelProcessor.dll - CCSDS Virtual Channel Processor
3. M1000_NullTransciever.dll - Null Transceiver Module
4. M1000_NullTransciever.dll - Null Transceiver Module
5. M0302_FileRecorder.dll - File Recorder Module
6. M0302_FileRecorder.dll - File Recorder Module
7. M0401_NetworkSocketsModule.dll - Network Sockets Module

Again, a GUI box for each module will be displayed on the Console as each module is loaded. The modules will now have to be configured as follows:

IPDU Receiver Module Configuration

Click on the *Configure* button to display the configuration window. Configure the module as shown in **Figure 5-17**. Echo generation is not needed in this example, therefore *Echo Path Enable* and *Echo Generator* can both be disabled. In addition, *Data Filter Enable* must be disabled. When enabled, *Data Filter Enable* allows only data streams labeled as Commands to pass through the module, all other Telemetry is blocked. Since the data entering this module is Real Time Telemetry, the *Data Filter Enable* needs to be disabled. Also, make sure the *PTP Address* and *PTP Port* match IP address and port of the *Simulator*. Finally, the *Destination Code Filter* needs to be set to *WFF*. Click the *Send* button to tell the PTP to accept the changes.

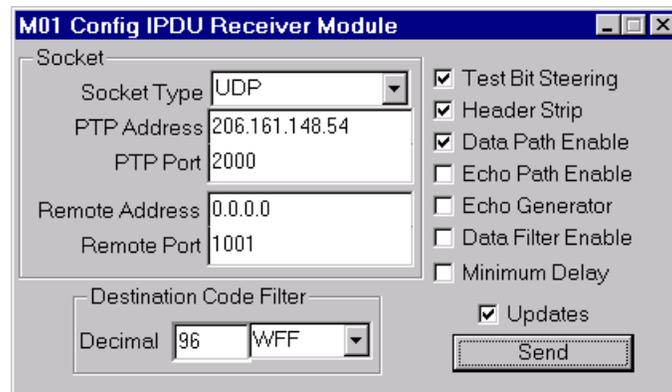


Figure 5-17: IPDU Receiver Module Configuration

CCSDS Virtual Channel Processor Configuration

Click on the *Configure* button to display the configuration window. Configure as shown in **Figure 5-18**.

Click on the *Extended Options* button. A window labeled Virtual Channels will appear with the buttons, *Select All* and *Disable All*, as well as a list of all 64 virtual channels (0-63). An edit box beside each virtual channel directs where that channel goes, i.e. placing a 2 in the edit box beside VC 4 sends the data in VC 4 out data port 2 of the Virtual Channel Processor module. Clicking

on *Select All* will set all the virtual channels to data port 1. Clicking on *Disable All* will set all virtual channels to data port 0, i.e. nowhere.

Click on *Disable All*. Set the channels as they appear in **Figure 5-19**. Go back to the main configuration window and click the *Send* button to accept the changes made. Close the *Extended Options* after clicking *Send* in the Configuration window, otherwise the changes will not be transmitted to the PTP NT Server. Close the Configuration windows.

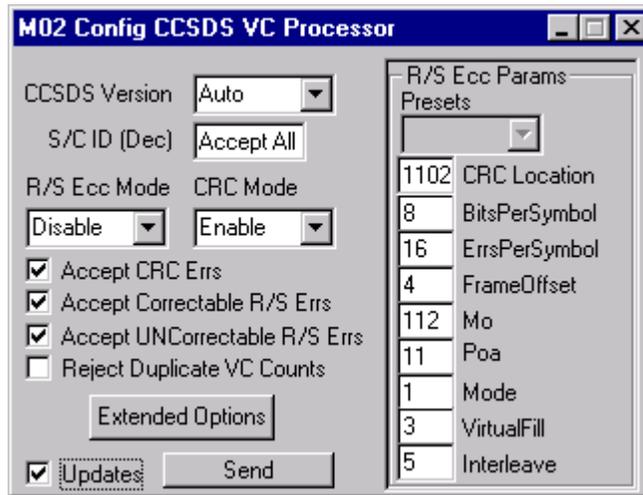


Figure 5-18: Virtual Channel Processor Configuration

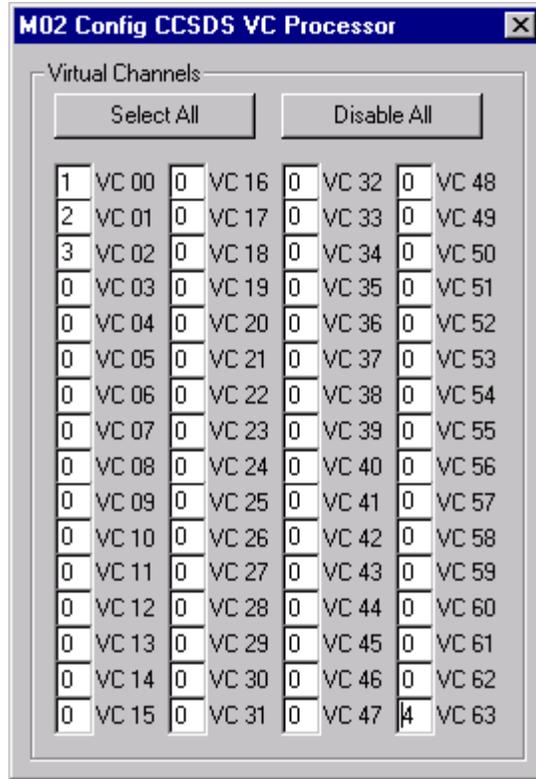


Figure 5-19: VC Processor Extended Options Configuration

Null Transceiver Modules Configuration

Click on the *Configure* button. Configure the Null Transceiver Modules as shown in **Figure 5-20** and **Figure 5-21**. Click the *Send* button to accept the changes.

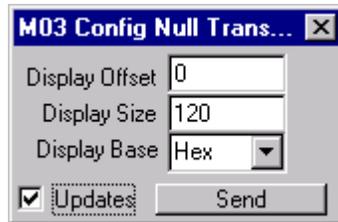


Figure 5-20: Null Transceiver 1 Configuration

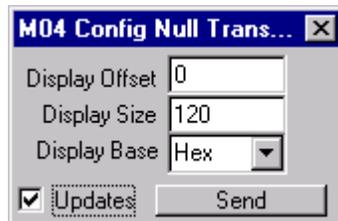


Figure 5-21: Null Transceiver 2 Configuration

File Recorder Modules Configuration

Click on the *Configure* button. The *File Name* defaults to *C:\PTP_USER\LOGS\DEFAULT.RC1* for both File Recorder modules. Configure the modules so that they resemble **Figure 5-22** and **Figure 5-23**. Change the file names so that each module points to a different file. Use *C:\...\DEFAULT.RC1* for the File Recorder that will take in VC00 and VC01 and *C:\...\DEFAULT.RC2* for the other File Recorder⁴. Click *Send* to accept the changes.

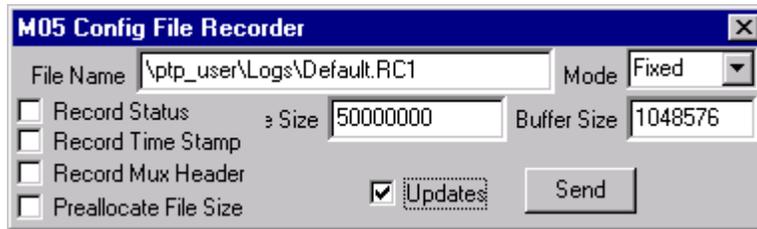


Figure 5-22: File Recorder 1 Configuration



Figure 5-23: File Recorder 2 Configuration

Network Sockets Module Configuration

Click on the *Configure* button. Configure the Network Sockets Module as shown in **Figure 5-24**. The *Remote Address* and *Remote Port* here would be set to the IP address of the host that is to receive the data stream. Click the *Send* button to accept the changes.

⁴ Note: If you wish to have your output file written to a network drive, make sure *Buffer* is set to a non-modulo-4096 number. If the value of the *Buffer* is modulo-4096, the File Recorder module is automatically set to non-buffered mode, which allows maximum write speed to the local disk, but does not allow writing to a network disk.

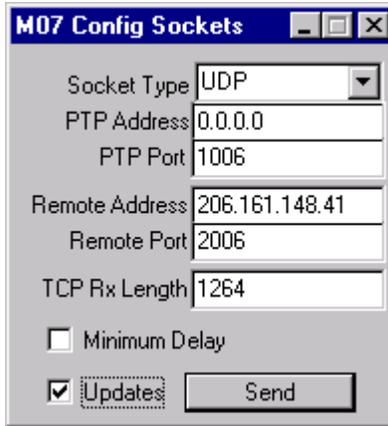


Figure 5-24: Network Sockets Module Configuration

Now that the modules are loaded and configured, we will need to set the connections between them. Select *Edit Module Connections* from the *Remote Desktop* menu.

IPDU Receiver Module Connections

Module 1: Make a connection from *DataPort 1* to the Virtual Channel Processor.

Virtual Channel Processor Module Connections

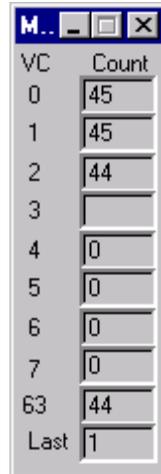
Module 2: Make a connection from *DataPort 1* to the first Null Transceiver module and to the first File Recorder module, from *DataPort 2* to the second Null Transceiver module and the first File Recorder module, from *DataPort 3* to the second File Recorder module, and from *DataPort 4* to the Network Sockets Module.

Now that we have made all the necessary connections between the modules, we can close the *Edit Module Connections* window. This desktop is now ready to receive, process and transmit data.

Testing the Desktops

All that remains is to test the desktop. From both Console windows, pull down the *Remote Desktop* menu and select *Enable All Streams...*, or use the key combination <CTRL>+F1. Data should now start flowing between the various modules and across the Network connection.

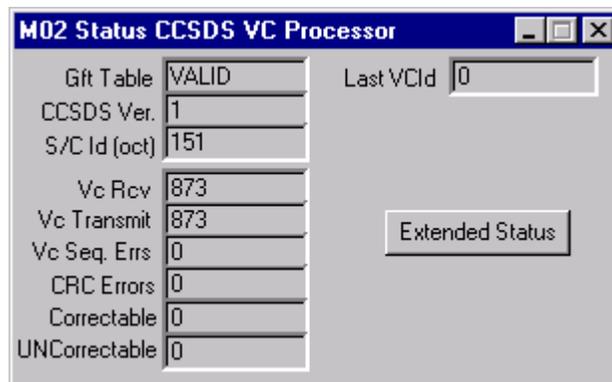
Click on the *Status* button for the Virtual Channel Simulator module on the *Simulator Console*. The status window should be similar to **Figure 5-25**. This window displays the transmitted frame count for each virtual channel (0 through 7 and 63). The display box labeled *Last* displays which of the virtual channels transmitted in the previous set was the last to be transmitted. In other words, of the set of virtual channels 0 through 7 and 63 that were just sent, if 1 appears in the *Last* box (see **Figure 5-24**), virtual channel 1 was the last of the data set sent. This can be useful for determining a slow point in transmission.



VC	Count
0	45
1	45
2	44
3	
4	0
5	0
6	0
7	0
63	44
Last	1

Figure 5-25: Virtual Channel Simulator Status

Open the Status window for the Virtual Channel Processor on the *Receiver Console* (see **Figure 5-26**). This window displays the number of virtual channel frames received (*VcRcv*) and transmitted (*VcTransmit*), the number of sequence errors, the number of CRC errors and how many of these were *Correctable* and *Uncorrectable*. It also displays the Spacecraft ID (*S/C Id*) as defined from the Simulator configuration, and the CCSDS version that is being transmitted. A *0* in the *CCSDS Ver* box refers to *Version 1* or *Version 1 Fill*. A *1* in the *CCSDS Ver* box refers to *Version 2* or *Version 2 Fill*.



M02 Status CCSDS VC Processor	
Gift Table	VALID
CCSDS Ver.	1
S/C Id (oct)	151
Vc Rcv	873
Vc Transmit	873
Vc Seq. Errs	0
CRC Errors	0
Correctable	0
UNCorrectable	0
Last VCId	0
Extended Status	

Figure 5-26: Virtual Channel Processor Status

The Status window for the Null Transceiver modules will display an ASCII text dump of the data passing out of another module and into the Null Transceiver. For our simulated data, the output for virtual channels 0 and 1 will resemble **Figure 5-27** and **Figure 5-28**.

C. File Playback Desktop

In this next example, we are going to demonstrate usage of the File Playback module using one of the data files that was recorded in the previous example. Recall that the file C:\...\DEFAULT.RC1 stored virtual channels 0 and 1 in the previous example. Suppose now that remote host A needs virtual channel 0, and remote host B needs virtual channel 1. Remote hosts A and B both require a UDP transmission. The data contained in virtual channels 0 and 1 is stored in a single file. This means that the virtual channels need to be separated and transmitted individually via Network Sockets modules.

Using the File Playback module with the Virtual Channel Processor module, we can separate the virtual channels and direct each to a Network Sockets module for transmission to their respective remote hosts. Basically, the File Playback will play back the data from the file C:\...\DEFAULT.RC1. This stream will be fed into the Virtual Channel Processor, which can be configured to process and direct each virtual channel out a different data port of the module. Each data port will then be directed to a Network Sockets module, which will transmit the data to a remote host.

To setup our File Playback desktop, we must load the following modules:

1. M0301_FilePlayBack.dll - File Playback Module
2. M1002_VirtualChannelProcessor.dll - Virtual Channel Processor Module
3. M0401_NetworkSockets.dll - Network Sockets Modules
4. M0401_NetworkSockets.dll - Network Sockets Modules
5. M100C_CPUTimer.dll - CPU Timer Module

The CPU Timer module will be used here again to provide time dependent Events to modules. In this example, we will be using the CPU Timer to provide a clock to the File Playback module. This will control how fast the data file is read and transmitted to the next module.

The File Playback module is used to play back a previously recorded file. This can be useful for testing systems as well as for transmitting previously recorded data.

Once loaded, these modules will need to be configured as followed:

File Playback Module Configuration

Click on the *Configure* button. Configure this module as shown in **Figure 5-29**. Selecting *Loop* in the *Mode* sub-menu tells the File Playback module to replay the data file from the start once the end of the file has been reached. Click the *Send* button to accept the changes and close the configuration window.

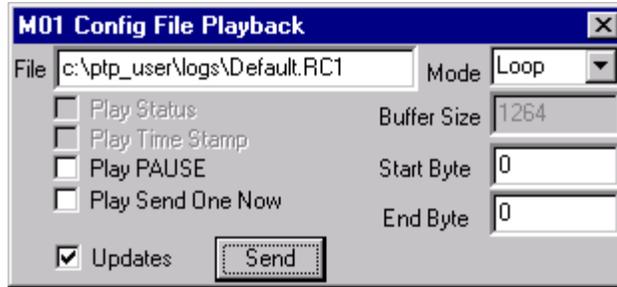


Figure 5-29: File Playback Module Configuration

Virtual Channel Processor Module Configuration

Click on the *Configure* button. Setup up the configure window as shown in **Figure 5-30**. Click on the *Extended Options* button. Click on *Disable All*. Configure the extended configuration options window as shown in **Figure 5-31**. Go back to the main configuration window and click on the *Send* button to accept the changes made. Close the *Extended Options* and the *Configuration* windows.

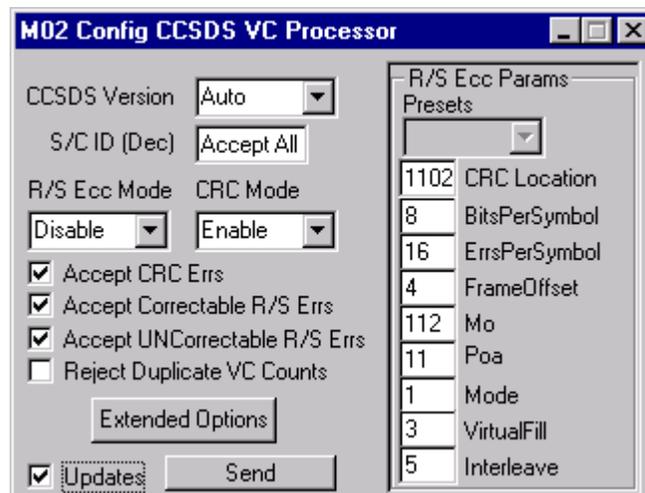


Figure 5-30: Virtual Channel Processor Configuration

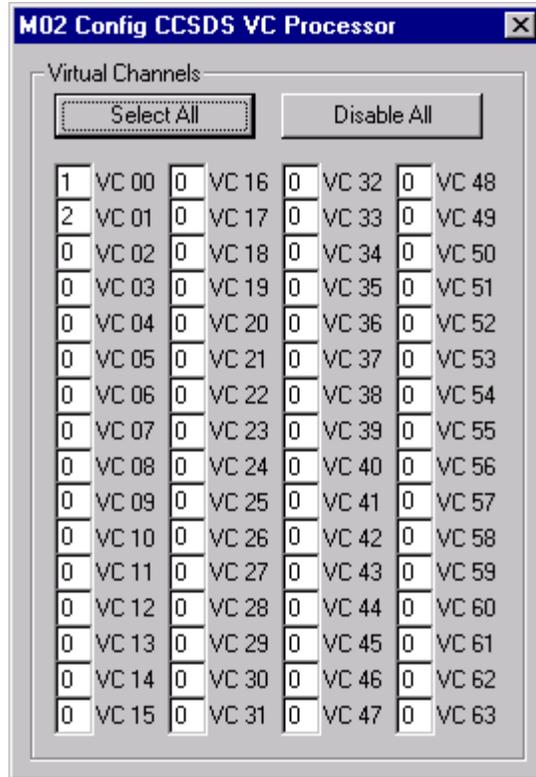


Figure 5-31: VC Processor Extended Options Configuration

Network Sockets Module #1 (Multicast Address A) Configuration

Click on the *Configure* button. Configure the module as shown in **Figure 5-32**. The *Remote Address* has to be a suitable Class D Multicast address. Click *Send* to accept the changes and close the configuration window.

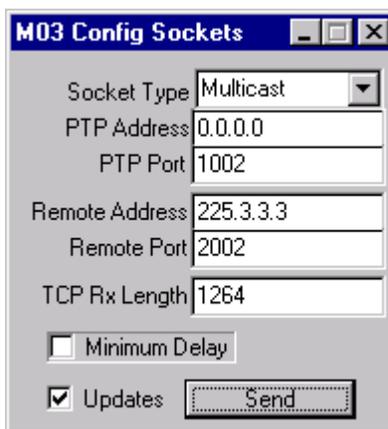


Figure 5-32: Network Sockets 1 Configuration

Network Sockets Module #2 (Multicast Address B) Configuration

Click on the *Configure* button. Configure the module as shown in **Figure 5-33**. Again, the *Remote Address* has to be a suitable Class D Multicast address. Click *Send* to accept the changes and close the configuration window.

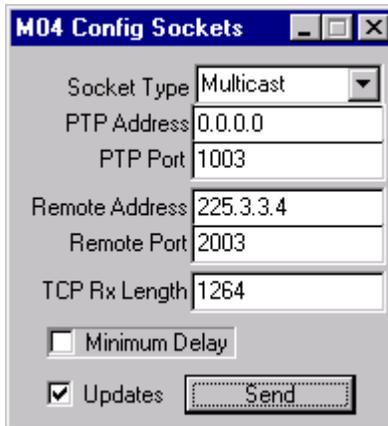


Figure 5-33: Network Sockets 2 Configuration

CPU Timer Module Configuration

Click on the *Configure* button. Configure the module as shown in **Figure 5-34**. Click the *Send* button to accept any changes and close the configuration window.



Figure 5-34: CPU Timer Module Configuration

Module connections must now be made. Select *Edit Module Connections* from the *Remote Desktop* menu.

File Playback Module Connections

Module 1: Make a connection from *Dataport 1* to the Virtual Channel Processor.

Virtual Channel Processor Module Connections

Module 2: Make a connection from *Dataport 1* to the Network Sockets Module for Remote Host A. Make a connection from *Dataport 2* to the Network Sockets Module for Remote Host B.

CPU Timer Module Connections

Module 5: Make an *Event* connection to the File Playback module.

Now that all the modules are connected appropriately, close the *Edit Module Connections* window. To run the desktop, select *Enable All Streams* from the *Remote Desktop* menu. If you open the Status window for the File Playback module, you will see a status bar indicating how much of the file has been played, as well as a display in bytes. The Status window for the Virtual Channel Processor module will show how many frames from each virtual channel have been taken in and processed, and if any errors occurred.

If you wish to verify each virtual channel, two Null Transceiver Modules can be loaded onto the desktop and connected to the Virtual Channel Processor. Connecting each Null Transceiver module to a different output port of the Virtual Channel Processor will allow you to view the data streams as they are being sent out the Network Sockets modules.

D. Telemetry Acquisition and Processing 2 Desktop

This final desktop will be another example of a Telemetry Acquisition System. In the previous Telemetry Acquisition and Processing example, the desktop was configured to take in telemetry via a network socket interface. In this example, hardware devices, represented by modules, will be configured to take in the data. The modules that will be used are a Bit Synchronizer, Serial Input, Time Processor, Virtual Channel Processor, IPDU Formatter, File Recorder, and Network Sockets modules.

The telemetry stream will enter the Bit Synchronizer. The Bit Sync recovers the clock from the input data signal. The Bit Sync module is used to configure both the Bit Synchronizer and Viterbi Decoder. If Viterbi decoding is enabled, the *Bit Rate* entry in the configuration dialog should be set to twice the actual data bit rate for rate $\frac{1}{2}$ convolutional decoding. The PTP NT has been designed to be compatible with multiple COTS products, and thus there are two unique Bit Sync modules that can be used. These are the Veda Bit Sync module and the Aydin Bit Sync module. In this example, we are using the Veda Bit Sync module. However, the module you select will be determined by the specific hardware installed in your PTP NT system.

The data is then sent to the Frame Synchronizer (Serial Input module). A time stamp is provided by the Time Code Processor. The Serial Input module then sends frame data with quality annotation and time stamp to the Virtual Channel Processor module. The Virtual Channel Processor can perform software Reed-Solomon decoding and CRC decoding for low data rate streams. However, if the MONARCH-E is being used as the Serial Input Module, these features can be disabled in the VC Processor since the MONARCH-E performs these functions in hardware.

The Virtual Channel Processor will then be configured to send Frames with VCID 0 to data port 1 for network transfer and Frames with VCID 1 and 2 to data port 2 for recording to hard disk.

VCID 0 frames will be received by the IPDU Formatter module. This module will format these frames for transfer over the network by accepting a 1264 byte frame from the Virtual Channel Processor and prepending the 32-byte IPDU header. The encapsulated VCID 0 frame data will

then be received by the Network Sockets module and sent to a TCP/IP client for network transfer.

The File Recorder module will receive VCID 1 and 2 frame data from the Virtual Channel Processor and write the data to a file on the hard disk.

Load the following modules onto the PTP Desktop:

1. M0101_VedaBitSync.dll - Veda Bit Sync Module
2. M0201_AvtecSerialInput.dll - Serial Input Module
3. M1002_VirtualChannelProcessor.dll - Virtual Channel Processor Module
4. M100E_IPDUFormatter.dll - IPDU Formatter Module
5. M0302_FileRecorder.dll - File Recorder Module
6. M0401_NetworkSockets.dll - Network Sockets Module
7. M0105_Time.dll - Time Code Processor Module

The following is a typical configuration for these modules:

Veda Bit Synchronizer Configuration

Click on *Configure*. Configure the module as shown in **Figure 5-35**. Click the *Send* button to accept the changes and close the configuration window.

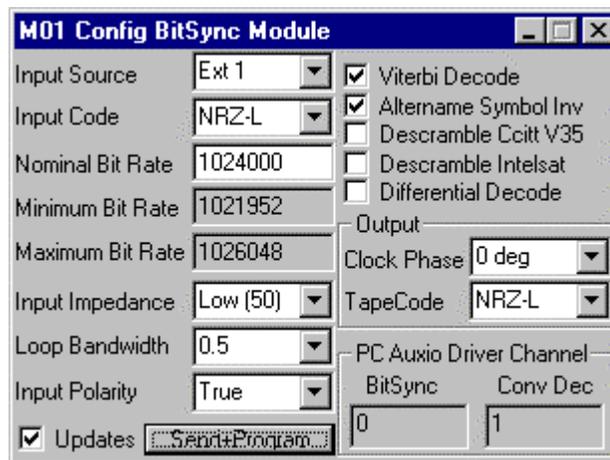


Figure 5-35: Veda Bit Sync Module Configuration

Serial Input Module Configuration

Click on *Configure*. Configure the module as shown in **Figure 5-36**. Click on the *Extended Options* button and configure this new window as shown in **Figure 5-37**. Click the *Send* button to accept the changes and close the configuration windows.

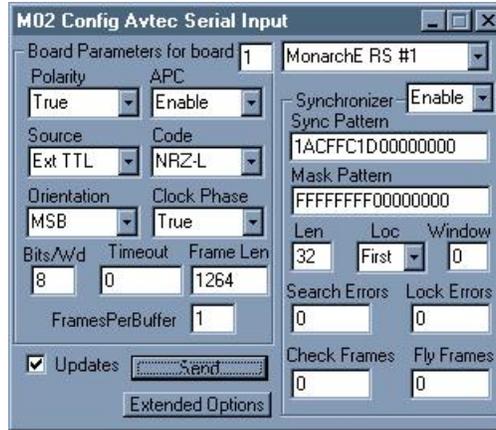


Figure 5-36: Avtec Serial Input Module Configuration

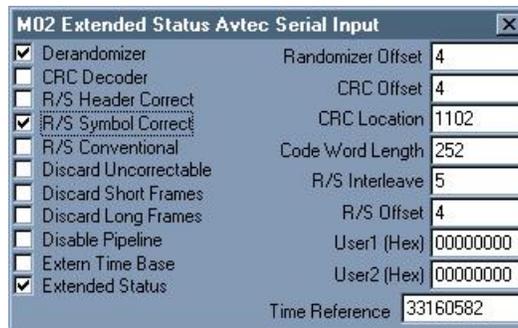


Figure 5-37: Avtec Serial Input Extended Options Configuration

Virtual Channel Processor Configuration

Click on *Configure*. Configure the module as shown in **Figure 5-37**. Click on the *Extended Options* button and configure this new window as shown in **Figure 5-38**. Click the *Send* button to accept the changes and close the configuration windows.

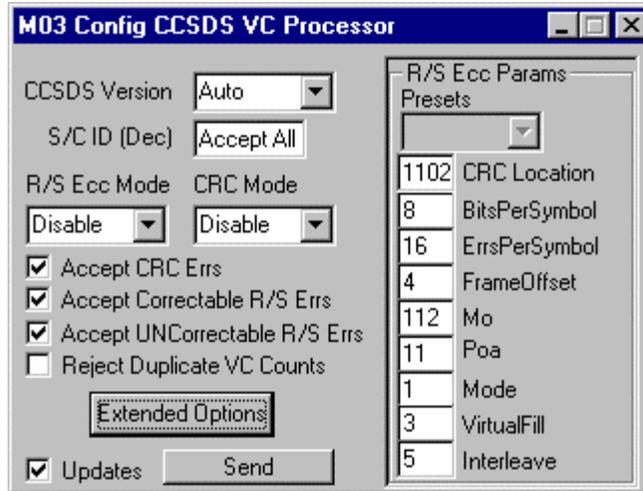


Figure 5-38: Virtual Channel Processor Configuration

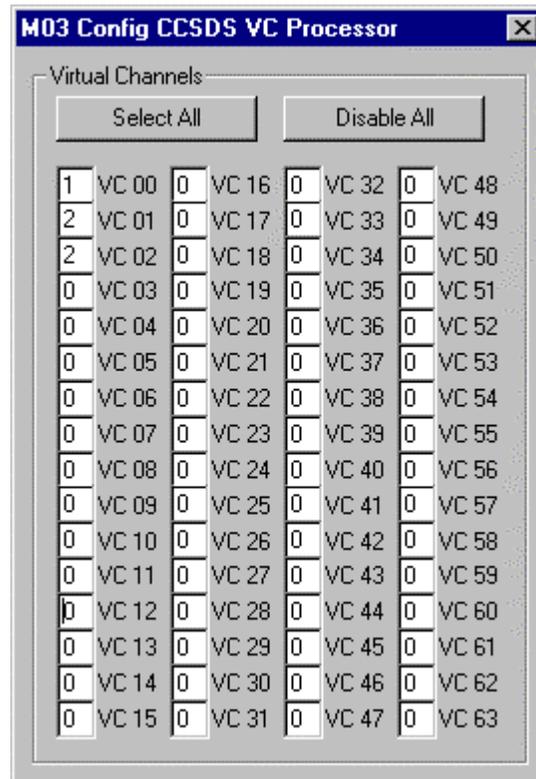


Figure 5-39: Virtual Channel Processor Extended Options Configuration

IPDU Formatter Configuration

Click on *Configure*. Configure the module as shown in **Figure 5-40**. Click the *Send* button to accept the changes and close the configuration windows.

Figure 5-40: IPDU Formatter Module Configuration

File Recorder Configuration

Click on *Configure*. Configure the module as shown in **Figure 5-41**. Click the *Send* button to accept the changes and close the configuration windows.

Figure 5-41: File Recorder Module Configuration

Network Sockets Configuration

Click on *Configure*. Configure the module as shown in **Figure 5-42**. Click the *Send* button to accept the changes and close the configuration windows.

Figure 5-42: Network Sockets Module Configuration

Time Code Processor Configuration

Click on *Configure*. Configure the module as shown in **Figure 5-43**. Click the *Send* button to accept the changes and close the configuration windows.

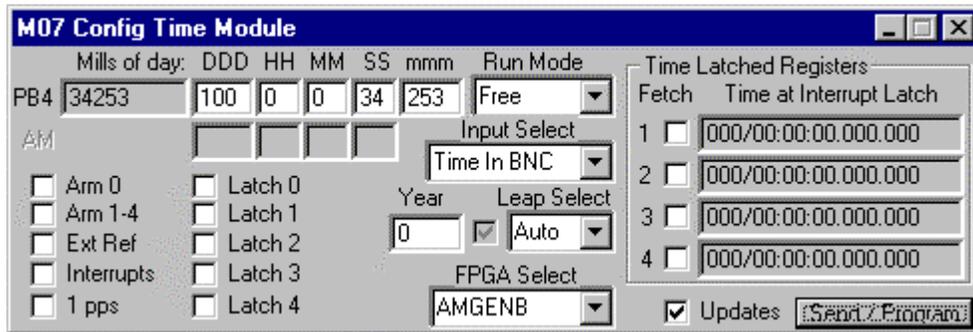


Figure 5-43: Time Module Configuration

Now, the following connections will need to be made between the modules. Select *Edit Module Connections* from the *Remote Desktop* menu.

Veda Bit Synchronizer Connections

The connection between the Bit Synchronizer and the Frame Synchronizer is not a software connection. A hard wired connection will have to be made between these boards. In other words, data flow between these boards occurs via external cable connections. Make the appropriate external connections.

Serial Input Module Connections

Module 2: Make a connection from *DataPort 1* to the Virtual Channel Processor Module.

Virtual Channel Processor Connections

Module 3: Make a connection from *DataPort 1* to the IPDU Formatter Module and from *DataPort 2* to the File Recorder Module.

IPDU Formatter Connections

Module 4: Make a connection from *DataPort 1* to the Network Sockets Module.

When the streams are enabled, data will taken in by the Bit Sync. The Bit Sync will then pass clock and data to the Frame Sync. The frame synced data will be time stamped, processed by the Virtual Channel Processor Module. The Virtual Channel Processor will transfer the individual VCIDs either to disk or to the network, as configured.

This previous section outlined a typical hardware/software configuration for a PTP NT system. Much of the hardware and software configuration can and will vary depending on the user's particular configuration.

Section 3 - Script Files

The PTP NT Server is the real-time portion of the PTP NT software. When modules are being loaded and module configurations and connections are being made from the Console window, what is actually happening is that the Console is sending commands to the Server, and the Server is loading the modules and making the connections between them. The Console window is a GUI that simplifies the task of configuring the PTP and allows these tasks to be accomplished from a remote computer.

However, many of these commands can be entered as command strings from the Server window itself. Most of the functions available in the GUI have a command line counterpart in the Server window. To use the commands, the Server window must be made active and the command string is typed in as a one-line command similar to a command typed from a DOS Shell prompt.

The most useful feature of having these commands available is the ability to write script files to quickly load and configure various desktops. A script file is a text file containing Server Commands that are executed line-by-line by the PTP NT when the script file is run. A script file can be used to load modules, configure data and event connections, and enable modules. Much of the configuration of a module must still be completed via the Console GUI, however, a script file can be a quick way to get a desktop started. Appendix A contains a list of the PTP NT Server commands and their respective definitions and usage.

This next example will use the PTP NT Server commands in writing a script file to setup and configure connections in a BERT Loopback desktop, the same one that was designed in *Section 2 - Building Desktops* of this Chapter.

To create a script file, open up a text editor such as the *Windows Notepad*. The BERT Loopback desktop can be built with the following string of commands:

```
NEWDESKTOP
ADDMODULE C:\PTP NT\modules\M0202_AvtecSerialOutput.dll
ADDMODULE C:\PTP NT\modules\M0201_AvtecSerialInput.dll
ADDMODULE C:\PTP NT\modules\M1001_BitErrorRateTester.dll
ADDMODULE C:\PTP NT\modules\M1000_NullTransciever.dll
ADDEVENT 1 3 0 0          */Output Module Connection/*
ADDOUTPUT 2 3 1 0        */Input Module Connection/*
ADDOUTPUT 2 4 1 0        */Input Module Connection/*
ADDOUTPUT 3 1 1 0        */BERT Module Connection/*
```

Save the file as *C:\ptp_user\Scripts\BERTSCRIPT.PTP*. Open the PTP NT Server and execute the following command:

```
READSCRIPTFILE C:\ptp_user\scripts\bertscript.ptp
```

After the file has been executed, the Server display should have four modules loaded, and the status for module 4 (the Null Transceiver Module) should be the active display. If further configuration of the modules is needed, then a Console window needs to be connected to the Server, and the configuration changes have to be made via the GUI.

The remaining desktops that were described in Section 2 of this Chapter can be loaded in a similar fashion. A good exercise in writing script files would be to recreate those desktops using the Server Commands. This will be left to the reader.

Appendix A:

PTP NT SERVER FUNCTIONS

PTP NT Server Commands

ADDEVENT [source] [destination] 0 0

This command allows an event connection to be made between two modules.

source - event source module number.

destination - event destination module number.

0 - future options, currently must be set to 0 (zero).

ADD LICENSE

This command is used to register the PTP NT Server software. When the software is not registered, the Server will only run in a demo mode for 20 minutes. All commands are available in this mode, however, after 20 minutes, the PTP NT desktop contents are cleared and a blank desktop is loaded. To register the PTP NT Server software, obtain a key from Avtec Systems, Inc., run the above command from the Server command line, and input the key when asked.

ADDMODULE [path\module_file_name]

This command opens the *Add Module* GUI, allowing the user to select a module to load. Typing the module name with the command loads that particular module. The module name must consist of the complete path (location) of the module and the module's filename. For example, to load the Serial Input Module, the command would be:

ADDMODULE C:\PTP NT\MODULES\M0201_AvtecSerialInput.dll

The above command could be used within a Script File to load the Serial Input Module without having to go through the *Add Module* GUI box.

ADDOUTPUT [source] [destination] [port#] [input]

This command allows a data flow connection to be made between two modules.

source - data stream source module number.

destination - data stream destination module number.

port# - port number of source module to send data from. Valid values are 1 - 32.

input - 0 (zero).

DISABLE [module#, all, display, ping, debug]

This command disables a particular stream (module), all streams, or the display, depending on the variable that follows the command.

module# - input a module number with the *disable* command to deactivate a particular module.

all - this variable deactivates all modules currently loaded.

display - this variable turns the Server display off.

ping - this variable disables the network assurance ping that is sent by any open TCP receiver socket within the PTP NT. The network assurance ping is used to make sure that the TCP connection is still "alive". This is enabled by default.

debug - this variable disables the logging of debug information to a file.

ENABLE [module#, all, display, ping, debug]

This command enables a particular stream (module), all streams, or the display, depending on the variable that follows the command.

module# - input a module number with the *enable* command to activate a particular module.

all - this variable activates all modules currently loaded.

display - this variable turns the Server display on (the display defaults to on whenever the black-box is first loaded).

ping - this variable enables the network assurance ping that is sent by any open TCP receiver socket within the PTP NT. The network assurance ping is used to make sure that the TCP connection is still "alive". This is enabled by default.

debug - this variable enables the logging of debug information to a file. Debug information is stored in the file *%SystemRoot%\ptpdebug.txt*, where *%SystemRoot%* is the system's root directory (usually C:\).

EXIT

This command closes the PTP NT Server program.

LOADDESKTOP [path\filename]

This command opens the *Load Desktop* window box, or, when followed by a path and filename, opens the specified desktop. The default file extension is *DTP*.

NEWDESKTOP

This command removes all currently loaded modules from the desktop and returns an empty Server desktop to the user.

Q

This command closes the PTP NT Server program.

QUIT

This command closes the PTP NT Server program.

READSCRIPTFILE [path\filename]

This command is used to execute a PTP NT Script File that has been written. Simply follow the command with the path and filename of the script file (default extension is *PTP*), and the file will be executed.

REMOVEEVENT [source] [destination] 0 0

This command allows an event connection to be removed from between two modules.

source - event source module number.

destination - event destination module number.

0 - future options, currently must be set to 0 (zero).

REMOVEMODULE [module#]

Removes the specified module from the PTP desktop.

REMOVEOUTPUT [source] [destination] [port#] [input]

This command allows a data flow command to be removed from between two modules.

source - data stream source module number.

destination - data stream destination module number.

port# - port number of source module to send data from. Valid values are 1 - 32.

input - future option, currently must be set to 0 (zero).

SAVEDESKTOP [path\filename]

This command opens the *Save Desktop* window box, or, when followed by a path and filename, directly saves the current desktop. The default file extension is *DTP*.

SECUREBUFFERMODE [enable, disable]

This command enables or disables *Secure Buffer Mode* operation of the PTP. When data is passed from one module to another, what is actually passed is a “pointer” to the data. If the same data is passed to multiple modules, all modules that are to receive the data will get the same pointer. This can be a problem if any one of these modules is going to alter the data in any way. With *Secure Buffer Mode* enabled, when one data stream is passed to multiple modules, a copy of the data is made for each module. In this case, if one module changes the data, it will not affect the data being used by the other modules.

enable - enables Secure Buffer Mode.

disable - disables Secure Buffer Mode.

SETACTIVITYDELAY [x]

This command configures the time delay, in milliseconds, between activity status updates of a module. The activity status shows whether a module is currently performing a task. Activity status is displayed in the PTP NT Server window, in the rows marked *Rx Status* and *Tx Status*, using a 0 (inactive) or 1(active). The first column represents module 1, the second module 2, and so on, up to 32 modules.

SETCONTROLPORT [port#]

This command sets the control port for the PTP Server, which is the socket number assigned for controlling the system. Valid values are 0 through 65536.

SETDEBUGMODE [X]

This command sets the level of detail used when the PTP Server is recording debug information in a debug log file. Valid values for the parameter *x* are 0 – 255. The higher the number, the more detailed the information that is logged. Also see commands *ENABLE debug* and *DISABLE debug*.

SETFREQUENCY [module#] [frequency]

This command sets the transmitter frequency, in Hertz (Hz), for the Serial Output module only. The command line must include the module number of the Serial Output module and the desired frequency.

SETMODULETTL [Module#] [x]

This command sets the number of hops (*x*) a status message from a particular module takes before it is killed.

SETMULTICASTADDRESS [address]

This command sets the Class D multicast IP address to which PTP NT status messages are sent.

SETMULTICASTNIC [xxx.xxx.xxx.xxx]

This command allows the Server status messages to be routed to a user specified Network Interface Card (NIC) where [xxx.xxx.xxx.xxx] is the NIC's address.

SETMULTICASTPORT [port#]

This command sets the multicast port address on the local machine to the port number indicated. The multicast port address sets which port PTP NT status messages are sent out on. Valid values are 0 through 65536.

SETMULTICASTTTL [x]

This command sets the number of hops (*x*) a status message takes before it is killed. This is a global command for all modules.

SETSTATIONID [userid]

This command configures the PTP NT Server's User ID number. The value, *userid*, is a 32-bit number.

SETSTATUSDELAY [x]

This command configures the time delay, in milliseconds, of status updates between modules. For example, module 1's status is updated, then the Server waits *x* ms before updating module 2's status, and so on through all the modules.

Once the last module's status display is updated, the loop starts again at module 1. Therefore, if there are 7 modules loaded on the desktop with a status delay of x ms, then each module's display will be updated every $7x$ ms.

VIEW [module#, que, stats, normal, enable, disable]

The *view* command adjusts the display of the Server window.

module# - using the *view* command with a module number displays the status for that particular module.

que - shows the que (buffer) usage for the current module.

stats - shows the statistics normally displayed for the current module.

normal - same as *stats*.

enable - enables the Server display (actively shows module statistics).

disable - disables the Server display (shows no module/stream/status information).

WAIT [x]

Used when writing Script Files, this command inserts an "x" ms pause between the execution of commands.

ZERO [stream, all]

This command zeros the count displays for a particular stream (module), or all streams, depending on the variable that follows the command.

stream - input a module number with the *zero* command to set the count displays to zero for that particular module.

all - all module counts are set to zero.

Creating PTP NT Server Script Files using PTP NT Server Commands

The PTP NT Server is the real-time portion of the PTP NT software. When modules are being loaded and module configurations and connections are being made from the Console window, what is actually happening is that the Console is sending commands to the Server, and the Server is loading the modules and making the connections between them. The Console window is a GUI that simplifies the task of configuring the PTP and allows these tasks to be accomplished from a remote computer.

However, many of these commands can be entered as command strings from the Server window itself. Most of the functions available in the GUI have a command line counterpart in the Server window. To use the commands, the Server window must be made active and the command string is typed in as a one-line command similar to a command typed from a DOS Shell prompt.

The most useful feature of having these commands available is the ability to write script files to quickly load and configure various desktops. A script file is a text file containing Server

Commands that are executed line-by-line by the PTP NT when the script file is run. A script file can be used to load modules, configure data and event connections, and enable modules. Much of the configuration of a module must still be completed via the Console GUI, however, a script file can be a quick way to get a desktop started. Appendix A contains a list of the PTP NT Server commands and their respective definitions and usage.

This example will use the PTP NT Server commands in writing a script file to setup and configure connections in a BERT Loopback desktop (See *Chapter 5, Section 2 - Building Desktops*).

To create a script file, open up a text editor such as the *Windows Notepad*. The BERT Loopback desktop can be built with the following string of commands:

```

NEWDESKTOP
ADDMODULE C:\PTP NT\modules\M0202_AvtecSerialOutput.dll
ADDMODULE C:\PTP NT\modules\M0201_AvtecSerialInput.dll
ADDMODULE C:\PTP NT\modules\M1001_BitErrorRateTester.dll
ADDMODULE C:\PTP NT\modules\M1000_NullTransciever.dll
ADDEVENT 1 3 0 0          */Output Module Connection/*
ADDOUTPUT 2 3 1 0        */Input Module Connection/*
ADDOUTPUT 2 4 1 0        */Input Module Connection/*
ADDOUTPUT 3 1 1 0        */BERT Module Connection/*

```

Save the file as *C:\ptp_user\Scripts\BERTSCRIPT.PTP*. Open the PTP NT Server and execute the following command:

```
READSCRIPTFILE C:\ptp_user\scripts\bertscript.ptp
```

After the file has been executed, the Server display should have four modules loaded, and the status for module 4 (the Null Transceiver Module) should be the active display. If further configuration of the modules is needed, then a Console window needs to be connected to the Server, and the configuration changes have to be made via the GUI.

Securing Control Sockets and TCP Server Sockets in PTP NT

All TCP Server mode connections are socketed connections by definition. All PTP NT modules that use network connections have an entry for "PTP Port" that controls the local socket number. TCP Server mode connections are not limited to modules, but also include PTP NT Server Control Ports (sockets 4000 and 5000). A Well-Known Hosts file can be created for each local socket number, limiting access to that particular port to a select set of users.

When a client attempts to connect to a TCP Server mode socket (port) on a PTP NT machine, the PTPAPI sockets library will check for a Well-Known Hosts file. A Well-Known Hosts file is created by a PTP NT administrator, and contains a list of all clients that are allowed to connect to

a particular socket. If the file does not exist, all clients will be allowed to connect, either to control the PTP Server or send/receive data to/from the Server. If the file exists, the PTP NT compares the client's IP address and/or subnet mask to those listed within the Well-Known Hosts file. If the client is listed within the file, the connection is allowed, otherwise, it is refused. If the file exists, but is empty, no clients will be allowed to connect. In either case where the socket connection is refused, the PTP simply disconnects the socket and recycles for the next client.

The Well-Known Hosts file is a text file that is stored locally on a PTP NT machine. The naming convention is as follows:

`\ptp_user\security\Sxxxx.txt`

The directory "`\ptp_user\security`" is located on the same local drive as the PTP NT application. "`xxxx`" in the filename represents the socket (port) number in decimal.

The file can be created using a simple text editor, such as Windows Notepad. The file will be similar to **Figure A-1**. It will contain a list of all clients allowed on a particular socket, in standard dotted notation. As in **Figure A-1**, subnet masks are allowed by following the client's IP address with a "#" and the subnet mask.

```
150.000.130.000
150.000.130.025
148.011.192.110#255.255.255.0
192.110.000.124
192.192.030.222#255.255.255.0
```

Figure A-1: Sample Well-Known Hosts File

Appendix B:

IPDU HEADER SUMMARY

Table B-1 below gives the complete IPDU header structure. The three “Use for” columns indicate which header entries are meaningful for each of the three IPDU data types. The header is fixed at 32 bytes with unused fields set to zero.

Table B-1: IPDU Header Structure

Item No.	Field Name	Format & Size	Value	Use for Tlm	Use for Cmd	Use for Trk
1	IPDU synchronization	unsigned integer (4 bytes)	74C2472C hex	x	x	x
2	IPDU length in bytes (everything from the start of IPDU sync field through the end of the IPDU)	unsigned integer (4 bytes)		x	x	x
3	Data type ID, consisting of:	(4 bytes total)				
3a	IPDU Source	unsigned integer (8 bits)	ID for source of IPDU (see Table B-2 for values)	x	x	x
3b	IPDU Destination	unsigned integer (8 bits)	ID for destination of IPDU (see Table B-2 for values)	x	x	x
3c	Message Type	unsigned integer (8 bits)	(see Table B-3 for values)	x	x	x
3d	Spare	(8 bits)	0			
4	Header version number	unsigned integer (4 bits)	1	x	x	x
5	Data type	unsigned integer (4 bits)	0 (unused for L7)			
6	Message Time (GMT)	NASA PB-5 Code (Option C) (7 bytes)	(See Table B-4 for values)	x	x	x
7	Ground station physical port ID	unsigned integer (1 byte)	0 (unused for L7)			
8	Source VCDU sequence counter discontinuity	logical (1 bit)	0 = no source VCDU discontinuity 1 = source VCDU discontinuity detected	x		
9	VCDU contains playback data	logical (1 bit)	0 = real-time telemetry 1 = playback telemetry	x		

Item No.	Field Name	Format & Size	Value	Use for Tlm	Use for Cmd	Use for Trk
10	Recovery processing indicator	logical (1 bit)	0 = live from spacecraft 1 = playback from ground station	x		x
11	Test data indicator	logical (1 bit)	0 = operational data 1 = test data	x	x	x
12	CRC failure indicator	logical (1 bit)	0 (unused for L7)			
13	Path SDU source sequence counter discontinuity	logical (1 bit)	0 (unused for L7)			
14	Packet length error	logical (1 bit)	0 (unused for L7)			
15	Packet fill indicator	logical (1 bit)	0 (unused for L7)			
16	Spare	(2 bits)	0			
17	Source VCDU ID, consisting of:	unsigned integer (14 bits total)				
17a	Spacecraft ID	unsigned integer (8 bits)	15 hex for L7	x	x	x
17b	Virtual Channel ID	unsigned integer (6 bits)	All VCDUs for L7 telemetry have a VCID = 0	x		
18	Location of first octet of ground-generated fill data for a path SDU	unsigned integer (2 bytes)	0 (unused for L7)			
19	Spare	(4 bytes)	0			
20	Reed-Solomon error control flag	logical (1 bit)	0 = no errors or errors were corrected 1 = uncorrectable errors	x		
21	Source VCDU header error decode results	unsigned integer (5 bits)	0 (unused for L7)			
22	Source VCDU error decode results	unsigned integer (10 bits)	This field is applicable only if the value of the Reed-Solomon error control flag = 0: 0 = no errors, If > 0, the number of corrected bits within the entire VCDU	x		
	Total Header Length	32 bytes				

Table B-2: L7 IPDU Source and Destination Codes

Source/Destination Name	Corresponding Integer Code (8 Bits) (Hex)	Corresponding 3 Character Ascii Code
Landsat-7 MOC	01	L7M
LGS, Sioux Falls, SD	02	SFS
AGS	03	ASK
SGS	04	SPZ
EOS White Sands Communications	05	WSC
Weilheim, Germany	06	WIL
Valley Forge, PA	07	VLF
Vandenburg AFB, CA	08	VAF
WPS	60	WFF

Table B-3: L7 IPDU Data Types

Message Type	Corresponding Integer Code (8 bits) (hex)
Narrowband real-time telemetry	01
Narrowband spacecraft recorder telemetry	02
Command data message	03
Command echo message	04
One-way tracking data	05
Two-way tracking data	06

Table B-4: NASA PB-5 Time Code Format (Option B)

Item No.	Field Name	Format & Size	Value
1	Flag bit	logical (1 bit)	1
2	Truncated Julian Day	unsigned integer (14 bits)	Truncate the most significant decimal digits, retaining only the four least significant ranging from 0 to 9999. The current Julian day epoch begins at midnight 1995, October 9, 10. October 10, 1995 is day 0.
3	Seconds of Day	unsigned integer (17 bits)	range = 0 to 86399
4	Milliseconds of Second	unsigned integer (10 bits)	range = 0 to 999
5	Microseconds of a Millisecond	unsigned integer (10 bits)	range = 0 to 999
6	Spare	(4 bits)	
	Total Length	7 bytes	

Table B-5: EDOS Service Header

Item No.	Name	Format & Size	Data Characteristics
1	ESH Version number	Unsigned Integer 4 Bits	Range -> 0- 15. (0=initial version, and incremented by 1 for each ESH modification thereafter)
2	SDU Type	Unsigned Integer 4 Bits	Value -> 0=VCDU, 1=CCSDS Packet (Path SDU), and 2=CLCW. Note: AM-1 provides VCDUs for the spacecraft AM-1 Trash Buffer
3	Date and Time Annotation of return link SDU receipt at EDOS.	NASA PB-5 Code Format 7 Bytes	Range -> Refer to Table 5.1.2.1-3 for the NASA PB-5 Code Format (containing GMT).
4	EDOS Physical Port Identification	Unsigned Integer 1 Byte	Range -> 0-63
5	Source VCDU Sequence Counter Discontinuity	Logical Bit 1 Bit	Values -> 0=false (No source VCDU discontinuity) and 1=true (source VCDU discontinuity)
6	VCDU contains Playback data	Logical Bit 1 Bit	Values -> 0 = False (nonplayback data) and 1 = True (playback data) (from the on-board spacecraft recorder)
7	Recovery Processing Indicator	Logical bit 1 Bit	Values -> 0 = false (live) and 1= true (data capture playback) (from EDOS's data capture recovery processing)
8	Test data indicator	Logical Bit 1 Bit	Values -> 0 = false (live-operational data) and 1 = true (EDOS test data)
9	Cyclic Redundancy Check (CRC) failure indicator	Logical Bit 1 Bit	Field is present, but it is not applicable for AM-1. AM-1 is a Grade 2 service using Reed-Solomon. Value = 0 (FALSE) for AM-1
10	Path SDU Source Sequence	Logical Bit	Values -> 0 = false (no

	Counter Discontinuity	1 Bit	discontinuity); 1 = true (packet discontinuity)
11	Packet Length Error	Logical Bit 1 Bit	Value -> 0=no detected length error; 1=detected length error.
12	Packet Fill Indicator	Logical Bit 1 Bit	Value -> 0=packet doesn't contain fill data; 1=packet contains fill data.
13	Fill/Spare, reserved for future use.	Unsigned Integer 2 Bits	Value -> zero (0)
14	Source VCDU-ID	Unsigned Integer 14 Bits	Spacecraft ID and VCID (SCID-8 Bits and followed by VCID-6 Bits (Extracted for VCDU))
15	Location of first octet of EDOS generated fill data for a Path SDU	Unsigned Integer 2 Bytes	Range -> 0 - 65535
16	Fill/Spare, reserved for future use	Unsigned Integer 4 Bytes	Value -> zero (0).
17	Reed-Solomon Error Control Flag	Logical Bit 1 Bit	Values -> 0=correctable or no errors; 1=Failed/uncorrectable Flag indicates whether the VCDU failed or passed R-S decoding.
18	Source VCDU Header Error Decode Results	Unsigned Integer 5 Bits	Values -> 0=error free. Field contains the number of corrected symbols within the VCDU header.
19	Source VCDU error decode results	Unsigned Integer 10 Bits	Values -> 0=error free Field contains the number of corrected symbols from the entire VCDU.
The EDOS Service Header has a total of twenty bytes			

Table B-6: NASCOM RTP Header

Item No.	Name	Size	Data Characteristics
1	version (V)	2 bits	This field identifies the version of RTP.
2	Padding (P)	1 bit	If the padding bit is set, the packet contains one or more additional padding octets at the end which are not part of the payload.
3	Extension (X)	1 bit	If the extension bit is set, the fixed header is followed by exactly one header extension, with a format as defined in RFC 1889, Section 5.3.1.
4	CSRC count (CC)	4 bits	The CSRC count contains the number of CSRC indentifies that follow the fixed header.
5	Marker (M)	1 bit	The interpretation of the marker is defined by a profile. A profile may define additional marker bits or specify that there is no marker by changing the number of bits in the payload type field. (see RFC 1889, Section 5.3)
6	Payload type (PT)	7 bits	This field identifies the format of the RTP payload and determines it interpretation by the application. A profile specifies a default static mapping of payload type codes to payload formats.
7	Sequence number	16 bits	The sequence number increments by one for each RTP data packet sent, and may be used by the receiver to detect packet loss and to restore packet sequence. The initial value of the sequence number is random.
8	Timestamp	32 bits	The timestamp reflects the sampling instant of the first octet in the RTP data packet.

Item No.	Name	Size	Data Characteristics
			The resolution of the clock must be sufficient for the desired synchronization accuracy and for measuring packet arrival jitter. The clock frequency is dependent on the format of data carried as payload and is specified statically in the profile or payload.
9	SSRC	32 bits	The SSRC field identifies the synchronization source. This identifier is chosen randomly, with the intent that no two synchronization sources within the same RTP session will have the same SSRC identifier.

Note: 'word' refers to 16 bits, an 'octet' or 'byte' refers to 8 bits.

Table B-7: LEO-T Telemetry Frame Delivery Header (TFDH)

Word	Bit(s)	Description
1	1-2	Version number of telemetry header
1	3-16	Length of frame in bytes (including header)
2	1	Annotation Flags: Reed-Solomon (RS) decoding enabled flag
2	2	RS decoding error flag
2	3	Frame CRC enabled flag
2	4	Frame CRC pass/fail flag
2	5	Master channel sequence (MCS) checking enabled flag
2	6	MCS number error
2	7-8	Data inversion flags
2	9-10	Fame sync make flags
2	11	Data forward/reverse flag
2	12-16	Data Class
3	1	Earth received time of data (PB-5 format): flag bit
3	2-15	Earth received time of data (PB-5 format): truncated Julian Day
3	16	Earth received time of data (PB-5 format): seconds of

Word	Bit(s)	Description
		day
4	1-16	Earth received time of data (PB-5 format): seconds of day
5	1-10	Earth received rime of data (PB-5 format): milliseconds of a second
5	11-16	Fill / spare

The fields in the LEO-T telemetry frame delivery unit are defined as follow. Bit 1 is the most significant bit and is transmitted first.

Word 1

- Bits 1-2: Version number for header format is set to "01" for frame data
 Bits 3-16: Length of frame in bytes, including frame delivery header.

Word 2

- Bit 1: Reed-Solomon decoding: "0" disabled, "1" enabled
 Bit 2: RS decoding error: "0" Frame was corrected, "1" Frame could not be corrected
 Bit 3: Frame CRC checking: "0" disabled, "1" enabled
 Bit 4: Frame CRC error - "0" pass, "1" fail
 Bit 5: Master Channel Sequence (MCS) checking: "0" disabled, "1" enabled
 Bit 6: MCS number error - "0" number increased monotonically, "1" number increased by 2 or more
 Bits 7, 8: Data inversion flags: "00" data true, "01" data inverted, "11" data inverted and corrected
 Bits 9, 10: Frame sync mode flags: "0" search frame, "01" check frame, "10" lock frame, "11" flywheel frame
 Bit 11: Data forward/reverse flag: "0" forward, "1" reverse
 Bits 12-16: Data Class 1 = CCSDS Frame, 2 = CCSDS Packet, 3 = TDM Frame, 4 = Stripped TDM Frame

Words 3-5

Earth received time as defined below. This is the UTC time when the telemetry frame is received by the telemetry processor. The accuracy of the time is 1 millisecond.

Word 3

- Bit 1: PB-5 flag bit, set to "0"
 Bits 2-15: Truncated Julian day (14 bits). Truncate the most significant decimal digits, retaining only the four least significant decimal digits ranging from 0000 to 9999. The current Julian day epoch begins on October 10, 1995, and continues for a period of 27 years. Value is binary unsigned integer.
 Bit 16: Seconds of day (17 bits), most significant bit.

Word 4

Bits 1-16: Seconds of day (17 bits), 16 least significant. Value is binary unsigned integer, ranging from 0 to 86,399.

Word 5

Bits 1-10: Milliseconds of the second. Value is binary unsigned integer from 9 to 999.

Bits 11-16: Fill/spares, set to zero's.

Note: 'word' refers to 16 bits, an 'octet' or 'byte' refers to 8 bits.

Table B-8: LEO-T Command Delivery Header (CDH)

BYTE(S)	Description
1	Message Type
2	Fill/Spare - Reserved for Future Use
3	Source Identification
4	Destination Identification
5	Fill/Spare - Reserved for Future Use
6-12	Message generation time - NASA PB-5 Time code format
13-14	Spacecraft Identification
15-16	Message Sequence number
17-18	EDOS software version number
19-20	Message Length
21-24	Fill/Spare - Reserved for future use
Command data field	

The fields in the TRACE command delivery header are defined as follows. Bytes are transmitted MSB first. Fields that are denoted with "*" are command validation parameters and shall be included in the configuration database. Remaining fields (except for Message Length) are not used by the TRACE and are for information purposes only.

Byte1: * Message type is 03 hex (Command data block)

Byte 2: Fill/spares, set to zero's.

Byte 3: * Source Identification

Byte 4: Destination Identification

Byte 5: Fill/spares, set to zero's

Bytes 16-12:

Earth received time as defined below. This is the UTC time when the command ground transfer unit is generated.

Bytes 6,7

- Bit 1: PB-5 flag bit, set to "0"
- Bits 2-15: Truncated Julian day (14 bits). Truncate the most significant decimal digits, retaining only the four least significant decimal digits ranging from 0000 to 9999. The current Julian day epoch begins October 10, 1995, and continues for a period of 27 years. Value is binary unsigned integer.
- Bit 16: Seconds of day (17 bits), most significant bit.

Bytes 8,9

- Bits 1-16: Seconds of day 17 (bits), 16 least significant. Value is binary unsigned integer, ranging from 0 to 86,399.

Bytes 10,11

- Bits 1-10: Millisecond (10 bits). Value is binary unsigned integer from 0 to 999.
- Bits 11-16: Microseconds of milliseconds (10 bits), 6 most significant. Value is binary unsigned integer from 0-999.

Byte 12

- Bits 1-4: microseconds of milliseconds, 4 least significant
- Bits 5-8: Fill, set to zero's

Bytes 13,14: * Spacecraft ID

Bytes 15,16 Message sequence number

Bytes 17,18: EDOS software version number, set to TBD

Bytes 19,20: Message Length. Number of bytes in the message, including the LEO-T command delivery header and the command data. Range is 24 to 8192.

Bytes 21-24: Fill/spares, set to zero's

Appendix C:

ACRONYMS AND ABBREVIATIONS

APID	Application ID
BERT	Bit Error Rate Tester
BI Φ L	Bi-phase Level
BI Φ M	Bi-phase Mark
BI Φ S	Bi-phase Space
BIT SYNC	Bit Synchronization
BPSK	Binary Phase Skift Keying
CCSDS	Consultative Committee for Space Data Systems
CLK	Clock
CLTU	Command Link Transmission Unit
COM	Communications
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DLL	Dynamic Link Library
DQM	Data Quality Monitor
Hex	Hexadecimal
GMT	
I/O	Input/Output
IDE	Integrated Drive Electronics
IP	Internet Protocol
IPDU	Inter-project Data Unit
IRIG	Inter-range Instrumentation Group
IRQ	Interrupt Request
ISA	Industry Standard Architecture

Acronyms and Abbreviations - (continued)

LAN	Local Area Network
LSB	Least Significant Bit
MB	Megabyte
Mbps	Megabits Per Second
MHz	Megahertz
MSB	Most Significant Bit
msec	millisecond
NRZL	Non-return to Zero Level
NRZM	Non-return to Zero Mark
NRZS	Non-return to Zero Space
PB	Parallel Binary (<i>can be numbered 1 through 5</i>)
PCI	Peripheral Component Interface
PCM	Pulse Code Modulation
PSK	Phase Shift Key
PTP	Programmable Telemetry Processor
ROM	Read-only Memory
RS	Reed-Solomon
Sync	Synchronizer
SCSI	Small Computer Systems Interface
TCP	Transmission Control Protocol
TDM	Time-division Multiplex
TTL	Time to Live
UDP	User Datagram Protocol
Uncon.	Unconnected
VCDU	Virtual Channel Data Unit
VCID	Virtual Channel ID
VCP	Virtual Channel Processor
WAN	Wide Area Network