ADST/

ADST Software Maintenance Manual for the Protocol Translator of Advanced Distributed Simulation Technology/ Crew Station Research and Development Facility

Loral Western Development Labs
Electronic Defense Systems Software Department
Software Engineering Laboratory
3200 Zanker Road
P.O. Box [illegible]
San Jose, CA [illegible]

April 1, 1993

Contract No. N61339-91-D-0001
CDRL No. Not Applicable

Prepared for:

Naval Training and Instrumentation Command
Naval Training Systems Center
12350 Research Parkway
Orlando, FL 32826-3275

REPORT DOCUMENTATION PAGE

Form approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1 April 1993	

4. TITLE AND SUBTITLE
ADST Software Maintenance Manual for the Protocol Translator of Advanced Distributed Simulation Technology/Crew Station Research and Development Factility

5. FUNDING NUMBERS
Contract No. N61339-91-D-0001

6. AUTHOR(S)
Au-Yeung, Anna; Kuczaj, Chuck; Bright, Rick; Thompson, Lynn, Mitchell, Gerry; Kruck, Mary

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Loral Systems Company
ADST Program Office
12443 Research Parkway, Suite 303
Orlando, FL 32826

8. PERFORMING ORGANIZATION REPORT NUMBER
ADST/WDL/TR-93-003064

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Simulation, Training and Instrumentation Command
STRICOM
Naval Training Systems Center
12350 Research Parkway
Orlando, FL 32826-3275

10. SPONSORING ORGANIZATION REPORT

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (Maximum 200 words)

This document provides an overview of the hardware and software in building the Protocol Translator.

14. SUBJECT TERMS

15. NUMBER OF PAGES
25

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	17. SECURITY CLASSIFICATION OF THIS PAGE	17. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

1.	Introduction	1
1.1	Scope	1
1.2	Purpose	1
1.3	Referenced documents.	1
2.	Overview	3
2.1	Architectural Overview	3
3.	Hardware Configuration	4
3.1	Hardware Description	4
3.1.1	Computer Peripherals	4
3.1.2	Custom Hardware	4
3.2	Hardware to Software Interfaces	4
4.	Software Description	5
4.1	Protocol Translator Software Description	5
4.1.1	Network Interface Software	5
4.1.2	PDU Transformation	6
4.1.3	Dead Reckoning	9
4.1.4	User Interface	9
4.2	Development Environment Description	10
4.2.1	Development Environment Directory Structure	10
4.2.2	Development Environment Build Configuration	12
4.3	Runtime Environment Description	13
4.3.1	Runtime Environment Directory Structure	13
4.3.2	Runtime Environment Data Files	14
4.3.3	Runtime Environment Startup Parameters	18
4.3.4	Runtime Protocol Translator User Interface Commands	18
4.4	Startup Procedure	18
4.4.1	Protocol Translator Startup Procedure:	18
4.4.1	Protocol Translator User Interface Startup Procedure:	18
5.	Utility Software	20
6.	Notes	24

Table of Figures

Figure 2.1 Protocol Translator Top Level Hardware Configuration 3
Figure 4.2.1 Protocol Translator Top Level Development Directory Structure 9

1. Introduction

1.1 Scope

This document provides an overview of the hardware and software in building the Protocol Translator. The overview is composed of five sections: introduction, architectural overview, hardware configuration, software configuration, and utility software. The Protocol Translator supports interoperability between two networks using different protocols, namely the Distributed Interactive Simulation (DIS) and Simulation Networking (SIMNET) protocols.

1.2 Purpose

This document describes the software and hardware needed to create a Protocol Translator, the function of each sub-directory in the software tree, the starting procedure, utility software, and module testing software.

1.3 Referenced documents.

[1] The SIMNET Network and Protocols, BBN Report No. 7627, June 1991, version 6.6.1

Prepared by:	Prepared for:
BBN Systems and Technologies	DARPA
10 Moulton Street	Information&Science Technology Office
Cambridge,	1400 Wilson Boulevard
Massachusetts 02138	Arlington, Virginia 22209-2308

[2] Military Standard - Protocol Data Units for Entity Information and Entity Interaction in a Distributed Interactive Simulation, DIS standard 1.0 with Extension.
Contract No. N61339-91-C-0091, STRICOM, DARPA, May 8, 1992.
Prepared by: Institute for Simulation and Training(IST) - IST-PD-91-1
12424 Research Parkway, Suite 300
Orlando FL 32826
University of Central Florida, Division of Sponsored Research.

[3] Map Projections - A Working Manual by John P. Snyder, U.S. Geological Survey Professional Paper 1395. (ref - Coordinate Transformation)

[4] Software Development Folder (SDF) for the Protocol Translator:
[4.1] Software Requirements Specification for the Protocol Translator of Advanced Distribution Simulation Technology (ADST) / Crew Station Research and Development Facility (CSRDF)
Rev. Basic: June 16, 1992, Contract No. N61339-91-D-0001

Prepared by: Loral Western Development Labs
3200 Zanker Road
P.O. Box 49041
San Jose, CA 95161-9041
Appendix B: PDU Translation Requirements, ADST CSRDF Requirements Analysis by Charles Kuczaj with Charles Von Hammerstein, May 18 1992, version 2

[4.2] Validation Tests, 1992 I/ITSC Interoperability Demonstration
S.H. Smith, June 16, 1992 - provided by Institution for Simulation and Training.

[5] Software Development File (SDF) for Network Interface:

[5.1] Internet Protocol, DARPA Internet Program, Protocol Specification
September 1981.

Prepared by:	Prepared for:
Information Science Institute	DARPA
University of Southern California	Information Processing Techniques
4676 Admiralty Way	Office. 1400 Wilson Boulevard
Marina del Rey, CA 90291	Arlington, Virginia 22209

[5.2] Interface Control Document (ICD) for the BDS-D Network Interface,
14 February 1992(Last updated 21-Feb-92 17:58)

[5.3] SIMNET Long Haul Network (LHN) Gateway Operations and Maintenance, ENCL 5

[5.4] A Standard for the Transmission of IP Datagram over IEEE 802 Networks
(Internet and Address Resolution Protocol on IEEE 802 Networks)
Network Working Group, Request for commnets(RFC): 1042
J. Postel and J. Reynolds, ISI, February 1988.

[5.5] The DARPA Wideband Network - Dual Bus Protocol
Winston Edmond, Karen Seo, Melisse Leib, Claudio Topolcic
Systems and Technology Division
Bolt Beranek and Newman. Inc,
Cambridge, Massachusetts 02138

[5.6] IP/UDP Network Performance, May 1992, Richard Sherman, Loral WDL

2. Overview

2.1 Architectural Overview

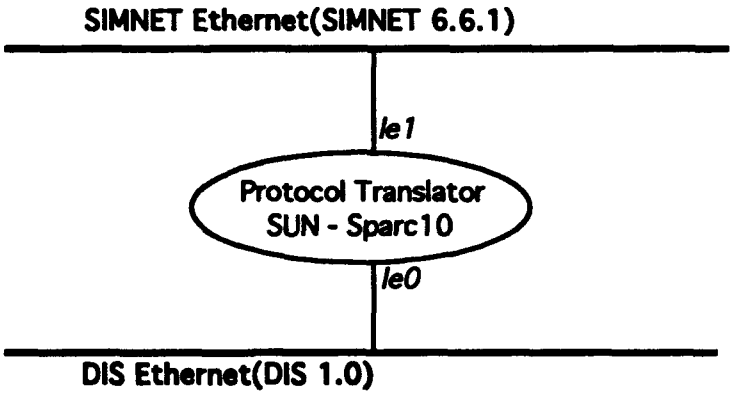

Figure 2.1 Protocol Translator Top Level Hardware Configuration

The Protocol Translator supports interoperability between SIMNET & DIS Simulation exercises within the constraints of translated PDUs. Refer to "The SIMNET Network and Protocols" and "Protocol Data Units (PDU) for Entity Information and Entity Interaction in a Distributed Interactive Simulation" for the SIMNET and DIS standards used in the Protocol Translator and PDUs translated.

The Protocol Translator consists of two parallel processes: DIS to SIM translation and SIM to DIS translation. The DIS to SIM translation receives a DIS UDP datagram from a designated DIS Ethernet Interface (le0) and sends the translated datagram in SIMNET format to the SIMNET Network through a designated SIMNET Ethernet Interface.

The SIM to DIS translation receives a SIMNET PDU from a designated SIMNET Ethernet Interface (le1) and sends the translated PDU in DIS format to the DIS Network through a designated UDP port.

3. Hardware Configuration

3.1 Hardware Description

The Protocol Translator executes on a Sun Microsystems Sparc10 or Sparc2 workstation with two Ethernet Interfaces, one for each network (SIMNET, DIS).

The Sun Sparc10 hardware configuration consists of:

One 400 Mega byte disk drive
64 Mega bytes Memory
One audio/AUI splitter adaptor
Two Ethernet Interfaces: - Internal on Mother board
- x1053A add-on Ethernet Interface Card with AUI adaptor.

3.1.1 Computer Peripherals

None.

3.1.2 Custom Hardware

There is no custom-built hardware incorporated in the Protocol Translator.

3.2 Hardware to Software Interfaces

The Protocol Translator utilizes vendor-supplied library (Sun's interfaces software) for connecting to Ethernet Interfaces. Refer to section 4.1.1 for more detail.

4. Software Description

4.1 Protocol Translator Software Description

The Protocol Translator invokes two parallel processes:
SIMNET to DIS Translation
DIS to SIMNET Translation.

Protocol Translator is divided into three sections:
Network Interface
PDU transformation - encode, decode, coordinate transformation
Dead Reckoning

4.1.1 Network Interface Software

The two Protocol Translator External Interfaces are SIMNET and DIS. The first Ethernet Interface (le0) is configured automatically during system boot. The second add-on Ethernet Interface (le1) can only be configured with changes to the /etc/rc.boot file (refer to section 4.3 for instruction) or it can be temporarily configured with the "ifconfig" command. A hostname file, consists of an interface hostname and must be created for each Ethernet Interface (see below for an example). Refer to reference [4.1] for SIMNET and DIS Protocol Translator External Interfaces Layout and [5.2] for Protocol Translator Module IPCs.

SIMNET Network Interface:
- Directs Ethernet interface with IEEE 802.3 network along with Sub-Network Access Protocol (SNAP).
- Interfaces between the Association layer (application and Logical Link Control (LLC). Reference to Figure 4.1.1.
- Receives PDUs in promiscuous mode and sends PDUs out with multicast address.

DIS Network Interface:
- Socket connection with Sun's Transport Layer Interface to User Datagram Protocol/Internet Protocol (UDP/IP). Refer to Figure 4.1.1.
- Interface between the Association(application) and Transport Layer.
- Receives any PDU that conforms to the dedicated Internet Address scheme and sends PDUs out with broadcast address.

With UDP/IP, all sites must agree upon a port number (e.g. 0x4128 - reference[5.2]) and an Internet Address scheme to make broadcasting PDUs possible. For the Protocol Translator, set the Internet Address in /etc/hosts.

Following are examples of /etc/hostname files and a /etc/hosts file:

Filename	File Content	Description
/etc/hostname.le0	adst23	hostname file: DIS network
/etc/hostname.le1	adst25	hostname file: SIMNET network
/etc/hosts	#	
	127.0.0.1 localhost	company's IP address
	#	
	137.249.44.25 adst25	SIMNET IP address
	137.249.32.48 adst21	DIS IP address
	137.249.32.23 adst23 loghost	login host(sparc10)

4.1.2 PDU Transformation

The Protocol Translator can handle transformations to and from a variety of commonly-found coordinate systems including

Geodetic (Latitude, Longitude, Elevation)
Universal Transverse Mercator (UTM) X,Y,Z
World Geodetic System (WGS) 84
World Geodetic System (WGS) 72

Additionally, support is provided for a number of specific databases commonly found in the SIMNET environment: Hunter Liggett, Fulda Gap and Fort Knox.

The current PT implementation is set up to transform any of these SIMNET databases into the DIS (geocentric) coordinate system and vice-versa, but additional transformations are made possible by the use of other conversion routines from the PT conversion library. Applicable files can found under the /xlat/coordinate_transformation directory.

Initialization parameters and constants for a variety of coordinate transformation constants are set up in the /xlat/coordinate_transformation/init_conversions.c file. Note that if changes to this must be performed, a new executable must be built.

The Protocol Translator only translates Simulation or Digital message Protocols. The translation process includes coordinate conversions, type transformations, angular transformations, articulated transformations, simple direct field copying and PDU mappings, among others. The primary modules which handle PDU translations are SIM_transform.c and DIS_transform.c, which can be found in /xlat/decode.

Typical SIM to DIS Translation Operation:

The Protocol Translator receives a SIMNET PDU packet from the designated network interface, validates the input PDU's Sub-Network Access Protocol (SNAP) header, checks if the input PDU is sent from the host itself then passes the PDU to the decoding process.

If the incoming user data is a SIMNET PDU, the decoding process will split up the incoming user data into individual Association Layer Protocol Datagram Units (ALPDU) since there may be multiple ALPDUs in one user datagram and validate the Association Layer header. In a normal operation, there is no Association Layer (AL) in DIS so that for a request ALPDU, the decoding process will call the transaction request/response service to write the responder site, responder simulator, and transaction ID into a shared memory. These shared memory contents become inputs to the transaction service routine on the DIS to SIM process which sends a response PDU back to the SIMNET network. The transaction service is not required when the Association Layer is chosen for SIM & DIS in the input parameter file, csrdf_xlat.dat (not normal operation). Refer to [4.1] Software Requirement Specification (SRS) for translated PDUs.

If filtering is requested, the input PDU's Exercise ID will be checked against the file parameter's Exercise ID (refer to section 4.3.2 - csrdf_xlat.dat for input parameters). If the Exercise ID is valid and the input PDU is either a Simulation or Digital message protocol, encoding functions will be called to separate out the encapsulated PDU and to translate the input PDU based on the PDU's message type.

Typical DIS to SIM Translation Operation:
The Protocol Translator receives a DIS datagram in the Internet Domain with UDP in connectionless mode, checks if the input PDU is sent from the host itself, then passes the datagram to the decoding process.

Within the decoding process, the input datagram Exercise ID will be checked against the file parameter's Exercise ID if filtering is requested (refer to section 4.3.2 - csrdf_xlat.dat) and encoding functions will be called to validate the datagram type and to translate the input datagram based on the datagram's message type. If the input datagram is a DIS Entity State PDU, the dead reckoning process will be performed (refer to 4.1.3).

An input PDU will not be translated for any of the following reasons:
1. input Network buffer is full
2. invalid SNAP header
    a. invalid Ethernet type
    b. invalid DSAP
    c. invalid SSAP
    d. invalid control
    c. invalid Protocol ID
3. PDU's source address is same as the host's source address (self-send message)

4. SIMNET Association Layer header:
    a. invalid ALPDU kind
    b. invalid data length
    c. invalid protocol (not Simulation or Digital Message)
5. Simulation or Digital Message Protocol:
    a. invalid Exercise ID
    b. insufficient bytes
    c. invalid PDU kind

4.1.3 Dead Reckoning

Both DIS and SIMNET support dead reckoning models intended to reduce the network traffic of entity/vehicle state information. However, DIS is capable of supporting a higher-fidelity dead reckoning model, which uses last known position, velocity and acceleration to predict an entity's movement. SIMNET uses only the last known position and velocity (no acceleration) to predict a vehicle's movement. The dead reckoning software in the Protocol Translator handles the difference between the two (2) algorithms by rebroadcasting updated SIMNET Vehicle Appearance PDUs as soon as a specific tolerance between the SIMNET dead reckoning algorithm and the DIS algorithm is exceeded (nominally three (3) degrees in orientation and one (1) meter in position), but this can be adjusted by changing applicable parameters in /xlat/dead_reckon/init_dead_reckoning.c.

When a DIS Entity State PDU is received, it is immediately translated into a SIMNET Vehicle Appearance PDU. If the DIS entity is supporting a dead reckoning model inherently different than the model SIMNET is capable of supporting, the Protocol Translator dead reckoning software begins handling the difference. Whenever a DIS Entity State PDU is translated, entity state information gets saved into a SIMNET vehicle state database. This first set of information can be thought of as the "time-zero" state of the vehicle. The dead reckoning software calculates the time at which the dead reckoning models will differ by significant tolerance between DIS and SIMNET. This time represents the time at which updated entity information must be made known to the SIMNET side, at which time a broadcast of an updated SIMNET Vehicle Appearance PDU should take place. The vehicle state at this time is then saved to the state database (representing a new "time-zero"), and a timer is then reset to countdown to the next time a new SIMNET update is required. This process of broadcasting after a certain fixed time continues until the next DIS Entity State PDU for an entity is received (at which time the whole process is started over again), or a "dead reckoning cease time" (nominally five (5) seconds) is exceeded (this ensures that if a DIS simulator goes offline that SIMNET broadcasts are eventually shut off, thereby preventing "infinite" updates).

Because the PT had only one (1) system interval timer available for the dead reckoning software, a linked list was maintained to hold the "future" identifiers of SIMNET vehicle identifiers to be updated by the PT dead reckoning software. This list is maintained by separate list insertion and list deletion routines which handle the insertion of vehicle identifiers into the linked list and the removal of vehicle identifiers from the linked list, respectively.

4.1.4 User Interface

The Protocol Translator (PT) uses three shared memories. Two of the shared memories are used for recording DIS to SIM and SIM to DIS translation statistics and the contents are used by the "Protocol Translator User Interface". The other shared memory is used for recording transaction request/response service within the Protocol Translator. See the following N**2 table:

	DIS Shared Memory	SIM Shared Memory	Transaction Shared Memory
PT User Interface	input data read from	input data read from	
PT's DIS to SIM translation	translation statistics logged		Transaction requests logged
PT's SIM to DIS translation		translation statistics logged	Transaction responses generated from

The transaction service is not required when the Association Layer is chosen for SIM & DIS in the input parameter file, csrdf_xlat.dat (not normal operation).

4.2 Development Environment Description

Development Tools:
Sun Sparc10 or Sun Sparc2 workstation

Operating System	- Sun Sparc2 SunOS Release 4.1.1 must support User Datagram Protocol - Sun Sparc10 SunOS Release 4.1.1 must support User Datagram Protocol
Environment	- UNIX
Language	- C
Compiler	- gcc version 1.40 The gcc's preprocessor may not work for "ioctl.h" and "ttychars.h". Use the standard C preprocessor.

4.2.1 Development Environment Directory Structure

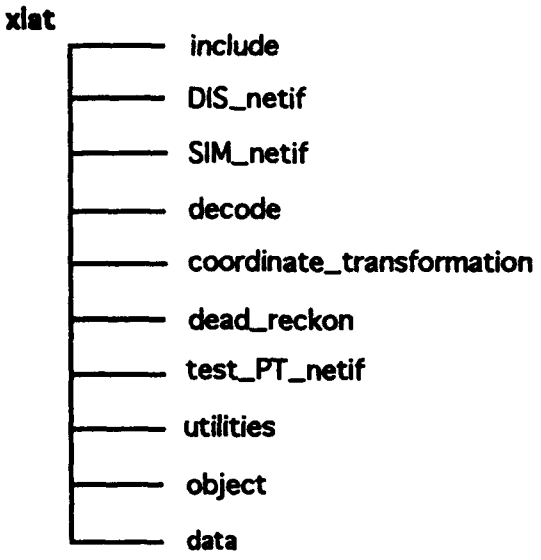

Figure 4.2.1 Protocol Translator Top Level Development Directory Structure

"xlat" is the main directory of Protocol Translator and it is where the Protocol Translator executable "xlat" resides. The following table describes each top level sub-directory:

Sub - directory	Description
include	header files
DIS_netif	source files for DIS network interface
SIM_netif	source files for SIM network interface
decode	source files for decoding and encoding SIM/DIS PDUs
coordinate_transformation	source files for coordinate transformation
dead_reckon	source files for dead reckoning
test_PT_netif	source files for Protocol Translator test utilities
utilities	source files for Protocol Translator utility software
object	object files created by gcc compiler
data	translation statistic output files.

4.2.2 Development Environment Build Configuration

Protocol Translator can be built on a Sun Sparc10 workstation, Sun Sparc2, or Silicon Graphic workstation. To build on a Silicon Graphic workstation, the compiler directive "-DSGI" must be included in the xlat/xlat.mak file; otherwise, the default condition is assumed (Sparc10 or Sparc2).

Compiler directives for building Protocol Translator with the xlat/xlat.mak file are:

Directive	Description
-DALT_FLUSH	use the alt_flush routine instead of the library routine "f_flush" for flushing a character buffer.
-DFUNC	compile the Network Interface for system build. Network Interface expects to receive and send PDU from Network Interface card.
-DSYSTEM	compile coordinate transformation routines for system build.

4.3 Runtime Environment Description

Protocol Translator resides entirely on a SPARC10 workstation. It receives PDUs from a designated network that is connected to one of the Ethernet cards on Sparc10. It outputs translated PDUs to the second designated network which is connected to the other Ethernet card.

Sparc10 workstation must be booted with the following changes in etc/rc.boot:
define the add-on Ethernet Interface - add:

```
        #set second Ethernet Interface card to Address Family "Ether"
        ifconfig le1 ether 8:0:20:12:74:82 -trailers private up
after:
        interface_names = " 'shcat /etc/hostname ...
        if test
        then
                {
                .....
                }
        fi
```

avoid self-send polling messages - comment out:

```
        #ifconfig -ad auto-revarp up
```

send system to private mode - add private in:

```
        ifconfig $1 " 'shcat /etc/hostname\. $1'" netmask+ -trailer private up
```

4.3.1 Runtime Environment Directory Structure

Protocol Translator executable resides in the "xlat" directory. User must be in superuser mode to access the Ethernet interface. To invoke Protocol Translator, type "xlat" in xlat directory.

4.3.2 Runtime Environment Data Files

Protocol Translator retrieves its runtime parameters from the following data files:

Data File	Description
xlat/csrdf_xlat.dat	created by : xlat/utilities/bin/create_params. It contains command arguments for running the Protocol Translator executable "xlat/xlat". Command arguments included: Type of translation: Simulation or Digital Message DIS Network Interface Card Assignment: le0 or le1 SIMNET will use the opposite of DIS DIS Network Interface selection: UDP/IP or Ethernet SIMNET always use raw Ethernet Interface DIS PDU with Association Layer: Yes/No SIM PDU with Association Layer: Yes/No Exercise Identification Number: 0 - to pass all Battle Scheme: 0 - other 1 - absolute 2 - relative *
xlat/info_0	created by: xlat/SIM_netif/get_sysaddr.c It contains the output of a system command "ifconfig le0" which displays the first Ethernet interface card's name, Internet and Ethernet address. info_0 is used for determining if an input PDU is sent from the same machine that is receiving it (to filter out self-send messages).
xlat/info_1	created by: xlat/SIM_netif/get_sysaddr.c It contains the output of a system command "ifconfig le1" which displays the first Ethernet interface card's name, Internet and Ethernet address. info_1 is used for determining if an input PDU is sent from the same machine that is receiving it (to filter out self-send messages).

xlat/FILE_EXTENSION	Protocol Translator records the current transformation statistic information in a file and saves it under xlat/data. The extension of this output file is updated for each Protocol Translator execution. The largest possible number of this extension is the possible value that an integer can hold (2**32 for a 32 bit integer). The last extension used is recorded in xlat/FILE_EXTENSION and this file is read by xlat/utilities/get_extension.c. If FILE_EXTENSION is deleted, extension on output file resets to 1.

* The typical setup: DIS Network Interface is le0
SIMNET Network Interface is le1
DIS uses UDP/IP
SIMNET uses raw Ethernet Interface
No Association Layer on DIS
Association Layer on SIMNET
Exercise ID assign by user. PT only accept PDUs with the assigned Exercise ID.
Battle Scheme value determines the mapping of the SIM Activate Request PDU.
(refer to section 4.1.2).

Protocol Translator also records the current transformation statistics in shared memories each time "logger" is called. These shared memory contents become input data for the Protocol Translator User Interface utility program. Refer to section 5 for more details on Protocol Translator User Interface.

The following are necessary coefficient data files that are used in init_conversion routine for coordinate transformation:

Coordinate Transformation Data File	Description
COEF10_Fulda_GE_to_geocentric	created by: xlat/coordinate_transformation/poly10.c contain data that is needed for GE flat-earth X.Y.Z to geocentric x,y,z (WGS 84 latitude and longtitude) conversion.
COEF10_Fulda_geocentric_to_GE	geocentric x,y,z to GE flat-earth X,Y,Z with (WGS 84 ellipsoid).
COEF10_Hunter_GE_to_geocentric	GE flat-earth X,Y,Z to geocentric x,y,z.
COEF10_Hunter_geocentric_to_GE	geocentric x,y,z to GE flat-earth X,Y,Z.

COEF10_Hunter_UTMWGS72_to_geocnetric	WGS 72 UTM X,Y,Z to geocentric x,y,z.
COEF10_Hunter_geocentric_to_UTMWGS72	geocentric x,y,z to WGS 72 UTM X,Y,Z.
COEF10_Hunter_UTMWGS84_to_geocentric	UTM WGS 84 X,Y,Z to geocentric x,y,z.
COEF10_Hunter_geocentric_to_UTMWGS84	geocentric x,y,z to UTM WGS84 X,Y,Z.
COEF10_Knox_UTMWGS84_to_geocentric	UTM WGS 84 X,Y,Z to geocentric x,y,z.
COEF10_Knox_geocentric_to_UTMWGS84	geocentric x,y,z to UTM WGS 84 X,Y,Z.
COEF4_Fulda_GE_to_geocentric	created by: xlat/coordinate_transformation/poly4.c GE flat-earth X,Y,Z to geocentirc x,y,z (WGS 84 ellipsoid).
COEF4_Fulda_geocentric_to_GE	geocentric x,y,z to GE flat-earth X,Y,Z (WGS 84 ellipsoid).
COEF4_Hunter_GE_to_geocentric	GE flat-earth X,Y,Z to geocentric x,y,z.
COEF4_Hunter_geocentric_to_GE	geocentric x,y,z to GE flat-earth X,Y,Z.
COEF4_Hunter_UTMWGS72_to_geocentric	UTM WGS 72 X,Y,Z to geocentric x,y,z.
COEF4_Hunter_geocentric_to_UTMWGS72	geocentric x,y,z to UTM WGS 72 X,Y,Z.
COEF4_Hunter_UTMWGS84_to_geocentric	UTM WGS84 X,Y,Z to geocentric x,y,z.
COEF4_Hunter_geocentric_to_UTMWGS84	geocentric x,y,z to UTM WGS 84 X,Y,Z.
COEF4_Knox_UTMWGS84_to_geocentric	UTM WGS84 X,Y,Z to geocentric x,y,z.
COEF4_knox_geocentric_to_UTMWGS84	geocentric x,y,z to UTM WGS 84 X,Y,Z.

The Following table lists data files that contain the data conversion origins for coordinate transformation process. Data files reside at xlat/coordinate_tranformation/data/ directory and are created by xlat/coordinate_transformation/conversion_driver.c. These data files are called by init_conversion.c, acc10.c, poly10.c and the test routine, rotation_driver.c.

Data File	Description
DATA_Fulda_GE_to_geocentric	Fulda GE x,y,z to geocentric x,y,z
DATA_Fulda_geocentric_to_GE	Fulda geocentric x,y,z to GE x,y,z
DATA_Hunter_GE_to_geocentric	Hunter GE x,y,z to geocentirc x,y,z

DATA_Hunter_geocentric_to_GE	Hunter geocentric x,y,z to GE x,y,z
DATA_Hunter_UTMWGS72_to_geocentric	Hunter UTM WGS72 x,y,z to geocentric x,y,z
DATA_Hunter_geocentric_to_UTMWGS72	Hunter geocentric x,y,z to UTM WGS72 x,y,z
DATA_Hunter_UTMWGS84_to_geocentric	Hunter UTM WGS84 x,y,z to geocentric x,y,z
DATA_Hunter_geocentric_to_UTMWGS84	Hunter geocentric x,y,z to UTM WGS84 x,y,z
DATA_Knox_UTMWGS84_to_geocentric	Knox UTM WGS84 x,y,z to geocentric x,y,z
DATA_Knox_geocentric_to_UTMWGS84	Knox geocentric x,y,z to UTM WGS84 x,y,z

Other test data files that are generated by init_conversion:

Data File	Description
DATA_Fulda_geodetic_to_GE	Fulda geodetic lat, long to GE x,y
DATA_Fulda_GE_to_geodetic	Fulda GE x,y to geodetic lat, long
DATA_Fulda_geodetic_to_UTM	Fulda geodetic lat, long to UTM x,y
DATA_Fulda_UTM_to_geodetic	Fulda UTM x,y to geodetic lat, long
DATA_Fulda_geodetic_to_geocentric	Fulda geodetic lat, long, elevation to geocentric x,y,z
DATA_Fulda_geocentric_to_geodetic	Fulda geocentric x,y,z geodetic lat, long, elevation
DATA_Hunter_geodeticWGS84_to_geodeticWGS72	Hunter WGS 84 geodetic to WGS 72 geodetic
DATA_Hunger_geodeticWGS72_to_geodeticWGS84	Hunter WGS 72 geodetic to WGS 84 geodetic
DATA_Hunter_geodeticWGS72_to_UTMWGS72	Hunter WGS 72 geodetic lat, long to WGS 72 UTM x, y
DATA_Hunter_UTMWGS72_to_geodeticWGS72	Hunter WGS 72 UTM x,y to WGS 72 geodetic lat, long
DATA_Hunter_geodetic_to_GE	Hunter geodetic lat, long to GE x,y
DATA_Hunter_GE_to_geodetic	Hunter GE x,y, to geodetic lat, long
DATA_Hunter_geodeticWGS84_to_UTMWGS84	Hunter WGS 84 geodetic lat, long to WGS 84 UTM x,y.
DATA_Hunter_UTMWGS84_to_geodeticWGS84	Hunter WGS 84 UTM x, y to WGS 84 geodetic lat, long
DATA_Knox_geodeticWGS84_to_UTMWGS84	Knox WGS 84 geodetic lat, long to WGS 84 UTM x,y
DATA_knox_UTMWGS84_to_geodeticWGS84	Knox WGS 84 UTM x,y to WGS 84 geodetic lat, long.

4.3.3 Runtime Environment Startup Parameters

Protocol Translator's executable "xlat/xlat" reads command arguments from xlat/csrdf_xlat.dat. If xlat/csrdf_xlat.dat does not exist, the user must run xlat/utilities/bin/create_params to create the csrdf_xlat.dat file. The csrdf_xlat.dat file only needs to be created once if no startup parameter update is needed. Refer to 4.3.2 for more detail on csrdf_xlat.dat.

4.3.4 Runtime Protocol Translator User Interface Commands

Enter "xlat/xlat" from keyboard to execute the Protocol Translator.

4.4 Startup Procedure

4.4.1 Protocol Translator Startup Procedure:

Window	Action
	login to a Sun Sparc10 workstation as root, superuser.
	type "openwin"
	create two windows with size equals to half of the screen (vertically) leaving an inch and half space for the shrunk console window.
1	invoke xlat/utilities/bin/create_params to create runtime command argument to the data file "xlat/csrdf_xlat.dat".
1	xlat/data/ directory must exists to store conversion statistic output files.
1	invoke xlat/xlat to start the Protocol Translator.
1	type "ctrl-C" to quit Protocol Translator.

4.4.1 Protocol Translator User Interface Startup Procedure:

Window	Action
2	If desired, the user may execute xlat/utilities/pt_userIF "Protocol Translator User Interface" to view on-line statistics. It may also be used to view the last executed Protocol Translator statistic.

2	A Protocol Translator User Interface main menu is displayed. Select the desired funtion by typing in the corresponding option number. 1. Display on-line Statistics 2. Reset logger counter 3. Shutdown all Shared Memories 4. Quit PT User Interface
2	If "1" is selected, an " on-line Statistics menu" is displayed. Select one of the options by typing in the corresponding option number. 1. SIM Statistics 2. SIM - Last 100 Error Codes 3. DIS Statistics 4. DIS - Last 100 Error Codes 5. SIM and DIS Statistics 6. Exit Statistics Menu 1 - displays the on-line SIM to DIS Statistics page. 2 - displays the last 100 logger calls. 3 - displays the on-line DIS to SIM Statistics page. 4 - displays the last 100 logger calls. 5 - displays the on-line SIM to DIS and DIS to SIM Statistics page. 6 - returns to Main Menu.
2	*If "2" is selected from the main menu, all logger counters in the SIM and DIS shared memories will be reset to zero. *If "3" is selected from the main menu, the DIS and SIM shared memories will be closed. If "4" is selected from the main menu, exits the Protocol Translator User Interface. * The <pause> symbol will appear after the action of "2" or "3". Hit the "return" key to return to main menu.

5. Utility Software

Utility software are stored in xlat/utilities directory. Each of these utilities contributes to the success of Protocol Translator. To invoke any of these utilities, type in their executable names. List of utility software:

Executable	Description
create_params	compile: "make -f xlat/xlat.mak create_params" creates the csrdf_xlat.dat for storing the Protocol Translator input parameters. csrdf_xlat.dat is read by xlat.
shm_shutdown	compile: "make -f xlat/utilities/pt_userIF.mak shm_shutdown" closes the existing shared memories that are used for storing Protocol Translator statistics.
pt_userIF	compile: "make -f xlat/utilities/pt_userIF.mak" displays the DIS and SIM translation statistics on screen. The statistics included are types of PDU errors, receive and send rates, number of PDUs received and sent, number of entity state or vehicle appearance PDUs received, and number of rebroadcast PDUs (dead reckoning). The Protocol Translator User Interface's runtime commands are displayed in listing format. User enters desired selection and hit "carriage return" to activate a command. In case of a <pause>, hit "carriage return" key to display the last selection menu. Enter "ctrl-C" to exit a statistic display. Protocol Translator User Interface must be run on a defined window which is approximately half the size of the console.
ho	compile: "xlat/utilities/make" displays a hexadecimal in other formats: float, integer, and double.
hd	compile: "xlat/utilities/make" displays a hexadecimal in double format.

io	compile: "xlat/utilities/make" displays an integer in other formats: float, double, and hex.
fo	compile: "xlat/utilities/make" displays a float in other formats: hex, double, and integer.
do	compile: "xlat/utilities/make" displays a double in other formats: hex, float, and integer.

Utility software in xlat/coordinate_transformation directory:

Executable	Description
bounds	compile: "xlat/coordinate_transformation/make" a driver routine to determine the bounds of a conversion data file.
poly10	compile: "xlat/coordinate_transformation/make" calculates the coefficients of a polynomial using a least-squares technique. output data filename: "COEF10_*conversion routines*".

The following table lists test utilities that are stored in xlat/test_PT_netif directory and are used as aids for testing the Protocol Translator. To compile, type "xlat/test_PT_netif/make".

Executable	Description
dis_test_send	sends a DIS PDU onto a DIS network.
dis_test_rec	receives a DIS PDU from a DIS network and displays its content on screen.
sim_test_send	sends a SIMNET PDU onto a SIMNET network.
sim_test_rec	receives a SIMNET PDU from a SIMNET network and displays its content on screen.
dis_test_send_multiple	sends a DIS PDU onto a DIS network forever.
sim_test_send_multiple	sends a SIMNET PDU onto a SIMNET network forever.

There are test software and compiler directives that are used to test individual modules:

Directory / Executable	Description
xlat	
dst	compile: "make -f xlat.mak dst" tests the DIS to SIM translation. No association layer transaction request/response service is involved.
sdt	compile: "make -f xlat.mak sdt" tests the SIM to DIS translation. No association layer transaction request/response service is involved.
DIS_netif / SIM_netif	set compiler directive to -MODULE_TEST instead of -FUNC
decode	
PDU_size	compile: "gcc xlat/decode/PDU_size.c -o -h xlat/include PDU_size" checks the size of PDU structures to ensure the compiler has not added any extra bytes. Checks SIM, DIS, and DMC PDUs.
dst_test	compile: "make -f xlat/decode/dst_test.mak" tests the decode modules for DIS to SIM translation. For individual task, transaction request/response service will not be included.
sdt_test	compile: "make -f xlat/decode/sdt_test.mak" tests the decode modules for SIM to DIS translation. For individual task, transaction request/response service will not be included.
coordinate_transformation	compile: "xlat/coordinate_transformation/make"
conversion_driver	a driver routine to interactively test the various conversion routines.
matrix_driver	a driver routine to interactively use and test the rotation matrix to yaw, pitch, roll conversion software.
rotation_driver	a driver routine to interactively test the rotation routines.
rotation_matrix_test	tests the ypr_to_rotation_matrix.c and rotation_matrix_to_ypr.c routines.
velocity_driver	a driver routine to interactively test the velocity routines.

acc10	performs accuracy check on conversion routines. Output filename in data/: ACC10_*conversion routine*"
dead_reckon	compile: "xlat/dead_reckon/make"
dr.c	tests various aspects of the linked list software. This driver tests both list insertion and deletion.
drtest.c	performs beta testing on dead reckoning routines.

PDU_builder is also used to create SIMNET and DIS PDUs as input to the Protocol Translator to aid in the debugging process.

6. Notes

List of Acronyms:

ADST	Advanced Distributed Simulation Technology
AL	Association Layer
APDU	Association Protocol Data Units (Association Layer)
CSRDF	Crew Station Research and Development Facility
DIS	Distributed Interactive Simulation
LLC	Logical Link Control
PDU	Protocol Data Units
PT	Protocol Translator
SIMNET	Simulation Networking
SNAP	Sub-Network Access Protocol
WGS	World Geodetic System
UDP/IP	User Datagram Protocol/Internet Protocol
UTM	Universal Transverse Mercator