

# SIEMENS

## SIMATIC TIWAY

### Host Software for PC

User Manual

Order Number: PPX:TIWAY-8108-3  
Manual Assembly Number: 2587871-0008  
Third Edition

 **DANGER**

**DANGER** indicates an imminently hazardous situation that, if not avoided, will result in death or serious injury.

**DANGER** is limited to the most extreme situations.

 **WARNING**

**WARNING** indicates a potentially hazardous situation that, if not avoided, could result in death or serious injury, and/or property damage.

 **CAUTION**

**CAUTION** indicates a potentially hazardous situation that, if not avoided, could result in minor or moderate injury, and/or damage to property.

**CAUTION** is also used for property-damage-only accidents.

**Copyright 1996 by Siemens Energy & Automation, Inc.  
All Rights Reserved — Printed in USA**

Reproduction, transmission, or use of this document or contents is not permitted without express consent of Siemens Energy & Automation, Inc. All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

Since Siemens Energy & Automation, Inc., does not possess full access to data concerning all of the uses and applications of customer's products, we do not assume responsibility either for customer product design or for any infringements of patents or rights of others which may result from our assistance.

## MANUAL PUBLICATION HISTORY

SIMATIC TIWAY Host Software for PC User Manual

Order Manual Number: PPX: TIWAY-8108-3

*Refer to this history in all correspondence and/or discussion about this manual.*

---

<b>Event</b>	<b>Date</b>	<b>Description</b>
Original Issue	08/85	Original Issue (2494061-0001)
Second Edition	04/86	Second Edition (2494061-0002)
Third Edition	05/96	Third Edition (2804790-0001) (combined 2494061 and 2601431 texts into one manual)

## LIST OF EFFECTIVE PAGES

---

Pages	Description	Pages	Description
Cover/Copyright	Third		
History/Effective Pages	Third		
iii — xvi	Third		
1-1 — 1-26	Third		
2-1 — 2-5	Third		
3-1 — 3-30	Third		
4-1 — 4-17	Third		
5-1 — 5-5	Third		
6-1 — 6-33	Third		
7-1 — 7-39	Third		
8-1 — 8-3	Third		
9-1 — 9-8	Third		
10-1 — 10-61	Third		
A-1 — A-2	Third		
B-1	Third		
C-1 — C-4	Third		
D-1 — D-16	Third		
E-1 — E-14	Third		
F-1 — F-10	Third		
G-1 — G-5	Third		
H-1 — H-2	Third		
Index-1 — Index-4	Third		
Registration	Third		

# Contents

---

## Preface

## Chapter 1 Installation

<b>1.1</b>	<b>Overview</b> .....	<b>1-2</b>
	Background Requirements .....	1-2
	Hardware Requirements .....	1-2
	Software Requirements .....	1-3
	Customer Support .....	1-3
<b>1.2</b>	<b>PC Software Package Files</b> .....	<b>1-4</b>
	Operation .....	1-6
<b>1.3</b>	<b>Software Installation and Operation</b> .....	<b>1-7</b>
	Backing Up Your Software .....	1-7
	Installation .....	1-8
<b>1.4</b>	<b>Using TIWAY</b> .....	<b>1-11</b>
<b>1.5</b>	<b>Using a Compiled Language</b> .....	<b>1-12</b>
<b>1.6</b>	<b>Program Linking</b> .....	<b>1-13</b>
	Object Code Files .....	1-13
	Linking on the PC .....	1-13
	Link Options .....	1-17
	Lattice C .....	1-19
	Compiling with Lattice C .....	1-20
<b>1.7</b>	<b>Executing in Interpreted BASIC</b> .....	<b>1-21</b>
	Getting Started .....	1-21
	Typical BASIC Startup .....	1-22
	Calling Subroutines in BASIC .....	1-23
	Subroutine Calling Parameters .....	1-24
	Passing Parameters to Subroutines .....	1-24
<b>1.8</b>	<b>Named Tag Table Maintenance</b> .....	<b>1-25</b>
	Creating the Named Tag Table .....	1-25
	Accessing a Named Tag Specification .....	1-26

## Chapter 2 Overview

<b>2.1</b>	<b>TIWAY System Characteristics</b> .....	<b>2-2</b>
<b>2.2</b>	<b>Host Software Functions</b> .....	<b>2-3</b>
	Levels of Communication .....	2-3
	Languages Supported .....	2-5
	Data Format Conversion .....	2-5

---

## Chapter 3 TIWAY Subroutine – Calling Arguments

3.1	Overview .....	3-2
3.2	Reference List of Subroutines and Arguments Reference List .....	3-3
3.3	Reference List of Arguments .....	3-6
3.4	List of Data Elements Types .....	3-14
3.5	<b>Addressing</b> .....	<b>3-17</b>
	Address Elements .....	3-17
	Highway Selection .....	3-17
	Secondary Selection .....	3-17
	Data Element Type Selection .....	3-17
	Data Element Location Selection .....	3-18
	Addressing Arguments .....	3-18
	Address Specification .....	3-18
	Tag Name Specification .....	3-19
	ASCII Specification .....	3-19
	Binary Specification .....	3-20
3.6	<b>Buffer Specification</b> .....	<b>3-21</b>
	Command Buffers .....	3-21
	Response Buffers .....	3-21
	Read Buffers .....	3-22
	Write Buffers .....	3-22
3.7	<b>Data Block Manipulation</b> .....	<b>3-23</b>
	UCL Instruction Code (cc Argument) .....	3-23
	Block List (cclst Argument) .....	3-23
	Number of Blocks (nblk Argument) .....	3-24
	Number of Data Element Locations (nnnn Argument) .....	3-24
	Number of Data Elements List (nnnnlst Argument) .....	3-24
	Pattern (pattern Argument) .....	3-24
	CIM Command Code Specifier (ss Argument) .....	3-24
	Tag List (taglst Argument) .....	3-24
3.8	<b>Diagnostics</b> .....	<b>3-25</b>
3.9	<b>Masks</b> .....	<b>3-26</b>
3.10	<b>Network Access</b> .....	<b>3-27</b>
3.11	<b>Status Determination</b> .....	<b>3-28</b>
	Composite Error Reporting (istat Argument) .....	3-28
	Error Type (errtyp Argument) .....	3-29
	CIM Status (cstat Argument) .....	3-29
	Secondary Status (sstat Argument) .....	3-30

---

## Chapter 4 TIWAY Interface Subroutines – Topical

4.1	Overview .....	4-2
4.2	Session Control Subroutines .....	4-5
4.3	Computer Port Setup Initialization .....	4-6
	INIT Subroutine Initialization .....	4-6
	Call Format .....	4-6
	Notes on Call Format .....	4-6
4.4	Host Adapter Command Code Subroutines .....	4-7
	ACTVAT .....	4-8
	ADIAG .....	4-8
	BRDCST .....	4-8
	DEACT .....	4-9
	POLL .....	4-9
	SDIAG .....	4-9
	SECLOG .....	4-10
	XPAR .....	4-10
4.5	TIWAY Primitive Subroutines .....	4-11
	CHNGST .....	4-12
	CONFIG .....	4-12
	DEFBLK .....	4-12
	FILL .....	4-12
	GATHER .....	4-12
	GETLEN .....	4-13
	NATIVE .....	4-13
	RDSTS .....	4-13
	TIGET .....	4-13
	TIPUT .....	4-13
	TIREAD .....	4-14
	TIWRIT .....	4-14
	WRBUF .....	4-14
	WRTGAT .....	4-14
4.6	CIM Functional Command Subroutines .....	4-15
	CCUSTS .....	4-16
	CIMDNL .....	4-16
	CIMRD .....	4-16
	CIMUPL .....	4-16
	CIMWR .....	4-16
	RDLOOP .....	4-16
	RNDRD1 .....	4-16
	RNDRD2 .....	4-17
	RNDRD3 .....	4-17
	RNDRD4 .....	4-17

---

## Chapter 5 TIWAY Support Routines

5.1	Overview .....	5-2
5.2	Descriptions of Support Subroutines .....	5-4
	BLDMSK Subroutine .....	5-4
	GETMSG Subroutine .....	5-4
	HST2TI Subroutine .....	5-4
	LKUFMT Subroutine .....	5-4
	LKUTGL Subroutine .....	5-4
	LKUTGS Subroutine .....	5-4
	PUTMSG Subroutine .....	5-4
	T12HST Subroutine .....	5-4
	TIXTN and TIXTNW Subroutines .....	5-5

## Chapter 6 File Transfer Subroutines

6.1	Overview .....	6-2
	Partial Memory Transfers .....	6-2
	Transfers Between Different Types of Devices .....	6-4
	Generic Upload/Download .....	6-4
	Specific Segment Transfers .....	6-5
6.2	Subroutine Descriptions .....	6-6
	UPLOAD .....	6-6
	DNLOAD .....	6-6
	Upload .....	6-6
	Download .....	6-7
6.3	Transfer File Descriptions .....	6-8
	Transfer File Name Specification .....	6-8
	Transfer File Construction .....	6-8
	Transfer File Formats .....	6-8

## Chapter 7 Interactive Operator Utilities

7.1	Using TIUSER and TIPROG .....	7-2
7.2	TIUSER Utility .....	7-3
	ACTIVATE Command .....	7-5
	CHANGESTATE Command .....	7-6
	DEACTIVATE Command .....	7-7
	DOWNLOAD Command .....	7-8
	FILESTATUS Command .....	7-11
	LIST Command .....	7-11
	STATISTICS Command .....	7-11
	UPLOAD Command .....	7-12
	VERIFY Command .....	7-15
	QUIT Command .....	7-15



---

<b>7.3</b>	<b>TIPROG Utility</b> .....	<b>7-16</b>
	Session Control Commands .....	7-20
	Host Adapter Commands .....	7-21
	Base HIU Commands .....	7-21
	TIWAY Primitive Commands .....	7-26
	CIM Functional Commands .....	7-32
	Support Commands .....	7-37
 <b>Chapter 8 Tag Table</b>		
<b>8.1</b>	<b>Defining Tag Names</b> .....	<b>8-2</b>
<b>8.2</b>	<b>Building the Tag Table</b> .....	<b>8-3</b>
 <b>Chapter 9 Network Autoconfiguration File</b>		
<b>9.1</b>	<b>Introduction</b> .....	<b>9-2</b>
<b>9.2</b>	<b>Network Autoconfigure File</b> .....	<b>9-3</b>
	Template Record .....	9-3
	Data Record .....	9-3
	Comments .....	9-7
	Record Editing .....	9-7
<b>9.3</b>	<b>Network Autoconfiguration Process</b> .....	<b>9-8</b>
 <b>Chapter 10 Subroutine Library</b>		
<b>10.1</b>	<b>ACTVAT</b> .....	<b>10-4</b>
<b>10.2</b>	<b>ADIAG</b> .....	<b>10-5</b>
<b>10.3</b>	<b>BLDMSK</b> .....	<b>10-7</b>
<b>10.4</b>	<b>BRDCST</b> .....	<b>10-8</b>
<b>10.5</b>	<b>CCUSTS</b> .....	<b>10-10</b>
<b>10.6</b>	<b>CHNGST</b> .....	<b>10-11</b>
<b>10.7</b>	<b>CIMDNL</b> .....	<b>10-12</b>
<b>10.8</b>	<b>CIMRD</b> .....	<b>10-14</b>
<b>10.9</b>	<b>CIMUPL</b> .....	<b>10-15</b>
<b>10.10</b>	<b>CIMWR</b> .....	<b>10-16</b>
<b>10.11</b>	<b>CONFIG</b> .....	<b>10-17</b>
<b>10.12</b>	<b>DEACT</b> .....	<b>10-18</b>
<b>10.13</b>	<b>DEFBLK</b> .....	<b>10-20</b>

---

10.14	DNLOAD	10-22
10.15	FILL	10-24
10.16	FIN	10-25
10.17	GATHER	10-26
10.18	GETLEN	10-27
10.19	GETMSG	10-28
10.20	HST2TI	10-29
10.21	INIT	10-30
10.22	LKUFMT	10-31
10.23	LKUTGL	10-32
10.24	LKUTGS	10-33
10.25	NATIVE	10-34
10.26	POLL	10-35
10.27	PUTMSG	10-36
10.28	RDLOOP	10-37
10.29	RDSTS	10-38
10.30	RNDRD1	10-40
10.31	RNDRD2	10-41
10.32	RNDRD3	10-42
10.33	RNDRD4	10-44
10.34	SDIAG	10-45
10.35	SECLOG	10-46
10.36	TI2HST	10-47
10.37	TIGET	10-48
10.38	TIPUT	10-50
10.39	TIREAD	10-51
10.40	TIWRIT	10-52

---

10.41	TIXTN and TIXTNW .....	10-53
10.42	UPLOAD .....	10-54
10.43	WRBUF .....	10-56
10.44	WRTGAT .....	10-58
10.45	XPAR .....	10-60
<b>Appendix A Host Adapter Command Codes .....</b>		<b>A-1</b>
<b>Appendix B TIWAY Primitives .....</b>		<b>B-1</b>
<b>Appendix C CIM Functional Command Codes .....</b>		<b>C-1</b>
<b>Appendix D PM550 CCU Task Codes .....</b>		<b>D-1</b>
<b>Appendix E Error Listings</b>		
E.1	Facility 1: TIWAY Subroutine Library Exceptions .....	E-2
E.2	Facility 2: Network Exceptions .....	E-5
E.3	Facility 3: Host Adapter Internal Exceptions .....	E-6
E.4	Facility 4: Host Adapter Exceptions .....	E-8
E.5	Facility 5: TIWAY Primitive Exceptions .....	E-9
E.6	Facility 6: Operating System Status Exceptions .....	E-11
E.7	Facility 7: I/O Status Exceptions .....	E-12
E.8	Facility 8: CIM Exceptions .....	E-13
E.9	Facility 9: Native Task Codes .....	E-14
<b>Appendix F Programmer's Notes for PC</b>		
F.1	General Information .....	F-2
F.2	Pascal .....	F-5
F.3	BASIC .....	F-6
F.4	C .....	F-7
F.5	Fortran .....	F-8
F.6	Linking Programs with the TIWAY Subroutine Library .....	F-9
F.7	Using Upload and Download from an Applications Program .....	F-10

---

Appendix G	500/505 and Host Computer Data .....	G-1
Appendix H	Unilink Dipswitch Reference Card .....	H-1

## List of Figures

---

2-1	Network Communication Levels .....	2-3
3-1	Fortran Declarations .....	3-8
3-2	Pascal Declarations .....	3-9
3-3	C Declarations .....	3-13
6-1	5TI Upload File Format .....	6-10
6-2	5TI Upload File Example .....	6-10
6-3	520/530 Upload File Format .....	6-11
6-4	530 Upload File Example .....	6-12
6-5	520C/530C Upload File Format .....	6-13
6-6	520C Upload File Example .....	6-14
6-7	560/565 Non-extended Upload File Format .....	6-15
6-8	565 Non-extended Upload File Example .....	6-17
6-9	560/565 Extended Upload File Format .....	6-21
6-10	565 Extended Upload File Example .....	6-23
6-11	560/565 Generic Upload File Format .....	6-27
6-12	560/565 Generic Upload File Example .....	6-28
6-13	PM550 and IT-160 MRCU Upload File Format .....	6-29
6-14	PM550 Upload File Example .....	6-29
6-15	Unilink Generic Upload File Format .....	6-30
6-16	Unilink Generic Upload File Example .....	6-31
6-17	Unilink Specific Segment Upload File Format .....	6-32
6-18	Unilink Specific Segment Upload File Example .....	6-33
7-1	TIUSER Help Menu Screen .....	7-4
7-2	TIPROG Help Menu .....	7-19
8-1	Sample Tag Table .....	8-3
9-1	Network Autoconfiguration File Template Record Format .....	9-3
9-2	Network Autoconfiguration File Data Record Format .....	9-3
9-3	HWY Record Format .....	9-4
9-4	CNFGHA Record Format .....	9-4
9-5	CFGHIU and ALCHBW Record Formats .....	9-5
9-6	ALCNMB and CNFGNM Record Formats .....	9-6
9-7	END Record Format .....	9-7
9-8	EOF Record Format .....	9-7

## List of Tables

---

1-1	PC Software Package Files .....	1-4
1-2	Object Files .....	1-18
3-1	Subroutines and Arguments Reference List .....	3-3
3-2	Argument Reference List for Fortran and Pascal .....	3-6
3-3	Argument Reference List for BASIC and C/C++ .....	3-10
3-4	Data Element Types .....	3-14
3-5	Accessing the Network with xtn .....	3-27
3-6	Facility Error Numbers .....	3-29
3-7	Seven Possible Subroutine Values .....	3-30
4-1	Host Adapter Command Code Subroutines .....	4-7
4-2	TIWAY Primitive Subroutines .....	4-11
4-3	CIM Functional Command Code Subroutines .....	4-15
5-1	Reference List of Support Subroutines .....	5-3
6-1	Partial Memory Transfer Codes .....	6-2
6-2	Memory Selectable for Upload .....	6-3
6-3	Memory Selectable for Download .....	6-3
6-4	Allowable Cross-Device Memory Transfers .....	6-4
6-5	Memory Selectable for Generic Upload/Download .....	6-5
7-1	Using TIUSER and TIPROG .....	7-2
7-2	Alphabetized List of TIUSER Commands .....	7-3
7-3	Alphabetized List of TIPROG Commands with Subroutines and Primitives .....	7-17
A-1	Host Adapter Command Codes .....	A-1
B-1	TIWAY Primitives (Universal Command Language) .....	B-1
C-1	CIM Functional Command Codes .....	C-1
D-1	CCU Task Codes for PM550 .....	D-1
E-1	Facility 1: TIWAY (Internal to TIWAY Subroutine Library Errors) .....	E-2
E-2	Facility 2: NETEXCEPT (Host Adapter—Network Exception Errors) .....	E-5
E-3	Facility 3: INTERNEXCP (Host Adapter—Adapter Internal Exception Errors) .....	E-6
E-4	Facility 4: HOSTEXCEPT (Host Adapter—Host Exception Errors) .....	E-8
E-5	Facility 5: PRIMITIVE (TIWAY Primitive Errors) .....	E-9
E-6	Facility 7: I/O Status Error Listings .....	E-12
E-7	Facility 8: CIM EXCEPT (CIM Functional Command Errors) .....	E-13

# Preface

---

The SIMATIC® TIWAY™ Host Software Package for personal computers (PCs) presents a unified interface between a personal computer as a host computer and the TIWAY Communications Network (through the SIMATIC® Unilink™ Host Adapter). This manual describes the TIWAY Host Software conventions. It is a programmer's reference guide; before using the TIWAY Host Software Package, you should be familiar with the host computer and its operating system, the Unilink Host Adapter, and the TIWAY Communications Network. Host-specific information is not included in this manual.

The manual describes the installation and use of the software, and includes programming notes for compiling and linking with the TIWAY library as well as information on PLC and Host Computer Data Formats.

You need to be familiar with MS-DOS® commands and operations before you install the software.

If you are not familiar with DOS, or if you do not understand how to connect devices via a serial communications (com) port to your personal computer, you may want to contact your PC consultant for installation assistance.

## Introduction to the User Manual

The purpose of this manual is to provide the applications programmer with the information necessary to create programs that communicate with the TIWAY network. The TIWAY Host Software Package consists of the following components:

- TIWAY Subroutine — Calling Arguments (See Chapter 3)

Because many of the calling arguments are common to most of the TIWAY subroutines, the calling arguments are discussed in a separate chapter. Included are tables that can be used as quick reference to arguments used by each subroutine, calling conventions, and data element types.

- TIWAY Interface Subroutines — Topical (See Chapter 4)

The TIWAY Subroutine Library, described later in this manual, encompasses all of the commonly used Host Adapter Command Codes, NIM primitives, and CIM Functional Commands.

- TIWAY Support Subroutines (See Chapter 5)

The TIWAY Support Subroutine Library, described later in this manual, provides a wide range of host computer services including logical to physical address conversion and data format conversion.

- File Transfer Subroutines (See Chapter 6)

- 
- Interactive Operator Utilities (See Chapter 7)

Two Interactive Operator Utilities, described later in this manual, provide generalized access to the facilities available through the TIWAY libraries. The purpose of the TIUSER Utility is to provide a shortcut to selected network management functions (e.g., connecting or disconnecting Secondaries). The primary purpose of the TIPROG Utility is to provide an interactive programmer interface to the Host Software. As such, TIPROG serves as a useful tool for learning the applications before coding them and is useful for short ad hoc requests for information or supervisory control.

Because many of the calling arguments are common to most of the TIWAY subroutines, the calling arguments are discussed in a separate chapter. Tables are included that can be used as a quick reference to the arguments used by each subroutine, their calling conventions, and the data element types. See Chapter 3.

The TIWAY Subroutine Library encompasses all of the commonly used Host Adapter Command Codes, TIWAY Primitives, and CIM Functional Commands. See Chapter 4.

The TIWAY Support Subroutine Library provides a wide range of host computer services including logical to physical address conversion and data format conversion. See Chapter 5.

The File Transfer Routines provide a way to upload and download programs and data to and from the Secondaries through a user-written application program written in any of the supported languages. See Chapter 6.

Two Interactive Operator Utilities provide generalized access to the facilities available through the TIWAY libraries. The purpose of the TIUSER utility is to provide a shortcut to selected network management functions (e.g., connecting or disconnecting Secondaries). The primary purpose of the TIPROG utility is to provide an interactive programmer interface to the TIWAY Subroutine Library. As such, TIPROG serves as a useful tool for learning about TIWAY subroutines. The utility also allows the programmer to test TIWAY applications before coding them and is useful for short requests for information or supervisory control. See Chapter 7.

A symbolic means of referencing address specifications, called tag names, is defined within a text file called the tag table. A means of automatically configuring the Unilink Host Adapters on a TIWAY network, called the Network Autoconfiguration File, is also described. See Chapter 9.



---

Eight appendices are provided. They are:

- Host Adapter Command Codes
- TIWAY Primitives
- CIM Functional Command Codes
- SIMATIC® PM550™ CCU Task Codes
- Error Listings
- Programmer's Notes for PC
- 500/505 and Host Computer Data
- Unilink Dipswitch Reference Card

**Related  
Publications**

Several related publications that are available from Siemens Energy & Automation, Inc., should be used in addition to this manual. These publications are listed below.

<b>Manual Title</b>	<b>Manual Number</b>
<i>SIMATIC Unilink Host Adapter User Manual</i>	PPX:TIWAY-8121
<i>SIMATIC 505 TIWAY I Network Interface Module (NIM) User Manual</i>	PPX:TIWAY-8124-x
<i>Unilink Universal Network Adapter Installation and Operation Manual</i>	PPX:TIWAY-8106

---

Several other related publications are available that may also be helpful in addition to those listed above.

<b>Manual Title</b>	<b>Manual Number</b>
<i>SIMATIC TIWAY I Systems Manual</i>	PPX:TIWAY-8101
<i>SIMATIC 545/555 Systems Manual</i>	PPX:545/555-8101-x
<i>SIMATIC 560/565 User Manual</i>	PPX:560/65-8101
<i>SIMATIC 520C/530C Programmable Controller User Manual</i>	PPX:530-8107
<i>SIMATIC 505 Programming Reference Manual</i>	PPX:505-8104-x
<i>SIMATIC 530 Programmable Controller User Manual</i>	PPX:530-8101
<i>SIMATIC 520 User Manual</i>	PPX:520-8101
<i>PM550 Network Interface Module User Manual</i>	PPX:550-8110
<i>PM550 CIM Manual</i>	PPX:550-8105
<i>PM550 System Manual</i>	PPX:550-8109
<i>SIMATIC® 5TI™ Programming Manual</i>	PPX:5TI-8101
<i>SIMATIC 5TI NIM User Manual</i>	PPX:5TI-8105
<i>Micro Remote Control Unit User Manual, PPX:IT-160</i>	2491106

**Technical Assistance**

If you need additional help, contact your Siemens distributor or sales agent. If you need assistance in contacting your distributor or sales office in the United States, call 800-964-4114. If additional technical assistance is needed, call the Technical Services Group in Johnson City, Tennessee, at 423-461-2522.

# Chapter 1

## Installation

---

<b>1.1</b>	<b>Overview</b> .....	<b>1-2</b>
	Background Requirements .....	1-2
	Hardware Requirements .....	1-2
	Software Requirements .....	1-3
	Customer Support .....	1-3
<b>1.2</b>	<b>PC Software Package Files</b> .....	<b>1-4</b>
	Operation .....	1-6
<b>1.3</b>	<b>Software Installation and Operation</b> .....	<b>1-7</b>
	Backing Up Your Software .....	1-7
	Installation .....	1-8
<b>1.4</b>	<b>Using TIWAY</b> .....	<b>1-11</b>
<b>1.5</b>	<b>Using a Compiled Language</b> .....	<b>1-12</b>
<b>1.6</b>	<b>Program Linking</b> .....	<b>1-13</b>
	Object Code Files .....	1-13
	Linking on the PC .....	1-13
	Link Options .....	1-17
	Lattice C .....	1-19
	Compiling with Lattice C .....	1-20
<b>1.7</b>	<b>Executing in Interpreted BASIC</b> .....	<b>1-21</b>
	Getting Started .....	1-21
	Typical BASIC Startup .....	1-22
	Calling Subroutines in BASIC .....	1-23
	Subroutine Calling Parameters .....	1-24
	Passing Parameters to Subroutines .....	1-24
<b>1.8</b>	<b>Named Tag Table Maintenance</b> .....	<b>1-25</b>
	Creating the Named Tag Table .....	1-25
	Accessing a Named Tag Specification .....	1-26

This chapter describes the installation and machine-dependent features of the SIMATIC TIWAY Host Software package developed for the personal computer (PC). Topics include:

- Making a backup copy of the TIWAY Host Software.
- Installing the TIWAY Host Software on your PC.
- Configuring your system to use the TIWAY Host Software Package.
- Linking application programs to the TIWAY Host Software libraries.

### Background Requirements

Please note that this chapter does not teach you how to compile programs written in any software language, nor does it contain a description of the TIWAY subroutines. The subroutines are described later in this manual. The following documentation is also helpful:

- MS-DOS operating system manual for your PC
- Operating instructions for your PC
- Appropriate software language manual(s).

### Hardware Requirements

The following hardware constitutes the basic requirements necessary to support the TIWAY Host Software Package:

- IBM®-compatible personal computer (PC)
- 640 Kbytes RAM
- One floppy disk drive (5 1/4" or 3 1/2" double-sided/double density)
- One hard disk drive (with 1 Megabyte available disk space)
- One serial port
- Supported compiler
- Appropriate software language manual(s).

---

**NOTE:** The hard disk drive can be replaced with the floppy disk drive, if desired. However, this is not recommended due to the amount of disk storage space necessary to store data and develop programs.

---

---

**Software Requirements**

You need the following software to install and use the TIWAY Host Software Package:

- Operating System: MS-DOS, Revision 3.0 or later.
- One or more of the following programming languages:
  - Microsoft C++
  - Microsoft C
  - MS-BASIC (Interpretive), or
  - BASIC for IBM® PC (Interpretive)
  - Compiled BASIC
  - MS-QUICKBASIC
  - MS-FORTRAN
  - MS-Pascal
  - Borland Turbo Pascal
  - Lattice C

---

**NOTE:** The software listed above may be purchased from your local computer dealer.

**NOTE:** Support for QUICKBASIC and Turbo Pascal is provided using a limited subset of this package, which includes only the functions INIT, XPAR, and FIN.

---

**Customer Support**

If you need assistance with the procedures outlined in this manual, contact your Siemens distributor or sales office. If you need assistance in contacting your distributor or sales office in the United States, call 800-964-4114. You may also contact the Technical Services Group at 423-461-2522.

## 1.2 PC Software Package Files

Table 1-1 contains a list of the files you will find in this software package. A brief description of each file and its contents follows the table.

Table 1-1 PC Software Package Files

<b>Title</b>	<b>File Extension</b>	<b>Description</b>
ALIGN	OBJ	This file enables the GETMSG subroutine in the TIWAY Host Software Package to find the messages used during system operation, and ensures that those messages are formatted correctly.
CONFIG	SYS	This is a file which MS-DOS looks for when it boots, and it may be placed in the system root directory or incorporated into an existing CONFIG.SYS file.
CVUTA	DEV	This file contains the device driver software for an IBM-compatible computer using the internal TIWAY adapter card.
DEF	BAS	This is essentially an "Include" file for use with Interpreted BASIC. It defines the entry points into the subroutines, and sets up variables so your program will know where to call them.
FULLSAMP	C	This file contains a sample C program to be used as an example when programming in that language. It has calls to all library functions.
HADRVAT	DEV	This file contains the device driver software for an IBM Personal Computer with a 80286 or higher processor, using an RS-232 port.
HADRVIBM	DEV	This file contains the device driver software for an IBM XT personal computer, using an RS-232 port.
HADRVTI	DEV	This file contains the device driver software for the Texas Instruments® Professional Computer (TIPC), using an RS-232 port.
MAKEC	BAT	A DOS batch file to compile and link any Microsoft C program.
MSG	OBJ	Contains text for error messages used in the TIWAY Host Software Package.
NETAUTOC	DAT	This file contains the settings which are used by NETAUTOC.EXE to automatically configure the host adapter.
NETAUTOC	EXE	This file contains a network management utility program which is used to configure the host adapter automatically. (Uses NETAUTOC.DAT).
README		This file contains instructions relating directly to installing the TIWAY Host Software for PC, calling up programs relating to software installation, as well as instructions for compiling TIWAY application programs.
SAMP_C	TYP	This is the C "Include" file which contains data types used in this package.
SAMP_PAS	PAS	This is an "Include" file for Pascal programs. It declares the TIWAY subroutines to be external, and gives all the procedure declarations. For the Pascal user, SAMP_PAS.PAS contains forward declarations to be used in programming.
SAMP_PAS	TYP	This is another Pascal "Include" file which contains the data types which are used in the SAMP_PAS.PAS declarations. For the Pascal user, SAMP_PAS.TYP contains type declarations to be used in programming.
SAMPLE	BAS	This file contains a sample BASIC program to be used as an example when programming in that language.

Table 1-1 PC Software Package Files (continued)

<b>Title</b>	<b>File Extension</b>	<b>Description</b>
SAMPLE	C	This file contains a sample C program to be used as an example when programming in that language.
SAMPLE	PAS	This file contains a sample Pascal program to be used as an example when programming in that language.
SAMPLE2	C	Another sample C program.
SAMPLEC	MAK	This file contains instructions for compiling and linking a C program with the TIWAY library.
TAG	EXE	This file contains a program which loads a logical tag table for the ASCII Tag names.
TAG	TBL	This file contains the table with the named Tag specifications and the corresponding routing information. It is an ASCII file which can be edited with any ASCII text editor.
SAMP_TAG	TAG	SAMP_TAG.TAG is an example of a TAG.TBL file.
TIBASIC	EXE	This file contains a program which loads the subroutines to be used under Interpreted BASIC and chains to the file BASIC.EXE.
TICONFIG	DAT	This file is used by the TIUSER utility. It contains permission information with switches which can be set to allow or disallow downloads to programmable controllers. The switches can also be set to control state changes in programmable controllers. The switches can also be set to control state changes in programmable controllers. TICONFIG.DAT should be located in the same directory as the TIUSER subroutine.
TIMSGBCM	OBJ	This is a message-handling file for Compiled BASIC subroutines.
TIMSGC	OBJ	This is a message-handling file for Lattice C subroutines.
TIMSGPF	OBJ	This file contains the TIWAY Host Software message handling subroutines, as well as utility subroutines such as: GETMSG (looks up TIWAY error message text) PUTMSG (outputs TIWAY error message to default output device) BLDMSK (builds data acquisition block mask)
TIPROG	EXE	This file contains a programming utility which can be used to manipulate the TIWAY Host Software subroutines. Fundamentally, its purpose is to enable you to get to know how the subroutines work and what kind of information they need for proper operation. TIPROG.EXE is executed to run the TIPROG interactive operator utility.
TIUSER	EXE	This file contains a network management utility program which is used to determine the status of network Secondaries. TIUSER.EXE is executed to run the TIUSER interactive operator utility.

## PC Software Package Files (continued)

---

Table 1-1 PC Software Package Files (continued)

<b>Title</b>	<b>File Extension</b>	<b>Description</b>
TIWAYBCM	OBJ	This file contains the subroutine library for Compiled BASIC.
TIWAYC	OBJ	This file contains the subroutine library for Lattice C.
TIWAYPF	OBJ	This file contains the subroutine library for FORTRAN, Pascal, Microsoft C, and Microsoft C++.

### Operation

This software support package provides two related but distinct interfaces:

- A subroutine library for application programs.
- Utilities for network management and network access.

The overall package, when implemented, will enable your host application program to communicate with devices attached to the TIWAY network. It also provides a high level interface between a TIWAY network with a host adapter and application programs running on a PC. In general, the package also does the following:

- Enables you to process Host Adapter command codes, TIWAY primitives, and CIM functional commands.
- Provides limited support for CCU task codes.
- Performs status checks.
- Supports programs written in the computer languages listed in Section 1.1.
- Provides an interactive operator interface to network facilities.
- Enables direct data format conversion embedded in selected routines.
- Enables logical to physical address conversion using site-specific tag table.
- Provides error detection and message generation for the following classes of errors:
  - Host Adapter errors
  - TIWAY Primitive errors
  - CIM functional command errors
  - Directive status errors
  - I/O status errors



- Supports asynchronous I/O completion for a subset of the routines.

## 1.3 Software Installation and Operation

---

This section explains how to do the following tasks:

- Back up the software provided in your TIWAY Host Software Package.
- Install your TIWAY Host Software.
- Configure your system to use the TIWAY Host Software Package.

### Backing Up Your Software

In order to avoid system down time due to loss or damage to the software diskette provided in the TIWAY Host Software Package, one back-up copy should be used as your working copy.

In order to back up the software, you must prepare a diskette. Use the MS-DOS FORMAT command. Refer to your MS-DOS operating instructions and proceed as directed.

---

**NOTE:** MS-DOS manuals list instructions for installing the computer operating system, formatting diskettes, and making back-up copies of your software.

---

Use the DISKCOPY command to copy the software from the distribution diskette. Again, the working copy should be the copy you create.

## Software Installation and Operation (continued)

---

### Installation

The installation procedure which follows assumes that a hard disk drive is being used as the system drive. A hard drive is recommended because it greatly reduces the time required to access information, and provides for much greater storage capacity than using a floppy-disk-only based system.

---

**NOTE:** Before installing TIWAY software, read the file called "readme." on disk 1. It contains pertinent information that may not be covered in this manual.

---

**Installing TIWAY Software** The following three steps are required to install the TIWAY Host Software into your computer. A special directory should be created for the TIWAY software.

1. Create a working directory on the hard disk in your system. This can be done using the MS-DOS Make Directory (MD) command.
2. Transfer control to the working directory. This can be done using the MS-DOS Change Directory (CD) command.
3. Copy the files from the TIWAY floppy disk (back-up copy) to the working directory.

**Configuring Your System** After software has been moved to the appropriate directory, add the contents of the CONFIG.SYS file in the working directory (the CONFIG.SYS file from your software package) to the CONFIG.SYS file in the root directory as appropriate.

If a CONFIG.SYS file does not exist in the root directory, copy the CONFIG.SYS file in the working directory into the root directory.

---

**Contents of the CONFIG.SYS file** The CONFIG.SYS file must contain a statement specifying the appropriate driver for your PC. You must use only one of the following:

- For a TIPC, the following statement is required (assuming you are using a hard disk as recommended)

DEVICE = e:\TIWAY\HADRVTI.DEV

- For a PC XT or compatible, the following statement is required (again assuming you are using a hard disk)

DEVICE = C:\TIWAY\HADRVIIBM.DEV

- For an IBM AT or compatible the following statement is required (again assuming you are using a hard disk).

DEVICE = C:\TIWAY\HADRIVAT.DEV

---

**NOTE:** TIPC's use E: for the hard disk; other IBM compatible personal computers use C:. An example of a subdirectory specification would be

DEVICE = c:\TIWAY\HADRIVAT.DEV (where TIWAY is the subdirectory name).

---

**Using BASIC with TI and IBM PCs** In the case of the TIPC, if Interpreted BASIC is used, it is necessary to copy the file BASIC.EXE into the same directory as the program TIBASIC.EXE. (BASIC.EXE is found on the disk supplied with the BASIC manual).

In the case of IBM-compatible PCs (not TIPC), it is necessary to copy either BASIC.COM or BASICA.COM into the same directory as TIBASIC.EXE, and rename BASIC.COM or BASICA.COM to BASIC.EXE.

For example, copy BASICA.COM from floppy to hard disk as follows:

```
COPY A:BASICA.COM C:\TIWAY\BASIC.EXE
```

**Compiled Languages** For compiled languages, the necessary compiler files can be copied to the same directory as the TIWAY files, or the DOS PATH command can be used to cause the system to search the appropriate directories.

**Loading the Device Driver** The steps listed below will install and load the TIWAY Host Software device driver into the correct directory.

1. Transfer control of your system to your root directory. This can be done by using the Change Directory (CD) command.
2. Copy the hardware device driver file for your system (TIPC: HADVRTI.DEV; IBM: HADVRIBM.DEV; AT: HADRVAT.DEV) to your root directory. Use the COPY command.
3. Re-boot the computer to load the TIWAY Host Software device driver.

## 1.4 Using TIWAY

---

Once the TIWAY Host Software is installed in your system, it is ready to be used. The TIWAY subroutine libraries support the following computer languages.

- MS-BASIC for TIPC (Interpretive)
- BASIC for IBM PC XT (Interpretive)
- Compiled BASIC
- MS-FORTRAN
- MS-Pascal
- Lattice C
- Microsoft C

With the exception of Interpretive MS-BASIC, programs in all of the languages listed above must be compiled and linked with one of the supported compilers.

BASIC programs may be executed in either interpretive or compiled versions.

For additional information regarding language support, refer to the "readme." file.

---

**NOTE:** The software listed above may be purchased from your local computer dealer.

Support for QUICKBASIC and Turbo Pascal is provided using a limited subset of this package, which includes only the functions INIT, XPAR, and FIN.

---

## 1.5 Using a Compiled Language

---

There are four steps to creating a compiled program. These are:

1. Writing
2. Compiling
3. Linking
4. Running

You should refer to your appropriate language manual for writing and compiling. The purpose of this chapter is to provide the “link-order” for your TIWAY Host Software program, concentrating on exceptions to standard procedures for writing, compiling, linking, and running the TIWAY Host Software.

If you have questions after reading the remainder of this chapter, call the number listed under Customer Support in Section 1.1.

## 1.6 Program Linking

---

The following paragraphs supply the information necessary to enable you to link your application programs to the TIWAY subroutine libraries, and to use the features provided by the TIWAY utility packages. This information should enable you to link your application with the TIWAY Host Software package to run under the MS-DOS operating system.

### Object Code Files

Each of the computer languages listed earlier, with the exception of Interpreted BASIC, has a compiler program which must be executed and provided an ASCII source code file to produce an object code file. This object code file can then be combined with other object code files to produce an executable machine code file.

The process of combining object files from one computer language with object files and/or subroutines from subroutine libraries in another computer language is called linking. (However, two languages are not necessarily involved; object files and subroutines in the same language can also be linked.)

### Linking on the PC

Linking on the PC is accomplished with the MS-DOS LINK command (or the linker supplied with your compiler). There are three different formats of the Link command.

- The first format is the easiest to use if you are a beginning programmer. Simply type "LINK" into the computer and respond to the four prompts:
  - The first prompt asks for the "object" files
  - The second prompt asks for the "run" files
  - The third prompt asks for the "list" file
  - The fourth prompt asks for the "library" files
- In the second format, all the required link information is entered on the command line, separated with commas.
- In the third format, place all of the link information in a file, then run Link and enter the name of the file.

Examples of the use of these three formats are given in the following paragraphs.



## Program Linking (continued)

---

**Using the Link Command** For the following discussion, you should assume that the program has been compiled using the MS-Pascal compiler.

Sample Link program:

```
C>LINK <return>
Object modules [.OBJ]: A:ALIGN+MSG+TIMSGPF+YOURPROG+ TIWAYPF <return>
Run file [CL:EXE]: YOURPROG <return>
List file [NUL.MAP]: YOURPROG <return>
Libraries [.LIB]: Pascal <return>
```

---

**NOTE:** In the first line of the above program, Link is your response. In all second, third, fourth and fifth lines, all data after the ]: is your response, with the exception of the <return>s. YOURPROG signifies your own program object file.

---

The response A:ALIGN tells the link program that the ALIGN.OBJ file is on disk drive A. The “+” character tells the link program that another file is being added to the object file list. Up to eight object files can be linked in this way.

The same “+” character is also used to access up to eight LIBRARY.LIB files in the library list (although for this example just one library was used.)

The information contained inside the brackets is the default values of the commands.

---

**Entering Link Information** This format assumes that the program to be linked has been compiled using the MS-DOS BASIC compiler.

The following command uses a single line to call the Link program to link YOURPROG.OBJ, TIWAYBCM.OBJ, TIMSGBCM.OBJ, ALIGN.OBJ, and MSG.OBJ together. The linker automatically searches for the BASIC run-time library BASRUNG.LIB.

```
LINK ALIGN+MSG+TIMSGBCM+TIWAYBCM+YOURPROG, YOURPROG,; <return>
```

The switch "P" will cause the following message to be output to the user:  
About to Generate .EXE file  
Change Disks <return>

---

**NOTE:** More information about switches is contained in the Link Options section of this chapter.

---

One additional observation about this particular command: because no disk drives were specified in this command, all of the above files are assumed by the linker to be on the default subdirectory on the hard disk.

**Entering Link Information From a File** This format assumes that the application program to be linked has been compiled using the MS-FORTRAN compiler.

For this format, an ASCII file must be created using an ASCII editor. The information entered into this file is the same information that you enter as commands in the other two Link formats.

## Program Linking (continued)

---

The following commands illustrate how the information file may be created using the EDLIN editor.

```
EDLIN YOURFILE <return>
```

```
New file
```

```
*I<return>
```

```
1: *ALIGN+MSG+TIMSGPF+YOURPROG+TIWAYPF <return>
```

```
2. */MAP <return>
```

```
3: *YOURPROG <return>
```

```
4: *A:FORTTRAN.LIB <return>
```

```
5: *<shift break>
```

```
*C> <return>
```

```
*C>
```

(1, 2, 3, 4, etc., are the computer prompts)

The above file combines YOURPROG.OBJ, TIWAYPF.OBJ, ALIGN.OBJ, TIMSGPF.OBJ, and MSG.OBJ files. The executable file will be placed in the default runfile (in this case TIWAY.EXE). A public symbol map is created and produced by the switch "/MAP" (see Link Options section for more information on switches). A list file called YOURPROG.MAP is produced, and the library file called FORTTRAN.LIB on disk drive A is searched for subroutines.

To invoke the Link using the above file, type the following command:

```
LINK @YOURFILE <return>
```

---

## Link Options

Seven options which control linking operations are recognized by Link. These seven options are called switches, and must be preceded in the command line by a forward slash (/). Switch names may be grouped at the end of any one command line, or scattered at the end of several command lines. Switches may be abbreviated using any number of the letters which make up the switch name; however, these letters must appear in sequential order.

The seven switches are described in the following paragraphs.

**/DSALLOCATE** The switch /DSALLOCATE directs Link to load all data into the high end of the data segment. By using the /DSALLOCATE switch without using the /HIGH switch, an application program is allowed to dynamically allocate any available memory which is below the area on the high end of the data segment occupied by data. The application program can have this allocated area pointed to by the same DS pointer.

---

**NOTE:** Pascal and FORTRAN programs require this dynamic allocation.

---

**/HIGH** The switch /HIGH causes Link to place the run image as high a possible in memory.

---

**NOTE:** Do not use this switch with Pascal or FORTRAN programs.

---

**/LINENUMBERS** The switch /LINENUMBERS directs Link to include the line numbers and addresses of source statements in the list file.

**/MAP** The switch /MAP directs Link to include all global symbols in the list file. These symbols are listed alphabetically at the end of the file.

**/NO** The switch /NO causes Link to ignore default libraries which may be specified by the compiler.

**/PAUSE** The switch /PAUSE causes Link to stop processing and allow diskettes to be changed in the disk drives before continuing.

**/STACK** The switch /STACK creates a stack of the same size specified by number. At least one of the object modules must contain a stack allocation statement, or Link returns an error message.

For further information concerning Link, consult the MS-DOS operating system manual.

## Program Linking (continued)

---

**Link Order** The TIWAY Host Software Object Library for compiled application programs contains eight separate object modules. Each supported language uses its own specific object modules except MS-FORTRAN and MS-Pascal, which use the same object modules. There are two object modules which are used with all languages when the message facilities of the software are included. The object files are listed in Table 1-2 by language.

Table 1-2 Object Files

Language	Link Order
MS-FORTRAN/ MS-PASCAL	1. TIWAYPF.OBJ 2. TIMSGPF.OBJ
MS-BASIC (Compiled)	1. TIWAYBCM.OBJ 2. TIMSGCM.OBJ
Lattice C	1. TIWAYC.OBJ 2. TIMSGC.OBJ
All compiled languages	1. ALIGN.OBJ 2. MSG.OBJ

The order in which the above object files and your application program are linked to the MS-DOS operating system is extremely important to the correct operation of your program. The link order for each language is specified. Carefully read the description of the linking order for the language you intend to use, and follow it exactly.

---

**NOTE:** Additional information regarding Linking with other compilers is provided in the file named "readme." The message facility may or may not be used, and for this reason has been separated from the rest of the package.

---

---

MS-FORTRAN without messages  
Link YOURPROG+TIWAYPF,,FORTRAN

MS-FORTRAN with messages  
Link ALIGN+MSG+TIMSGPF+YOURPROG+TIWAYPF, YOURPROG,,FORTRAN

MS-Pascal without messages  
Link YOURPROG+TIWAYPF,,Pascal

MS-Pascal with messages  
Link ALIGN+MSG+TIMSGPF+YOURPROG+TIWAYPF, YOURPROG,,Pascal

MS-BASIC without messages  
Link TIWAYBCM+YOURPROG, YOURPROG;

MS-BASIC with messages  
Link ALIGN+MSG+TIMSGBCM+TIWAYBCM+YOURPROG, YOURPROG;

#### Lattice C

Special batch files are provided to link Lattice C programs to the MS-DOS operating system if the MAKELC.BAT file is invoked to load the Lattice C compiler into your computer (see Lattice C Compiler).

These batch files may be used without modification to link the TIWAY subroutine library to the operating system if no message facilities are to be invoked. If, however, you plan to use the error message routines provided in the TIWAY subroutine library, you must first modify the link batch files as follows.

Modify LINKL.BAT

From:

Link /1c /1 /c %1 %2 %3 %4 %5 %6 %7 %8 %9 , %1 /1c /1 /c

To:

Link %1 %2 %3 /1c /1 /c %4 %5 %6 %7 %8 %9, %4,, /1c /1 /c

To invoke the above file for TIWAY, type:

LINKL ALIGN MSG TIMSGC YOURPROG

## Program Linking (continued)

---

Modify LINKML.BAT

From:

Link /1c /1/c %1 %2 %3 %4 %5 %6 %7 %8 %9, %1,, /1c /1 /1cu + /1c / 1 /lc

To:

Link %1 %2 %3 /1c /1 /c %4 %5 %6 %7 %8 %9, %4,, /1c /1 /1cu + /1c /1 /c

To invoke the above file for TIWAY, type:

LINKML ALIGN MSG TIMSGC YOURPROG

If MAKELC.BAT has not been used to install your Lattice C compiler, the link order is as follows:

Lattice C without messages

Link CL+YOURPROG+TIWAYC, TIWAYC,,LCL

Lattice C with messages

Link ALIGN+MSG+TIMSGC+CL+YOURPROG+TIWAYC, TIWAYC,,LCL

Compiling with  
Lattice C

Lattice C provides four different memory models to the programmer: s,p,d,l.

These memory models affect the addressing capabilities and efficiency of the compiled program. To use TIWAY subroutine libraries with a program written in Lattice C, the programmer must compile that program using the L-memory model.

If MAKELC.BAT has been used to install the Lattice C compiler, the batch file LCL.BAT will compile your programs using the L-memory model. If your compiler has not been installed using MAKELC.BAT, refer to the *Lattice C Compiler Reference Manual* provided with your Lattice C compiler for information on how to compile your programs using the L-memory model.

## 1.7 Executing in Interpreted BASIC

---

As previously noted, a program in BASIC may be executed in the interpretive mode. This means that each line of the program is read, converted to machine code, and executed before the next program line is read. This allows you to write and debug your program in an interactive environment.

The disadvantage of executing a program in the interpretive mode is that it runs much more slowly than the compiled version of the same program.

---

**NOTE:** See the *MS-BASIC Software Library Manual* for additional information.

---

Due to the dynamic nature of the MS-BASIC interpreter, there are differences in the programming procedures for TIWAY subroutines. These differences are in variable declaration and initialization.

Methods of passing arrays and single variables to the TIWAY subroutines are also different.

### Getting Started

To use the TIWAY subroutines in the interpretive mode, follow these steps:

1. Use the MS-DOS COPY command to load your licensed copy of MS-BASIC into the directory containing the TIWAY subroutines.
2. Execute the TIBASIC.EXE program to load the TIWAY subroutines and start the BASIC interpreter. (This is why BASIC.EXE must be in the same directory.)
3. In your program, locate the starting address of the subroutine library by reading memory locations 0:03E0 and 0:03E1. The contents of these locations are:

The low byte of the starting address is in 03E0.

The high byte of the starting address is in 03E1.

---

**NOTE:** The file DEF.BAS contains the necessary statements to load the starting address into your BASIC program. The BASIC instruction "Merge" can be used to add this file to your program. (Refer to the MS-BASIC Language Manual for additional information.)

---



## Executing in Interpreted BASIC (continued)

---

### Typical BASIC Startup

For this typical application, assume that the TIWAY files are contained in a hard disk directory named TIWAY. Assume also that the default drive is the hard drive, and that you are using a TIPC.

---

**NOTE:** The IBM program includes two files: BASIC.COM and BASICA.COM. One of these files must be renamed to BASIC.EXE.

---

**Procedure** The following gives the procedure for starting up BASIC.

1. Put the disk containing your licensed copy of MS-BASIC into the floppy disk drive (which should be drive A:) and close the drive latch.

2. Make certain you are in the TIWAY directory. If you are not, type the following commands:

```
E> CD\           ; get into the root directory
E> CD\TIWAY     ; get into the TIWAY directory
```

3. Copy MS-BASIC to the TIWAY directory as follows:

```
E>COPY A:BASIC.EXE E:
```

You should get a response from the computer telling you how many files were copied.

4. Execute the TIBASIC.EXE program to start BASIC and load the subroutines.

```
E:>TIBASIC
```

If all goes well, the BASIC interpreter should respond with an editor screen and prompt "OK".

5. Start building your program using the following commands to access the TIWAY subroutine library. Two methods are shown below:

#### Method 1:

```
10 DEF INT I           ' Define variable as integer
20 DEF SEG=&H0         ' Set segment pointer to point
30 REM                ' to memory address 0:0
40 I=PEEK(&H3E1)*256+PEEK (&H3E0) ' Get high and low byte of
50 REM                ' starting address
60 DEF SEG=I          ' set segment pointer to TIWAY
70 REM                ' starting address
```

#### Method 2:

```
10 CHAIN MERGE "DEF.BAS", 20,ALL ' DEF.BAS define
20 GOSUB 65000                ' TIWAY entry points (starts
                              ' at 65000)
```

---

## Calling Subroutines in BASIC

BASIC uses a special procedure to branch to subroutines not contained within the BASIC program being executed. In order to call an external subroutine, a variable with the subroutine name must be created and initialized to the proper value for linking to the subroutine. This value, when combined with the value loaded by the DEF SEG command, is used to establish the address to branch to in the PC's memory.

Two calling methods are used to link a BASIC program to the TIWAY support package, as follows:

**Method 1:** Use the entry point offset of 0. While not necessary, it is suggested that a variable name of TIWAY be created and initialized to the value of 0. The appropriate subroutine is selected by making the first calling parameter an ASCII string containing the name of the subroutine to be executed. In BASIC this is accomplished by creating a string variable containing the name of the desired subroutine.

**Method 2:** Create a variable initialized to the proper offset value for the required subroutine. With this method, the subroutine is called directly, rather than by going through one subroutine to get to another. This method is slightly faster, and when using the subroutine names as the calling variable names, allows for a more easily read program. Both of these methods can be freely mixed in the same program.

The following example illustrates the use of both methods calling subroutines:

### Example of Subroutine Calls in BASIC

```
100 TIWAY=0 : INS="INIT      " : PORT="P1, 19200,7,1,E,3,A"  
110 CALL TIWAY (INS,ISTAT,PORT) ' Call INIT subroutine  
120 ADIAG=55 : XTN=0 : HWY=1  
130 CALL ADIAG (ISTAT,XTN,HWY,RILEN,RI132(0)) ' Call ADIAG subroutine
```

The file DEF.BAS creates and initializes variables that can reference all of the TIWAY subroutines. This file can be MERGED into a BASIC program and executed as a subroutine, thereby saving the programmer time and effort. If the BASIC program is to be compiled, the MERGE statement can be deleted and the program will be ready to compile.

## Executing in Interpreted BASIC (continued)

---

### Subroutine Calling Parameters

Because MS-BASIC uses a method different from FORTRAN and Pascal to pass variable and array addresses, several differences exist in the manner in which parameters are declared when using the TIWAY subroutines.

With the exception of variables used for floating point numbers, all variables and arrays used in the TIWAY subroutines should be declared as INTEGER.

Those variables declared in FORTRAN as INTEGER\*4 can be declared as INTEGER arrays with a dimension of 2. A complete list of the variable declarations for BASIC is found in Chapter 3 of this manual.

### Passing Parameters to Subroutines

BASIC does not pass parameter addresses in the same manner as does FORTRAN or Pascal. Because of this, the naming of parameters, specifically arrays, is different with BASIC. Rather than give only the name of the array, as in FORTRAN, in BASIC the first data location in the array must also be provided.

This has the advantage of allowing you to begin passing data from anywhere in the array, but care must be taken to prevent overrunning the dimension of the array. The following statement is an example of how arrays are passed to the TIWAY subroutines:

```
CALL TIPUT (ISTAT,XTN,TAGS,NNNN,WI135(0),SSTAT)
```

---

**NOTE:** BASIC does not allow the passing of data constants or expressions to a subroutine. Only named variables and arrays can be passed. If a constant is to be passed, a variable with the constant's value must be created.

---

Since BASIC uses dynamic memory allocation for variables, all variables must be defined by initialization, including those that are used for the subroutine calls (such as SSTAT), before the subroutine call is made. It is sufficient to define these variables by setting them to 0.

Character type variables, such as tags and the naming of subroutines within a call to TIWAY, are also handled somewhat differently. These must be declared as STRING variables. Their length is also important, for when using a string variable as an ASCII tag, the length of the variable must be 11 characters. If it is a short tag, spaces must be added to fill it out. This is also true for variables used to name subroutines when calling TIWAY; the length must be 6 characters.

## 1.8 Named Tag Table Maintenance

---

This section tells you what a tag and a tag table mean, and how to build and install the TIWAY tag table.

A tag is a data structure that uniquely identifies the path to a piece of information located in a TIWAY Secondary. There are three methods of specifying a tag:

- Binary specifications (array of integers immediately specifying the path)
- ASCII specifications (ASCII string immediately specifying the path)
- Named specifications (logical name indirectly referencing a path). Specifying of a tag is further discussed in Chapter 8.

### Creating the Named Tag Table

Named tag specifications are defined in the MS-DOS file TAG.TBL. This file can be edited using any ASCII text editor. Comments for the user of this file are indicated by the “{” character in the first column of any line.

The maximum length of a tag is determined by a four-digit hexadecimal number that is the first uncommented line in the file. After this appears, tag specifications or additional comments can follow.

Each uncommented line in the file is used to specify one tag. The first item in each line specifies the tag name. A space is used to separate the tag name from the second item, which is the highway number (always 01 for the PC version of the TIWAY Host Library). The third item specifies the Secondary address in hex; the fourth specifies the data element (TT) type, and the fifth specifies the address in hex.

Comments can be added to each line by placing a space between the data address and the comment.

The tag table is loaded into memory by executing the file TAG.EXE. This file is executed by typing TAG and pressing the return key when the monitor prompt is displayed. This program will load the tag table in the PC's memory. If desired, the table can be edited and reloaded by issuing the TAG command again at any time. Tags are stored in file TAG.TBL.

## Named Tag Table Maintenance (continued)

---

### Accessing a Named Tag Specification

To access a Named Tag specification from a program, use the name in the form of an ASCII string as the tag argument in the TIWAY subroutine calls. The TIWAY subroutines interpret a value in a tag argument field as follows:

- If the first byte in the string is an ASCII “#” then the remainder of the string is interpreted as an ASCII specification.
- If the first byte is an 0, then the string is interpreted as a binary specification.

Otherwise, the entire string is a key and the TAG region is searched for the specification definition.

When using Pascal, FORTRAN, C, or C++, the tag name may be specified immediately (within quotes) or may be the value of a variable which contains the ASCII string. In BASIC, only the specification of a tag name within a variable is allowed. BASIC does not support the immediate specification.

# Chapter 2 Overview

---

<b>2.1</b>	<b>TIWAY System Characteristics</b> .....	<b>2-2</b>
<b>2.2</b>	<b>Host Software Functions</b> .....	<b>2-3</b>
	Levels of Communication .....	2-3
	Languages Supported .....	2-5
	Data Format Conversion .....	2-5

## 2.1 TIWAY System Characteristics

---

TIWAY is a Local Area Network (LAN) that is designed for industrial environments. The TIWAY network connects a series of 500/505 Secondaries with a host computer. Secondaries include such devices as programmable logic controllers, Unilink Secondary Adapters, and IT Micro-Remote Control Units. With the network, you can obtain, modify, or replace data stored in any of the following Secondaries:

- SIMATIC 500 controllers
- SIMATIC 505 controllers
- Non-Siemens equipment through a Unilink Secondary Adapter
- IT-160 Micro-Remote Control Unit

TIWAY is a multiple-hosted network, meaning that several host computers can direct and collect information from Secondary devices. With appropriate host software, a TIWAY operator can program, monitor, and control any Secondary on the network from a single location.

---

**NOTE:** In this document, the term “Secondary” is used to refer to both the attached device and the interface module.

---

## 2.2 Host Software Functions

The TIWAY Host Software Package provides a communication interface between the TIWAY network and a host computer using the Unilink Host Adapter. The Unilink Host Adapter allows communication with up to 254 controllers or other Secondaries on a TIWAY network by connecting a host computer to the network. The TIWAY Host Software Package provides access to all of the Secondary functions available through the network, as well as a complete set of network management and diagnostic facilities.

### Levels of Communication

The TIWAY Host Software Package provides the highest level of communication between an operator and the Secondaries on a TIWAY network. To make communication with a Secondary more user-friendly, there are different levels of communication between each of the network components, as shown in Figure 2-1. Each level has its own format for communicating, with each level building on the next lower level.

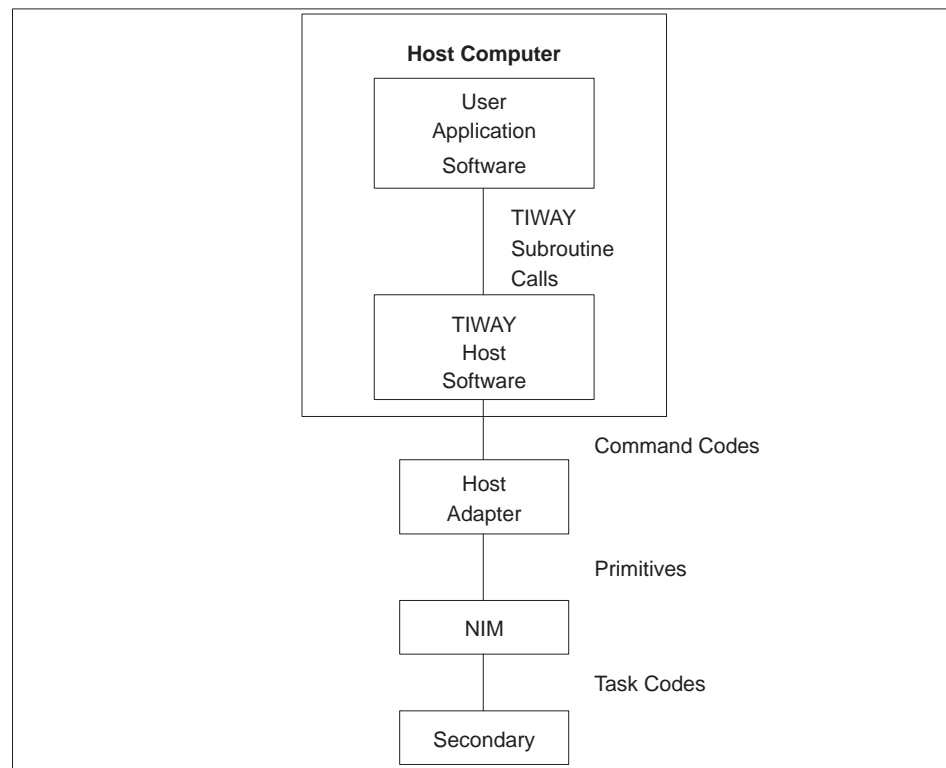


Figure 2-1 Network Communication Levels



## Host Software Functions (continued)

---

First, at the most basic level, there is Task Code communication between the Network Interface Module (NIM) and the controller. Each class of controller model uses a different group of Task Codes. The next level of communication is that between the NIM and the host adapter. At this level, primitives (or the Universal Command Language Instructions) are used. The Universal Command Language (UCL) is a unified language recognized by all NIMs. The third level of communication is between the host adapter and the host computer. Communication at this level is accomplished through the use of Host Adapter Command Codes. Finally, at the highest level, there is the communication between an operator and the host computer. This level involves the host software that is used to generate the command codes. It is at this level that you interface to the TIWAY Host Software Package.

The following is an example of the series of events that occur when a specific instruction is sent to a controller:

At the highest level:

- You run a program written to do a specific job. The program asks for the required data (e.g., which controller to talk to).
- Your program calls a TIWAY Subroutine to generate the network instruction. [For example: CALL TIWAY ('TIGET',istat,xtn,...)]
- The subroutine generates the instruction from the data that you supplied to it, and formats the instruction into a TIWAY Primitive embedded in a Host Adapter Command Code. The subroutine then sends the instruction to the Host Adapter Device Driver.
- The device driver embeds the instruction within either a Non-Intelligent Terminal Protocol (NITP) frame or a Binary Data Link Control (BDLC) frame, and sends it to the host adapter.
- The host adapter strips away the command code section of the instruction and uses it for routing the remainder of the instruction (primitive) to the network.
- The NIM takes the instruction, isolates the part it needs, and translates the remainder into specific PLC Task Codes that it sends to the controller.

---

At the lowest level:

- The controller executes the Task Codes and sends the response to the NIM.
- The NIM uses the responses to the PLC Task Codes to generate the network response. It embeds the response in the appropriate TIWAY Primitive and sends it through the network to the host adapter.
- The host adapter appends command code data to the response, embeds it within an NITP frame, and sends it to the host computer.
- The device driver in the host computer strips off the NITP section of the response and sends the remainder to the TIWAY Subroutine.
- The TIWAY Subroutine strips off the command code section of the response and sends the remainder to the user program.
- Your program takes appropriate action, based upon the response.

Languages Supported

The TIWAY Host Software Package, which provides a convenient user interface to the TIWAY facilities, makes the commands transparent to you or to any user, and allows programming in high-level languages. Languages supported are Fortran, Pascal, BASIC (compiled), C, and assembly language.

Data Format Conversion

In the 500/505 Secondaries, data are represented in 500/505 format (the format in which numbers are stored in 500/505 Secondaries). Specific conventions follow.

- Packed discrete data
- Unpacked discrete data
- Integer data
- Instructions
- Floating point data

The TIWAY Host Software Package allows access to data in 500/505 format and allows you to access data in host 500/505 format (the data format representations supported in the host computer). You should refer to the *SIMATIC TIWAY I Systems Manual* (PPX:TIWAY-8101-x) for a complete description of 500/505 format.

# TIWAY Subroutine – Calling Arguments

---

3.1	Overview .....	3-2
3.2	Reference List of Subroutines and Arguments Reference List .....	3-3
3.3	Reference List of Arguments .....	3-6
3.4	List of Data Elements Types .....	3-14
3.5	Addressing .....	3-17
3.6	Buffer Specification .....	3-21
3.7	Data Block Manipulation .....	3-23
3.8	Diagnostics .....	3-25
3.9	Masks .....	3-26
3.10	Network Access .....	3-27
3.11	Status Determination .....	3-28

## 3.1 Overview

---

The calling arguments used in the TIWAY Interface Subroutines are described in this chapter. The argument descriptions are grouped by function; for instance, all buffer arguments are described in the section entitled Buffer Specification. A thorough understanding of the functions of the arguments is essential for effective use of the TIWAY Interface Subroutines.

In addition to the descriptions, three reference tables are provided. Table 3-1 lists each subroutine and the arguments it uses, and Table 3-2 and Table 3-3 list each argument and its description, declaration (Fortran, Pascal, BASIC, and C), and access. Refer to the appropriate Secondary manual and to the *SIMATIC Unilink Host Adapter User Manual* for more assistance in using these tables.

---

**NOTE:** Chapter 3 describes all calling arguments used in the TIWAY Interface Subroutines (the subroutines contained in Chapter 4). However, the TIWAY Support Routines (described in Chapter 5) use arguments that do not occur in the Interface Subroutines. Therefore, the arguments used in the Support Subroutines are defined in the Notes on Call Format section of each Support Subroutine (see Chapter 10).

---

## 3.2 Reference List of Subroutines and Arguments Reference List

Table 3-1 Subroutines and Arguments Reference List

Subroutines Name TIPROG		Arguments	Description	Related Codes
<b>Session Control</b>				
INIT	(IN)	istat, xtn, portstr	Initializes subroutines	
FIN	(FI)	istat, xtn	Finishes subroutines	
<b>Host Adapter Command Codes</b>				
ACTVAT	(AC)	istat, xtn, hwy, wilen, wi4, rilen, ri4	Connects Secondaries	04
ADIAG	(AD)	istat, xtn, hwy, rilen, ri32	Reads host adapter diagnostics	08
BRDCST	(BR)	istat, xtn, hwy, cmdlen, cmd275	Broadcasts messages	02
DEACT	(DA)	istat, xtn, hwy, wilen, wi4, rilen, ri4	Disconnects Secondaries	05
POLL	(PO)	istat, xtn, tag, rsplen, rsp275	Polls Secondaries for responses	03
SDIAG	(SD)	istat, xtn, tag, rilen, ri32	Reads Secondary diagnostics	07
SECLOG	(SL)	istat, xtn, hwy, rilen, ri254	Reads Secondary log	06
XPAR	(XP)	istat, xtn, hwy, cmdlen, cmd275, rsplen, rsp275, errtyp	Sends user-constructed command	ALL <sup>1</sup>
<b>TIWAY NIM Primitives<sup>2</sup></b>				
CHNGST	(CH)	istat, xtn, tag, state, sstat	Changes state of Secondary	10
CONFIG	(CN)	istat, xtn, tag, rilen, ri32, sstat	Reads Secondary configuration	03
DEFBLK	(DE)	istat, xtn, tag, nblk, cclst, taglst, nnnlst, sstat	Defines blocks	50
FILL	(FL)	istat, xtn, tag, nnnn, pattern, sstat	Fills blocks	32
GATHER	(GA)	istat, xtn, tag, .mask, rsplen, rsp275, sstat	Gathers blocks using mask	51
GETLEN	(GL)	istat, xtn, tag, primlen, nblks	Reads maximum primitive and block lengths	04
NATIVE	(NA)	istat, xtn, tag, cmdlen, cmd275, rsplen, rsp275, sstat	Sends PLC task code	01
RDSTS	(RS)	istat, xtn, tag, rilen, ri32, sstat	Reads Secondary status	02
TIGET	(TG)	istat, xtn, tag, nnnn, rsplen, ril35, sstat	Reads block of data (host format)	20
TIPUT	(TP)	istat, xtn, tag, nnnn, wil35, sstat	Writes to block of data (host format)	30
TIREAD	(TR)	istat, xtn, tag, nnnn, rsplen, rsp275, sstat	Reads block (500/505 format)	20

The related code numbers given for each subroutine are Host Adapter Command Code numbers. (See *SIMATIC TIWAY Unilink Host Adapter User Manual* for a further description of these commands.)

<sup>1</sup> XPAR can be used to send any TIWAY NIM commands.

<sup>2</sup> The related codes for all the NIM primitives are primitive code numbers. All NIM primitive subroutines use Host Adapter Command Code 01. See relevant NIM manual for further description of these primitives.

## Reference List of Subroutines and Arguments (continued)

Table 3-1 Subroutines and Arguments Reference List (continued)

Subroutines Name TIPROG		Arguments	Description	Related Codes
<b>TIWAY NIM Primitives<sup>2</sup> (continued)</b>				
TIWRIT	(TW)	istat,xtn,tag,nnnn,wb275,sstat	Writes a block of data (SIMATIC TI format)	30
WRBUF	(WB)	istat, xtn, tag, cc, nnnn, wb275, pstat, sstat	Performs buffered write	33
WRTGAT	(WG)	istat, xtn, tag, nnnn, wb275, mask, rsplen, rsp275, sstat	Writes and gathers blocks with mask	52
<b>CIM Functional Commands<sup>3</sup></b>				
CCUSTS	(CC)	istat, xtn, tag, retsts	Read Secondary status	69
CIMDNL		istat, xtn, tag, ss, wblen, wb275, cstat, mm	Downloads data	65
CIMRD	(CR)	istat, xtn, tag, nnnn, rsplen, rsp275	Reads Secondary status	60 or 62
CIMUPL		istat, xtn, tag,ss, rsplen, rsp275, cstat, mm	Uploads data	66
CIMWR	(CW)	istat, xtn, tag, nnnn, wb275	Writes data	61 or 63
RDLOOP	(RL)	istat, xtn, tag, rtype, rsplen, rsp275	Reads loop data	6A
RNDRD1	(R1)	istat, xtn, tag, nblk, taglst, nnlst	Defines blocks	6D(00)
RNDRD2	(R2)	istat, xtn, tag, cmask, rsplen, rsp275	Gathers blocks using mask	6D(01)
RNDRD3	(R3)	istat, xtn, tag, cmask, rsplen, rsp275, nblk, taglst, nnlst	Defines and gathers blocks/mask	6D(02)
RNDRD4	(R4)	istat, xtn, tag, nnnn, wb275, cmask, rsplen, rsp275	Writes and gathers blocks using mask	6D(03) 6D(04)
<b>Status and Support Routines</b>				
BLDMSK		mask, list	Builds a mask of block numbers.	
GETMSG <sup>1</sup>		istat, rsplen, str80	Returns error message.	
HST2TI	(HT)	istat, xtn, tt, nnnn, hstbuf, tibuf	Converts data from host format to 500/505 format.	
LKUFMT	(LF)	istat, xtn, tt, fmt, tilen, hostlen, cimtyp	Returns information on format.	
LKUTGL	(LL)	istat, xtn, tag, hwy, secadd, tt, aaaa	Looks up a tag name; returns an address parameter list	
LKUTGS	(LS)	istat, xtn, tag, addlst	Returns address information.	
PUTMSG <sup>1</sup>		istat	Displays error message.	

<sup>1</sup> These subroutines are not part of the main TIWAY subroutine library. In order to reduce the size of applications programs, they are located in a separate library module (TIMSG) which may be linked if needed.

<sup>2</sup> The related codes for all the NIM primitives are primitive code numbers. All NIM primitive subroutines use Host Adapter Command Code 01. See relevant NIM manual for further description of these primitives.

<sup>3</sup> The related code numbers given for the CIM Functional Commands are Functional Command Task Code numbers. All CIM Functional Command subroutines use Host Adapter Command Code 01. See *SIMATIC PM550 CIM Manual*, PPX:550-8105-x, for further functional command descriptions.

Table 3-1 Subroutines and Arguments Reference List (continued)

<b>Subroutines Name</b>		<b>Arguments</b>	<b>Description</b>	<b>Related Codes</b>
<b>TIPROG</b>				
<b>Status and Support Routines (continued)</b>				
TIXTN		istat, xtn	Performs I/O with return before completion.	
TIXTNW		istat, xt	Performs I/O and waits for completion.	
TI2HST	(TH)	istat, xtn, tt, nnnn, tibuf, hstbuf	Converts data from 500/505 format to host format.	
<b>File Transfer</b>				
DNLOAD	(--)	istat, tag, ftype, flen, fnam	Downloads file to a Secondary.	
UPLOAD	(--)	istat, tag, ftype, flen, fnam, pflen, pname	Uploads file from a Secondary.	

### 3.3 Reference List of Arguments

Table 3-2 contains the Argument Reference List for Fortran and Pascal.

Table 3-2 Argument Reference List for Fortran and Pascal

Argument	Definition	Declaration (Fortran) <sup>1</sup>	Declaration (Pascal) <sup>3</sup>	Access
<i>Addressing</i>				
addlst	binary tag	integer array	iary32	read only
hwy	highway number	integer*2	integer	read only
mm	data element type	byte <sup>2</sup>	byte	read only
portstr	PC port setup string	byte array <sup>2</sup>	bary275	read only
rtype	data element type	integer	integer	read only
tag	address specification	integer*2 array	tag_type	read only
tt	data element type	integer	integer	read/write
<b>Buffer Specifications</b>				
<i>Command Buffers</i>				
cmd275	command buffer	byte array <sup>2</sup>	bary275	read only
cmdlen	command buffer length	integer*2	integer	read only
<i>Read Buffers</i>				
ri4	response buffer	integer*2 array	iary4	read/write
ri32	response buffer	integer*2 array	iary32	read/write
ri135	response buffer	integer*2 array	iary135	read/write
ri254	response buffer	integer*2 array	iary254	read/write
rilen	response buffer length	integer*2	integer	read/write
<i>Response Buffers</i>				
rsp275	response buffer	byte array <sup>2</sup>	bary275	read/write
rsplen	response buffer length	integer	integer	read/write
<i>Write Buffers</i>				
wblen	write buffer length	integer	integer	read
wb275	write buffer	byte array <sup>2</sup>	S_bary275	read
wi4	write buffer	integer array	iary4	read
wi135	write buffer	integer array	iary135	read
wilen	write buffer length	integer	integer	read

<sup>1</sup> The Fortran declarations are determined according to Figure 3-1.

<sup>2</sup> MS-FORTRAN does not support type byte. All arguments declared as type byte above should be declared as integer for MS-FORTRAN.

<sup>3</sup> The Pascal declarations are determined according to Figure 3-2.



Table 3-2 Argument Reference List for Fortran and Pascal (continued)

Argument	Definitions	Declaration (Fortran) <sup>1</sup>	Declaration (Pascal) <sup>3</sup>	Access
<b>Data Manipulations</b>				
cc	UCL instruction option code	integer*2	integer	read only
cclst	array of block numbers to define	integer*2 array	iary32	read only
cimtyp	equivalent CIM data type	integer	integer	read only
fmt	type of format	byte <sup>2</sup>	char	read only
hostlen	length of data type in host	integer	integer	read only
hstbuf	data in host format	byte array <sup>2</sup>	S_bary275	read/write
list	array of block numbers to mask from	integer array	iary32	read/write
nblk	number of blocks to define	integer*2 array	integer	read only
nnlst	array of data elements per block	byte <sup>2</sup>	bary16	read/write
nnnn	number of data elements	integer*2	integer	read/write
nnnnlst	array of data elements per block	integer*2 array	iary32	read only
pattern	pattern to fill memory with	byte array <sup>2</sup>	integer	read/write
ss	CIM function option code	integer*2 array	integer	read only
str128	error message	byte array <sup>2</sup>	string128	read only
taglst	array of addresses	integer*2	tag_array	read only
tibuf	conversion buffer	byte array <sup>2</sup>	S_bary275	read/write
tilen	length of data type in 500/505 format	integer	integer	read/write
<b>Diagnostics</b>				
nblks	maximum number of data acquisition blocks allowed	integer*2	integer	read/write
primlen	maximum primitive length	integer	integer	read/write
<b>Masks</b>				
cmask	CIM mask	integer*2	cim_mask	read only
mask	NIM mask	integer*2	mask_type	read only
<b>Network Access</b>				
xtn	transaction number	integer*2	integer	read only

<sup>1</sup> The Fortran declarations are determined according to Figure 3-1.

<sup>2</sup> MS-FORTRAN does not support type byte. All arguments declared as type byte above should be declared as integer for MS-FORTRAN.

<sup>3</sup> The Pascal declarations are determined according to Figure 3-2.

## Reference List of Arguments (continued)

Table 3-2 Argument Reference List for Fortran and Pascal (continued)

Argument	Definitions	Declaration (Fortran) <sup>1</sup>	Declaration (Pascal) <sup>3</sup>	Access
<b>Status</b>				
cstat	CIM status	byte <sup>2</sup>	byte	read/write
errtyp	error type	integer	integer	read/write
istat	composite error status returned for the call	integer	integer	write only
pstat	additional primitive status	integer	integer	read/write
retsts	returned status	byte <sup>2</sup>	byte	read/write
sstat	Secondary status	integer	integer	read only
state	state Secondary should be changed to	integer	integer	read/write
<b>File Transfer</b>				
fnam	transfer file name	byte array <sup>2</sup>	fname_typ	read only
flen	length of file name	integer	integer	read only
ftype	transfer request type	integer	integer	read only
pname	length of program name	integer	integer	read only
pnlcn	Unilink program name	byte array <sup>2</sup>	fname_typ	read only

<sup>1</sup> The Fortran declarations are determined according to Figure 3-1.

<sup>2</sup> MS-FORTRAN does not support type byte. All arguments declared as type byte above should be declared as integer for MS-FORTRAN.

<sup>3</sup> The Pascal declarations are determined according to Figure 3-2.

integer	=	16-bit integer
bary16	=	packed array [1..16] of byte;
bary275	=	packed array [1..275] of byte;
cim_mask	=	packed array [1..16] of boolean;
iary4	=	array [1..4] of integer;
iary32	=	array [1..32] of integer;
iary135	=	array [1..135] of integer;
iary254	=	array [1..254] of integer;
mask_type	=	packed array [1..32] of boolean;
tag_array	=	array [1..32] of tag ptr; (tag ptr = tag type);
tag_type	=	case integer of
		0 : array [1..5] of integer; (binary)
		1 : packed array [1..11] of character; (ASCII)
		2 : packed array [1..16] of character; (tag name)
		End;

Figure 3-1 Fortran Declarations

---

The Pascal declarations are determined according to the following list.

integers are 16-bit integer values.

```
bary 16    =   Packed Array [1..16] of byte;
bary275   =   Packed Array p1..275] of byte;
cim_mask  =   Packed Array [1..16] of Boolean;
iary4     =   Array [1..4] of integer;
iary32    =   Array [1..32] of integer;
iary135   =   Array [1..135] of integer;
iary254   =   Array [1..254] of integer;
mask_type =   Packed Array [1..32] of Boolean;
```

Case ARRAY\_OPTION of

```
  o_byte
  : (e_byte : bary275) ;
  o_int
  : (e_int : Array [1..69] of integer) ;
  o_real
  : (e_real : Array [1..69] of real) ;
End ;
```

```
tag_array =   Array [1..32] of tag_type ;
tag_type  =   Case tag_option of
  asci    : Packed Array [1..11] of char; (ASCII)
  binry   : Array [1..5] of integer; (Binary)
  tag_nam : Packed Array [1..16] of char; (Tag Name)
  End;
```

```
wiary256 =   Array [1..256] of word_integer;
```

Figure 3-2 Pascal Declarations

## Reference List of Arguments (continued)

Table 3-3 contains the Argument Reference List for BASIC and C/C++.

Table 3-3 Argument Reference List for BASIC and C/C++

Argument	Definition	Declaration (BASIC)	Declaration (C/C++) <sup>1</sup>	Access
<b>Addressing</b>				
addlst	binary tag	integer array	iary32	read only
hwy	highway number	integer	int	read only
mm	data element type	integer	char	read only
portstr	PC port setup string	byte array	bary275	read only
rtype	data element type	integer	int	read only
tag	address specification	integer array	tag_type	read only
tt	data element type	integer	int	read/write
<b>Buffer Specification</b>				
<i>Command buffers</i>				
cmd275	command buffer	integer array	bary275	read only
cmdlen	command buffer length	integer	int	read only
<i>Read buffers</i>				
ri4	response buffer	integer array	iary4	read/write
ri32	response buffer	integer array	iary32	read/write
ri135	response buffer	integer array	iary135	read/write
ri254	response buffer	integer array	iary254	read/write
rilen	response buffer length	integer	int	read/write
<b>Buffer Specification</b>				
<i>Response buffers</i>				
rsp275	response buffer	integer array	bary275	read/write
rsplen	response buffer length	integer	int	read/write
<b>Buffer Specification</b>				
<i>Write buffers</i>				
wblen	write buffer length	integer	char	read only
wb275	write buffer	integer array	bary275	read only
wi4	write buffer	integer array	iary4	read only
wi135	write buffer	integer array	iary135	read only
wilen	write buffer length	integer	int	read only

<sup>1</sup> The C declarations are determined according to Figure 3-3.

Table 3-3 Argument Reference List for BASIC and C/C++ (continued)

Argument	Definition	Declaration (BASIC)	Declaration (C/C++) <sup>1</sup>	Access
<b>Data Manipulation</b>				
cc	UCL instruction option code	integer	int	read only
cclst	array of block numbers to define	integer array	iary32	read only
cimtyp	equivalent CIM data type	integer	int	read only
fmt	type of format	byte	char	read only
hostlen	length of data type in host	integer	int	read only
hstbuf	data in host format	byte array	bary275	read/write
list	array of block numbers to mask from	integer array	iary32	read/write
nblk	number of blocks to define	integer array	int	read/write
nnlst	array of data elements per block	integer array	bary16	read/write
nnnn	number of data elements	integer	int	read/write
nnnnlst	array of data elements per block	integer	iary32	read only
pattern	pattern to fill memory with	integer array	int	read/write
ss	CIM function option code	integer	int	read only
str128	error message	byte array	string128	read only
taglst	array of addresses	integer	tag_array	read only
tibuf	conversion buffer	byte array	bary275	read/write
tilen	length of data type in 500/505 format	integer	int	read/write
<b>Diagnostics</b>				
nblks	maximum number of data acquisition blocks allowed	integer	int	read/write
primlen	maximum primitive length	integer	int	read/write
<b>Masks</b>				
cmask	CIM mask	integer	cim_mask	read only
mask	NIM mask	integer	mask_type	read only
<b>Network Access</b>				
xtn	transaction number	integer	int	read only

<sup>1</sup> The C declarations are determined according to Figure 3-3.

## Reference List of Arguments (continued)

Table 3-3 Argument Reference List for BASIC and C/C++ (continued)

Argument	Definition	Declaration (BASIC)	Declaration (C/C++) <sup>1</sup>	Access
<b>Status</b>				
cstat	CIM status	integer	char	read/write
errtyp	error type	integer	int	read/write
istat	composite error status returned for the call	integer	int	write only
pstat	additional primitive status	integer	int	read/write
retsts	returned status	integer	char	read/write
sstat	Secondary status	integer	int	read only
state	state Secondary should be changed to	integer	int	read/write
<b>File Transfer</b>				
fnam	transfer file name	byte array	fname_typ	read only
flen	length of file name	integer	integer	read only
ftyp	transfer request type	integer	integer	read only
pname	length of program name	integer	integer	read only
pnlen	Unilink program name	byte array	fname_typ	read only

<sup>1</sup> The C declarations are determined according to Figure 3-3.

---

The C declarations are determined according to the following list:

integer is a 16-bit word (Packing and unpacking of bits  
within integers is required)

char is an unsigned 8-bit byte

```
typedef char    bary16[16];
```

```
typedef char    bary275[275];
```

```
typedef char    cim_mask[16];
```

```
typedef int     iary4[4];
```

```
typedef int     iary32[32];
```

```
typedef int     iary135[135];
```

```
typedef int     iary254[254];
```

```
typedef char    mask_type [42];
```

```
typedef union
```

```
{
```

```
    int binary [5];
```

```
    char ascii [11];
```

```
    char name [16];
```

```
} tag_type ;
```

```
typedef tag_type tag_array[32];
```

Figure 3-3 C Declarations

### 3.4 List of Data Elements Types

---

Table 3-4 shows data element (TT) types for the controller.

Table 3-4 Data Element Types

Data Element Type <sup>1</sup> (in hex)	Format	Data Element Description
00 <sup>2</sup>	integer	Ladder (L) memory
01 <sup>2</sup>	integer	Variable (V) memory
02 <sup>2</sup>	integer	Constant (C or K) memory
03 <sup>2</sup>	Boolean	Discrete input, X memory
04 <sup>2</sup>	Boolean	Discrete output, Y memory
05 <sup>2</sup>	Boolean	Discrete control register (CR) memory
06	packed Boolean	X, packed
07	packed Boolean	Y, packed
08	packed Boolean	CR, packed
09 <sup>2</sup>	integer	WX
0A <sup>2</sup>	integer	WY
0B	Boolean	Discrete forced
0C	Boolean	CR forced
0D	integer	Word forced
0E	integer	Timer/counter preset
0F	integer	Timer/counter current
10	1	Drum step preset
11	1	Drum step current
12	1	Drum count preset
13	integer	Timer preset <sup>3</sup>
14	integer	Timer current <sup>3</sup>
15	integer	Counter preset <sup>3</sup>
16	integer	Counter current <sup>3</sup>
17	long-integer	Secondary system status
20	real	Loop gain
21	real	Loop reset
22	real	Loop rate

<sup>1</sup> See the individual NIM user manuals for the subset of data element types supported in each Secondary and format description.

<sup>2</sup> This data element type is available in CIM-based attached devices.

<sup>3</sup> These are used in attached devices (controllers) that do not have mutually exclusive numbered timers and counters.



Table 3-4 Data Element Types (continued)

Data Element Type (in hex)	Format	Data Element Description
23	real	Loop high alarm
24	real	Loop low alarm
25	real	Loop process variable
26	real	Loop high process variable limit
27	real	Loop low process variable limit
28	real	Loop orange deviation limit
29	real	Loop yellow deviation limit
2A	integer	Loop sample rate
2B	real	Loop setpoint
2C	real	Loop output
2D	integer	Loop status
2E <sup>2</sup>	long_integer	Loop C-flags status
2F	word_integer	Ramp/soak status
30	real	Loop error
31	real	Loop bias
32	real	Loop High-High Alarm Limit
33	real	Loop Low-Low Alarm Limit
34	real	Rate of Change Alarm Limit
35	word_integer	Loop Mode
36	<sup>1</sup>	Ramp/Soak step
37	real	Ramp Destination Setpoint
38	real	Ramp Rate of Change
39	real	Soak Time
3A	real	Soak Deadband
3B	<sup>1</sup>	Ramp/Soak Event Status bit
3C	real	A.A. High Alarm Limit
3D	real	A.A. Low Alarm Limit

<sup>1</sup> See the individual NIM user manuals for the subset of data element types supported in each Secondary and format description.

<sup>2</sup> The Series 500 loop C-flags are most directly related to, but not compatible with, the primitive data element type 2D, loop status.

word\_integer = two bytes  
 long\_integer = four bytes

## List of Data Elements Types (continued)

---

Table 3-4 Data Element Types (continued)

<b>Data Element Type (in hex)</b>	<b>Format</b>	<b>Data Element Description</b>
3E	real	A.A. Process Variable
3F	real	A.A. Process Variable High Limit
40	real	A.A. Process Variable Low Limit
41	real	A.A. Orange Deviation Alarm Limit
42	real	A.A. Yellow Deviation Alarm Limit
43	real	A.A. Sample Rate (seconds)
44	real	A.A. Setpoint
29	real	Loop yellow deviation limit
2A	real	Loop sample rate
2B	real	Loop setpoint
45	word_integer	A.A. V-Flags
46	long_integer	A.A. C-Flags
47	real	A.A. Error
48	real	A.A. High-High Alarm Limit
49	real	A.A. Low-Low Alarm Limit
4A	real	A.A. Rate of Change Alarm Limit
6X	real	IEEE Floating Pt. corresponds to 2X
7X	real	IEEE Floating Pt. corresponds to 3X
8X	real	IEEE Floating Pt. corresponds to 4X

## 3.5 Addressing

---

Address Elements	<p>Addressing in a TIWAY network has four elements:</p> <ul style="list-style-type: none"><li>• a highway (or network)</li><li>• a Secondary within the highway</li><li>• a data element type identifying the area of the Secondary memory</li><li>• a data element location identifying the starting address within the data element type</li></ul>
Highway Selection	<p>The communications link between a host computer and a TIWAY network can be configured:</p> <ul style="list-style-type: none"><li>• As point-to-point communications: the computer communicates through a single communications port to a single Unilink Host Adapter (UHA), controlling a single network.</li><li>• As several UHAs, each controlling a network, that are physically linked to the host computer through a single communications port.</li><li>• As several host computers that communicate through their respective communications ports to a single network.</li></ul> <p>A logical means for identifying each network is provided through the <code>hwy</code> argument. This argument is a logical means of identifying a specific UHA (TIWAY network) attached to a host computer communications port.</p> <p>The value of <code>hwy</code> is always set to 1 on the PC version of the TIWAY Host Software Package.</p>
Secondary Selection	<p>Each Secondary on a TIWAY network has a unique Secondary address in the range 01 to 254 (decimal). If the Secondary is attached to redundant media, this address is used for both media.</p>
Data Element Type Selection	<p>The data element type specifies a set of memory locations within a Secondary. The list of data element types is listed in Table 3-4. Secondaries using CIM interfaces (PM550s) use a different set of data element types than Secondaries using NIM interfaces; however, data element types are always specified as one of the NIM data element types, and the software converts that to the appropriate CIM type.</p>

## Addressing (continued)

---

Data Element Location Selection	<p>After the data element type has been specified, an individual piece of data is selected by a data element location. When a single address must reference a block of data, the base data element location specifies the first address of a block of data. For the 5TI NIM, PM550 NIM, and Series 500 NIM (Rel. 1.0), the data element location is a 16-bit value. However, for the Series 500/505 NIM (Rel. 2.1) that supports extended memory, the data element location may be a 16-bit (non-extended addressing) or 32-bit (extended addressing) value. Data element type and data element location must be specified if you plan to use any memory functions.</p>
Addressing Arguments	<p>TIWAY Interface Subroutines generally require all four address elements: the highway, the Secondary, the data element type, and the data element location. The argument for the full address is the tag argument. The full-address tag argument is called the long form of the tag throughout this document.</p> <p>Some subroutines require only a portion of the full tag argument. Subroutines that need the highway and the Secondary (but not the data element type and data element location) are identified as needing only the short form of the tag. (Subroutines that require the short form of the tag discard both the data element type and data element location if they are provided.) Subroutines that address all Secondaries on a specified highway use only the hwy argument and need not include the Secondary, the data element type, or the data element location.</p> <p>Three CIM Functional Command Subroutines further define the tag argument with either the mm argument or the rtype argument. The mm argument specifies a subset of the CIM data element types, called memory areas. The rtype argument identifies which of three possible data types (display only, tuning only, or display and tuning) is to be returned. These extensions to the tag are described in Chapter 10 with the subroutine arguments for CIMDNL, CIMUPL, and RDLOOP.</p>
Address Specification	<p>After an address has been determined, it can be specified in one of three ways: as an alphanumeric name (stored in a tag table), as an ASCII string, or as a binary number. These specification methods are discussed in the following sections.</p>

---

**Tag Name  
Specification**

The TIWAY Host Software Package supports a tag table, which consists of a set of alphanumeric tag names and the physical address that corresponds to each tag name. Any address that does not begin with a “#” or “/” (ASCII identifiers) or a null (binary identifier) is assumed to be a tag name. Tag names and the tag tables are discussed in further detail in Chapter 8.

A tag name is a string of ASCII characters (but it cannot begin with a “#” or “/”). It must either be as long as the tag name length specified at installation or be terminated in an ASCII null character.

Keep in mind that the time difference between looking up tag names and converting them from one of the direct specification forms is very short compared to the length of a single network transaction. Therefore, as a general practice, use the form that is most logical for an application instead of one that is based on time efficiency. Only severe limitations, such as insufficient memory for the tag table, may force you to use an alternative that is not the most logical.

**ASCII Specification**

Specifying the address directly as an ASCII string is an alternative that allows you to specify addresses that do not appear in the tag table and is particularly applicable to interactive use. The ASCII string for non-extended addressing consists of 11 bytes of ASCII data. The first byte of the character string specified is a “#” or “/”, followed by the highway, Secondary address, data type, and data location. The format follows:

#HHSSTTAAAA

where:

#	=	signifies ASCII specification follows
HH	=	Highway number in hex (2 hex digits)
SS	=	Secondary address in hex (2 hex digits)
TT	=	Data element type in hex (2 hex digits)
AAAA	=	Data element location in hex (4 hex digits)

If extended addressing is being accessed, the address can be specified by a similar ASCII character string. The first byte of the character string specified is a “/”, followed by the highway, Secondary address, data type, and data location. The format follows:

/HHSSTTAAAAAAA

where:

HH	=	Highway number in hex
SS	=	Secondary address in hex
TT	=	Data element type in hex
AAAA AAAA	=	Data element location in hex

## Addressing (continued)

---

The short form of an address can be generated simply by omitting the data element type and the data element location. For instance, #0101 would be a sufficient address specification for highway 1, Secondary 1, if you were issuing a Secondary diagnostic command. If you wanted to read data, however, you would have to specify both the data element type and data element location.

### Binary Specification

Specifying the address directly in binary is useful for applications in which the program steps through a set of consecutive address items under program control. For instance, for a program to poll all Secondaries, a loop could be used with the sec word of the binary specification as the index variable. Binary specification, however, is not appropriate for interactive use.

For non-extended addressing in binary, the four address elements (highway, Secondary, data element type, and data element location) are specified as an array of integers of the form:

0	= integer zero for non-extended binary specifications
HWY	= integer containing highway number (hh)
SEC	= integer containing Secondary address (ss)
TYP	= integer containing data element type (tt)
ADDR	= integer containing data element location (aaaa)

For extended addressing in binary the four address elements are specified as an array of integers of the form:

1	= integer one for extended binary specifications
HWY	= integer containing highway number (hh)
SEC	= integer containing Secondary address (ss)
TYP	= integer containing data element type (tt)
ADDR	= integer containing the low-order word (16 bits) of the data element location (aaaa)
ADDR	= integer containing the high-order word (16 bits) of the data element location (aaaa)

For example, the following Fortran program segment creates a binary specification, which references highway 2, Secondary 5, and location V100 on a 530.

```
TAG(1) = 0
TAG(2) = 2
TAG(3) = 5
TAG(4) = 1
TAG(5) = 100
```

## 3.6 Buffer Specification

---

The names of buffer arguments used by the TIWAY subroutines are designed to make the buffer's function as easy to recognize as possible. The following conventions apply:

- The first unit in each buffer name indicates the buffer's function with respect to the network: `cmd` indicates a command buffer, `rsp` indicates a response buffer, `r` indicates a read buffer, and `w` indicates a write buffer.
- The second letter in read and write buffers indicates the format of the data contained in the buffer, with `i` standing for integer and `b` standing for byte.
- If the last three letters in the buffer name are `len`, that buffer specifies the buffer length.
- Digits, when given, indicate the optimum length for the buffer.

Buffers, used by the TIWAY subroutines to return data to the application program, use both a standard length and buffer format, and the buffer arguments usually occur in pairs (for instance, `rsplen`, `rsp275`). The length argument has two distinct uses: On entry to TIWAY, the length specifies the maximum buffer length; to avoid buffer overflow, the TIWAY Subroutine does not return more than that amount of data. On return, the TIWAY Subroutine returns the length actually used.

---

**NOTE:** Do not use a constant for the length of a buffer in which the TIWAY Subroutines returns data.

---

<b>Command Buffers</b>	The TIWAY Subroutine Library has two command buffer arguments, <code>cmdlen</code> and <code>cmd275</code> . <code>Cmdlen</code> specifies, in integer format, the number of bytes to be contained in <code>cmd275</code> . <code>Cmd275</code> contains, in an array of bytes, the command built by the user.
<b>Response Buffers</b>	The response buffer arguments are <code>rsplen</code> and <code>rsp275</code> . <code>Rsplen</code> specifies, in integer format, the number of bytes to be contained in <code>rsp275</code> . <code>Rsp275</code> contains, in an array of bytes, the response returned to the call. (All response buffers are byte oriented.)

## Buffer Specification (continued)

---

- Read Buffers**                    The read buffers, *ri1en*, *ri4*, *ri32*, *ri135*, and *ri254*, are those that you read (returned by the call to the **TIWAY** subroutine). *ri1en* specifies, in integer format, the number of integers to be contained in *ri4*, *ri32*, *ri135*, or *ri254*. The other buffers contain, in an array of integers, the data returned by the call to the subroutine.
- Write Buffers**                    The write buffers, *wilen*, *wblen*, *wi4*, *wi135*, and *wb275*, are read by the computer (written by you). *wilen* specifies, in integer format, the number of integers to be contained in *wi4* or *wi135*. *wblen* specifies, in byte format, the number of bytes to be contained in *wb275*. *wi4* and *wi135* contain, in an array of integers, the data being sent to the network. *wb275* contains the data being sent to the network in an array of bytes.



### 3.7 Data Block Manipulation

Many of the subroutines in the TIWAY Subroutine Library allow you to perform I/O operations: to obtain, modify, or replace data. The subroutines also allow you to manipulate data or blocks of data. The following arguments are used to manipulate blocks of data.

**UCL Instruction Code (cc Argument)**

The code (cc) argument defines which of four memory storage actions should take place in a NIM-based attached device during a write-buffer operation (the WRBUF subroutine). Consult the description of WRBUF for a description of these actions.

**Block List (cclst Argument)**

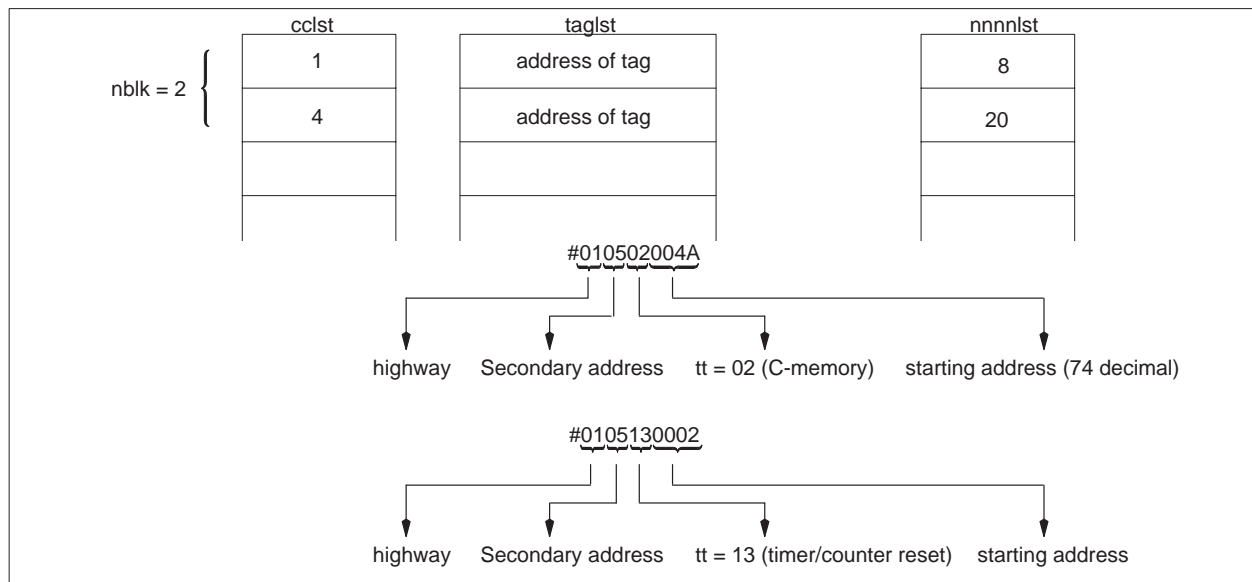
The block list (cclst) argument is an array in which each array element contains a block number.

Example:

If you want to define the following two data acquisition blocks in PM550 number 5 on highway number 1:

- block 1: 8 words starting at C74
- block 4: 20 timer presets starting with timer 2

You initialize the DEFBLK argument as shown below:



The actual subroutine call would be as follows:

```
defblk (&istat, &xtn, "#0105", &nblk, cclst, taglst, nnnlst, &sstat);
```

## Data Block Manipulation (continued)

---

Number of Blocks (nblk Argument)	The number of blocks (nblk) argument specifies how many data acquisition blocks are to be defined. The argument is also used in RDRAND and WRRAND to specify the number of data groups to be transferred.
Number of Data Element Locations (nnnn Argument)	The nnnn argument is used to specify how many data element locations to write, read, or transfer.
Number of Data Elements List (nnnnlst Argument)	The nnnnlst argument specifies the number of data elements to return for a defined block. It is an array of numbers specifying the number of elements for each of a series of blocks to be transferred.
Pattern (pattern Argument)	The pattern argument, used only in the FILL subroutine, is an integer field that fills a consecutive block of memory on a specified NIM-based attached device. The pattern argument is always a word in the host format that is converted to 500/505 format before being sent to the attached device. The valid range of values for pattern is 0 to 65535.
CIM Command Code Specifier (ss Argument)	The CIM Command Code specifier argument defines which of four operations is to take place in a CIM-based attached device during uploading or downloading of data. Specific values for ss are discussed in Chapter 10.
Tag List (taglst Argument)	<p>The taglst argument allows an array of addresses to be specified at one time. taglst is an array of addresses, with each element of the array containing the address of a corresponding tag. See Chapter 8 for information on building tag tables.</p> <p>The arguments in question include aaaalst, blklst, nrecs, nrecs1, nrecs2, numblks, recblks, recnums, reclst1, reclst2, and ttlst.</p>

---

**NOTE:** Many other arguments listed in Table 3-2 and Table 3-3 are not discussed in this section because they are not general in nature, but are specific to a certain routine. These arguments are discussed within the context of the routine description in Chapter 10.

---

## 3.8 Diagnostics

---

The `nblks` argument (number of blocks available) and `primlen` argument (primitive length) are used only in the `GETLEN` subroutine, a diagnostic subroutine that returns the maximum primitive length and the maximum number of data acquisition blocks supported by a specified NIM-based attached device. The numbers are returned as integers in the host format, with the maximum primitive length in `primlen` and the maximum number of data acquisition blocks available in `nblks`.

Five of the subroutines in the TIWAY Subroutine Library (GATHER, WRTGAT, RNDRD2, RNDRD3, and RNDRD4) allow you to read information using a mask. A mask selects which data acquisition blocks are to be read. In languages other than Pascal, the mask can be built using the BLDMSK subroutine shown in Chapter 5. When using Pascal, the mask must be constructed within the application's code.

If you need to construct a mask, the mask is defined (if BLDMSK is not used) in host format with the least-significant bit representing block 1 and the most-significant bit representing block 32. The software reverses this order before using the mask in the command buffer. This order is opposite to that used on the network, but it is done this way to facilitate the Pascal programmer using a packed array of Boolean, and to be more convenient for the programmer using INTEGER\*4 mask representation.

The `mask` argument for NIM-based attached devices is a 32-bit mask.

The `cmask` argument for CIM-based attached devices is a 16-bit mask, each bit corresponding to a block number that data are to be gathered from. The mask returned by BLDMSK is a 32-bit mask. To isolate the 16-bit CIM mask from the 32-bit mask, use only the lowest significant word of the 32-bit mask.

## 3.10 Network Access

---

By default, the TIWAY Interface subroutines provide synchronous completion (the subroutines issue I/O to the Host Adapter and wait for the response before returning to the application program). However, the TIWAY Subroutine Library also supports overlapping processing between the task and the I/O or issuing the I/O request manually. The type of network access is specified in the `xtn` argument.

Any subroutine that issues I/O to the Host Adapter actually has two parts: first, the subroutine formats a buffer and issues the I/O request; and second, the subroutine completes the I/O request and returns status and data.

In some cases, synchronous completion is not desirable. The `xtn` argument, therefore, allows for asynchronous completion. In asynchronous completion, the buffer is formatted and the I/O request is sent, but the status and data are not returned until you call for them (using the `TIXTN` or `TIXTNW` subroutine).

The transaction number determines whether the second half of the subroutine is executed (whether the status and data is returned automatically as soon as they are available). Two transaction numbers are available: 0 and -1. If the `xtn` argument is 0 (synchronous I/O), both the first and second half of the subroutine is executed. If the `xtn` argument is -1 (asynchronous I/O), only the first half of the subroutine is executed before control returns to you. See Table 3-5. You must later issue another call for the response, using the `xtn` argument value returned by the first half call.

Table 3-5 Accessing the Network with `xtn`

<b>xtn</b>	<b>Response</b>
0	The I/O request is sent. The response to the request is returned.
-1	The I/O request is sent. Control returns to you. You must issue the <code>TIXTN</code> or <code>TIXTNW</code> subroutine to read the response.

Synchronous and asynchronous completion, buffered I/O, and `TIXTN` and `TIXTNW` subroutines, are discussed in Chapter 10.

## 3.11 Status Determination

---

The TIWAY Host Software Library allows you to determine the status of several different variables. Most significant, and common to every subroutine, is the `istat` argument.

### Composite Error Reporting (`istat` Argument)

Each TIWAY subroutine call returns a single 2-byte integer status. The returned status contains the facility number and the message number within that facility, with the facility in the high byte and the error in the low byte. For example, if a host adapter cable is not connected, a network transaction initiated by a TIWAY subroutine call “times out”, and the `istat` value returned equals 1794 decimal or 0702 hex. This is interpreted as Facility 07, Message 2, which reads “Device timeout” (see Appendix E).

In addition to the status information provided in `istat`, a message generation facility and a message reporting facility provide a mechanism to convert a status code into a message and print it on the operator’s terminal.

GETMSG subroutine can be used to retrieve this error message, while the PUTMSG subroutine displays it (See Chapter 5 for GETMSG and PUTMSG details. The error codes and their meanings are given in Appendix E.) In addition, TIUSER and TIPROG display these messages if an error occurs during execution of a utility command.

The status represents either success or the first fatal error detected. If an error is detected, processing ceases and the status is returned with the value representing the error. If the subroutine call is successful, then the returned status is zero.

All errors are considered fatal errors and when they occur, the subroutine will stop running and return the error code without completing the rest of the subroutine’s functionality.

A single TIWAY call can have as many as nine independent sources of errors, each with several possibilities for error, each with overlapping sets of error numbers. To aid the TIWAY programmer in identifying and correcting errors, error reporting for the TIWAY Subroutine Library occurs as a detailed set of status reporting cases. Nine facilities are provided, as shown in Table 3-6.

Table 3-6 Facility Error Numbers

Facility #	Type of Error
1	TIWAY (detected by the TIWAY Subroutine Library)
2	NETEXCEPT (detected by the Host Adapter, network exception)
3	INTERNEXCP (detected by the Host Adapter, adapter internal exception)
4	HOSTEXCEPT (detected by the Host Adapter, host exception)
5	PRIMITIVE (detected by NIM-based attached devices while processing a TIWAY Primitive)
6	OPERATING SYSTEM (detected by the operating system while processing an operating system request)
7	I/O STATUS (detected by the device driver while processing I/O)
8	CIMEXCEPT (detected by CIM-based attached devices while processing a CIM Functional Command)
9	NATIVE TASK CODES (detected by the Secondary being accessed)

Error Type (errtyp Argument)

The error type `errtyp` argument, used only in the XPAR subroutine, contains a code that indicates the amount and type of error-checking to be performed. Possible codes are shown in the Notes on Call Format section of the XPAR subroutine in Chapter 10.

CIM Status (cstat Argument)

The CIM status field contains additional status as defined by the `ii` field in the response buffer. Refer to the *CIM User Manual* for a more complete description of potential responses. The following status reporting conventions assist you in identifying the source of error.

## Status Determination (continued)

---

### Secondary Status (sstat Argument)

The Secondary status argument contains the code for the current state of a specific NIM-based attached device. Current state means the state the device is in when the response to the subroutine call is returned. Current state is returned for all attached devices except Series 500 NIMs, Release 1.1 and earlier. (For these Series 500 NIMs, the state that is returned is the state that existed before the subroutine call was issued.) Table 3-7 shows the seven possible values.

Table 3-7 Seven Possible Subroutine Values

<b>Value</b>	<b>Definition</b>
00	Operational and performing instruction data type and loop execution
01	Operational and performing instruction data type and loop execution with a non-fatal* error detected
02	Operational and not performing instruction data type execution with loop execution
03	Operational and not performing instruction data type or loop execution
04	Operational and not performing instruction data type execution with loop execution and a non-fatal* error detected
05	Operational and not performing instruction data type or loop and a non-fatal* error detected
80	Not operational due to a fatal error condition

\*Fatal and non-fatal errors are defined by each unique device.



# TIWAY Interface Subroutines – Topical

---

<b>4.1</b>	<b>Overview</b> .....	<b>4-2</b>
<b>4.2</b>	<b>Session Control Subroutines</b> .....	<b>4-5</b>
<b>4.3</b>	<b>Computer Port Setup Initialization</b> .....	<b>4-6</b>
	INIT Subroutine Initialization .....	4-6
	Call Format .....	4-6
	Notes on Call Format .....	4-6
<b>4.4</b>	<b>Host Adapter Command Code Subroutines</b> .....	<b>4-7</b>
	ACTVAT .....	4-8
	ADIAG .....	4-8
	BRDCST .....	4-8
	DEACT .....	4-9
	POLL .....	4-9
	SDIAG .....	4-9
	SECLOG .....	4-10
	XPAR .....	4-10
<b>4.5</b>	<b>TIWAY Primitive Subroutines</b> .....	<b>4-11</b>
	CHNGST .....	4-12
	CONFIG .....	4-12
	DEFBLK .....	4-12
	FILL .....	4-12
	GATHER .....	4-12
	GETLEN .....	4-13
	NATIVE .....	4-13
	RDSTS .....	4-13
	TIGET .....	4-13
	TIPUT .....	4-13
	TIREAD .....	4-14
	TIWRIT .....	4-14
	WRBUF .....	4-14
	WRTGAT .....	4-14
<b>4.6</b>	<b>CIM Functional Command Subroutines</b> .....	<b>4-15</b>
	CCUSTS .....	4-16
	CIMDNL .....	4-16
	CIMRD .....	4-16
	CIMUPL .....	4-16
	CIMWR .....	4-16
	RDLOOP .....	4-16
	RNDRD1 .....	4-16
	RNDRD2 .....	4-17
	RNDRD3 .....	4-17
	RNDRD4 .....	4-17

To assist application programmers, the TIWAY Interface Subroutines provide high-level network services through the subroutine call facility. This set of subroutines must be linked with the applications program. The classes of subroutines available in this library include:

- **Session Control Subroutines.** These subroutines must be called by the application program to initiate and terminate the use of the TIWAY library subroutines. See Section 4.2.
- **Host Adapter Command Code Subroutines.** These subroutines correspond directly to the Host Adapter Command Codes. The data necessary to create a Host Adapter Command Code command buffer and the pertinent data to be returned by the response to that command code are passed as arguments to the subroutine call. See Section 4.4.
- **TIWAY Primitive Subroutines.** These subroutines correspond directly to a subset of the TIWAY Primitives. The data necessary to create a TIWAY Primitive command buffer and the pertinent data to be returned by the response to that Primitive are passed as arguments to the subroutine call. See Section 4.5.
- **CIM Functional Command Subroutines.** These subroutines correspond directly to a subset of the CIM Functional Commands. The data necessary to create a CIM Functional Command buffer and the pertinent data to be returned by the response to that Functional Command are passed as arguments to the subroutine. See Section 4.6.

The details on each specific function are presented in alphabetical order in Chapter 10.

---

In this manual, the calling conventions for each function are shown in the following six formats.

- The Fortran and BASIC call through the TIWAY entry point.
- The Fortran and BASIC call directly to the TIWAY Interface subroutine.
- The Pascal call through the TIWAY entry point.
- The Pascal call directly to the TIWAY Interface Subroutine.
- The C call through the TIWAY entry point.
- The C call directly to the TIWAY Interface subroutine.

The information provided is intended to be applicable to the implementation of TIWAY applications programs on any supported host computer. However, some arguments are system-specific. These arguments are flagged in the Notes on Call Format section.

---

**NOTE:** More details are provided on each function in the Reference Section in alphabetical order (Chapter 10).

---

## Overview (continued)

---

Examples of TIWAY Subroutine calls are:

BASIC,        CALL TIWAY ("TIGET", istat, xtn, counter32, nnnn, rsplen, ri135, sstat)  
FORTRAN:     CALL TIGET (istat, xtn, counter32, nnnn, rsplen, ri135, sstat)

Pascal:        TIWAY ("TIGET", istat, xtn, counter32, nnnn, rsplen, ri135, sstat);  
               TIGET (istat, xtn, counter32, nnnn, rsplen, ri135, sstat);

C:             tiway ("tigt", &istat, &xtn, counter32, &nnnn, &rsplen, &ri135, &sstat);  
               tiget (&istat, &xtn, counter32, &nnnn, &rsplen, &ri135, &sstat);

This call reads a specified number of consecutive pieces of data from the location specified by the address, in this case the tag name `counter32`. The data is returned in the response buffer array `ri135` and a composite status is returned in `istat`. The services described below are performed.

1. A Host Adapter Command Code 01 command buffer is formatted, and a TIWAY Primitive 20 command is appended to the command buffer by the call to TIGET.
2. The tag name, `counter32`, is converted to a highway, Secondary, data element type, and data element location to identify the data to be returned. `Counter32` is defined in the tag table with its appropriate tag specification (see Chapter 8 for further discussion).
3. The number of data to return `nnnn` is appended to the command buffer.
4. The maximum length of the response buffer in bytes, indicated in `rsplen`, is passed to the device driver.
5. The request is issued to the unit of the Host Adapter device driver (each unit corresponds to a particular Host Adapter) through an I/O request native to the operating system (the `xtn` argument).
6. The response is checked for the following errors:
  - A directive error reported when the I/O request is issued.
  - A device driver error representing a link level error.
  - Any error in one of the three sets of Host Adapter error codes.
  - A Primitive error code.
7. Any detected errors are reported in `istat`.
8. If no errors were detected, the data portion is returned to the caller's buffer, `rsp275`, and the length of the response is returned to `rsplen`.
9. The Secondary-status argument contains the code for the current state of a specific NIM-based attached device. Current state means the state the device is in when the response to the subroutine call is returned. Current state is returned for all attached devices except Series 500 NIMs, Release 1.1 and earlier. (For these Series 500 NIMs, the state

that is returned is the state that existed before the subroutine call was issued.) Table 3-7 shows the seven possible values.

## 4.2 Session Control Subroutines

---

The subroutines in the TIWAY Subroutine Library and the TIWAY Support Subroutine Library require some initialization before you can use them. This includes some data structure initialization and access to the site-specific tag table. To perform this initialization, call the INIT subroutine.

The complement of INIT is the FIN (finish) subroutine, which can be used to release the system resources reserved during the INIT call. If a program exits or aborts, you do not need to call FIN.

INIT reserves system resources and initializes the port used by the software package.

FIN is the complement of the INIT subroutine. It releases system resources reserved during the INIT call. After FIN has been called, none of the other subroutines can be called until an INIT is called.

## 4.3 Computer Port Setup Initialization

---

**INIT Subroutine Initialization**      You must always invoke the INIT subroutine first, in your application program, in order to use the host TIWAY adapter library. INIT has different arguments, depending on the specific host package involved. For the PC TIWAY package, the INIT routine is defined as follows:

**Call Format**

BASIC:      CALL TIWAY ("INIT", istat, xtn, portstr)  
FORTRAN:    CALL INIT (istat, xtn, portstr)

Pascal:      TIWAY ("INIT", istat, xtn, portstr);  
              INIT (istat, xtn, portstr);

C:            tiway ("init", &istat, &xtn, &portstr);  
              init (&istat, &xtn, &portstr);

**Notes on Call Format**

Explanations of the terms used in the call format follow.

istat – The status argument must be specified, but it may be specified as a null argument in FORTRAN.

xtn – The transaction number must be zero.

portstr – Portstr is used to select and initialize the appropriate communications port. This must be an ASCII string specifying the port number, baud rate, number of data bits, number of stop bits, parity, number of retries, and length of time-out in hex. The following is an example of a port set-up string:

"P1,19200,7,1,E,3,A"

This string specifies port 1, 19200 baud; 7 data bits and 1 stop bit (standard for the host adapter); even parity, 3 retries, and a time-out of 10 (hex A) seconds per try.

The maximum time-out period is 15 seconds. The port set-up string must be fully specified in order for the TIWAY host package to communicate with the network.

## 4.4 Host Adapter Command Code Subroutines

---

Each Host Adapter Command Code subroutine corresponds to a Host Adapter Command Code. The intent is to provide the application programmer with a natural interface to the command codes. The arguments correspond directly to information required to build the request buffer, or data returned by the host adapter in the response to the request. For additional information on Host Adapter Command Codes, see the *SIMATIC TIWAY I Unilink Host Adapter User Manual* (PPX:TIWAY-8121-x).

Five Host Adapter Command Codes are not represented by these subroutines: commands 01, FC, FD, FE, and FF. Response code 00 is returned with error responses and does not require representation. All primitive commands are embedded within a command code 01 and, again, representation is not required. 01 access can be achieved through use of the XPAR routine. Command codes FC, FD, FE, and FF (reset host adapter) are not represented, but can also be accessed through the XPAR routine. Table 4-1 shows the Host Adapter Command Code subroutines.

---

**NOTE:** You must call the INIT subroutine before any other subroutine can be executed.

---

Table 4-1 Host Adapter Command Code Subroutines

Subroutine	Command Code	Description
	00	Returned with error responses.
	01	Used by the NIM and CIM subroutines to send network data
BRDCST	02	Broadcasts a user-specified command buffer on the specified highway.
POLL	03	Polls a specified Secondary for its response to most recently issued broadcast command.
ACTVAT	04	Logically connects (activates) a Secondary or list of Secondaries to the specified highway.
DEACT	05	Logically disconnects (deactivates) a Secondary or list of Secondaries from the specified highway.
SECLOG	06	Returns a list (log) of Secondaries currently connected to the specified highway.
SDIAG	07	Returns a list of statistics (diagnostics) collected by the Host Adapter about a specific Secondary.
ADIAG	08	Returns a list of statistics (diagnostics) collected by the Host Adapter about the Host Adapter's network usage.
XPAR	all	Provides high-level direct access to the Host Adapter command/response buffers.



## Host Adapter Command Code Subroutines (continued)

---

ACTVAT	Before any communication is allowed with a Secondary, the Secondary must be logically connected to the network, using the ACTVAT subroutine. After the Secondary has been connected, the host computer can communicate freely with it. If a Secondary is connected to the highway when the call is issued, it can be included in <code>wi4</code> and is listed as successfully connected. ACTVAT is the implementation of command code 04.
ADIAG	The ADIAG subroutine returns a list of statistics (diagnostics) collected by the Host Adapter about the Host Adapter's network usage. ADIAG is the implementation of command code 08.
BRDCST	The BRDCST subroutine broadcasts the user-specified command buffer to all Secondaries on the specified highway. The command buffer must be one that all the Secondaries can interpret. Using BRDCST lowers the overhead associated with long command buffers that are issued to several Secondaries. BRDCST is the implementation of command code 02.

---

NOTE: The Secondaries do not immediately respond to the broadcast; the individual responses must be solicited by using POLL.

PM550 CIMs (PPX:PM550-502 and PPX:PM550-503) do not support the BRDCST subroutine and ignore all broadcast messages. PM550 NIMs (PPX:PM550-5038, PPX:PM550-5039, and PPX:PM550-5040) support BRDCST.

The BRDCST routine is intended to be immediately followed by the POLL routine. Calling another TIWAY routine after BRDCST and before POLL causes an invalid response (i.e., the Secondary attempts to answer the new routine as if it were responding to the POLL request).

When using BRDCST and POLL through a UHA, refer to the *Unilink Host Adapter User Manual*, PPX:TIWAY-8121-x, for the implementation of these commands for your specific UHA mode.

---

---

**DEACT**

The DEACT subroutine logically disconnects a Secondary or list of Secondaries from the specified highway. Disconnecting a Secondary removes that Secondary's entry from the Host Adapter's Secondary log, preventing any communications between that Secondary and the Host Adapter until the Secondary is again connected to the Host Adapter with the ACTVAT subroutine. DEACT is the implementation of command code 05. The DEACT command results in the issuing of an HDLC DISC to the Secondary depending on the configuration of the UHA.

---

**NOTE:** A Secondary is successfully disconnected only if it is connected and functioning properly when DEACT is issued. When a NIM is disconnected, it resets (the test mode is entered and the NIM executes its startup sequence). A 500/505 NIM attached to a 560 or 565 has its memory configuration table updated at this time.

---

**POLL**

The POLL subroutine polls the specified Secondary for the response to the broadcast it received most recently. POLL solicits a response only from the single specified Secondary. POLL is the implementation of command code 03.

---

**NOTE:** The POLL routine is intended to immediately follow the BRDCAST routine. Calling another TIWAY routine after BRDCAST and before POLL causes an invalid response (i.e., the Secondary attempts to answer the new routine as if it were responding to the POLL request).

When using BRDCAST and POLL through a UHA, refer to the *Unilink Host Adapter User Manual*, PPX:TIWAY-8121-x, for the implementation of these commands for your specific UHA mode.

---

**SDIAG**

The SDIAG subroutine returns a list of statistics (diagnostics) collected by the Host Adapter about a specified Secondary. The statistics contain data about that Secondary's network usage and errors. SDIAG is the implementation of command code 07.

## Host Adapter Command Code Subroutines (continued)

---

**SECLOG**                    The SECLOG subroutine returns a list of all Secondaries currently connected to the specified highway. SECLOG is the implementation of command code 06.

---

**NOTE:** The list returned from SECLOG represents only the existence of an entry in the Host Adapter Secondary log table for that Secondary, not that the Secondary is currently available. Secondaries not disconnected (DEACT routine), remain in this list even if the Secondary is physically disconnected or powered down.

---

**XPAR**                    The XPAR subroutine provides a simple subroutine interface for formatting a command buffer, sending it to a Host Adapter (and by proper format, through a Host Adapter to any connected Secondary), and receiving a copy of the entire response. This transaction gives you the ability to use facilities that are not supported by other TIWAY Subroutines (TIWAY Primitives that are not supported, CIM Functional Commands that are not supported, and all Task Codes). A specific example is the Host Adapter Reset Command Code FF.

XPAR provides two services. First, the command and response buffers are formatted to contain the data to be interpreted by the Host Adapter without including the link level protocol encapsulation (NITP or BDLC protocol packet). Second, optional error detection is provided for Host Adapter, TIWAY Primitives, and CIM Functional Commands. (The CIM Functional Command error-checking is also suitable for PM550 CCU Task Codes.) Use an error-checking argument to determine the amount and type of error-checking performed.

The XPAR simply generates the message delimiters, the character count and the checksum for the message body you supply.

---

**NOTE:** You should be familiar with the Host Adapter and any addressed Secondaries before using the XPAR subroutine.

---

## 4.5 TIWAY Primitive Subroutines

---

Each TIWAY primitive subroutine corresponds with a TIWAY Primitive. The intent is to provide the applications' programmer with a natural interface to each primitive. The arguments correspond directly either to information required to build the request buffer or to data returned in the response to the request. All TIWAY primitive subroutines (except for GETLEN and GETCON subroutines) include the status of the destination Secondary in the response (*sstat* argument). The TIWAY primitive subroutines are shown in Table 4-2. Values in the Primitive column are given in hex.

Table 4-2 TIWAY Primitive Subroutines

Subroutine	Primitive	Description
CHNGST	10	Changes operational state of a Secondary.
CONFIG	03	Reads memory configuration of the specified Secondary.
DEFBLK	50	Defines data to be returned from one or more data acquisition blocks.
FILL	32	Fills consecutive section of memory in attached device.
GATHER	51	Gathers data from previously defined data acquisition blocks.
GETLEN	04	Gets the maximum primitive length and the maximum number of data acquisition blocks supported by the specified attached device.
NATIVE	01	Bypasses the normal primitive processing and sends a native Task Code directly to the attached device.
RDSTS	02	Reads status of a specified NIM-based attached device.
TIGET	20	Reads consecutive memory locations from an attached device and converts data from 500/505 to host format.
TIPUT	30	Writes consecutive memory locations to an attached device and converts data from host to 500/505 format.
TIREAD	20	Reads consecutive memory locations from an attached device (500/505 format).
TIWRIT	30	Writes consecutive memory locations in an attached device (500/505 format).
WRBUF	33	Writes consecutive memory locations into an attached device that is equipped with an L-memory buffer.
WRTGAT	52	Combines the functions of TIWRIT and GATHER.

---

**NOTE:** All NIM primitive subroutines use the host adapter command code 01. Before attempting to use these subroutines, you should be familiar with the primitives you intend to use. Refer to the NIM user manuals for more information. You must successfully execute the INIT subroutine before any other subroutine can be executed.

---

## TIWAY Primitive Subroutines (continued)

---

---

**NOTE:** There are a few differences between how data is stored within a Secondary and how data is stored within the personal computer. Read Appendix G to understand this difference and how it is treated by TIWAY subroutines.

---

### CHNGST

The CHNGST subroutine is used to change the operational state of a specified NIM-based attached device. CHNGST is the implementation of Primitive 10. This changes the operational state of a controller, either from STOP to RUN or vice versa, and must be selected carefully. See Warning below.

 <b>WARNING</b>
--

Remote state changes can be dangerous in some installations.

Remote state changes could cause unpredictable controller behavior that could result in death and/or serious injury, or damage to equipment.

Do not use this routine without thoroughly understanding how it might impact operations directed by the controller.

### CONFIG

The CONFIG subroutine returns the configuration (memory sizes) of the specified NIM-based attached device. CONFIG is the implementation of Primitive 03.

### DEFBLK

The DEFBLK subroutine is used to define controller memory locations. See the applicable NIM user manuals for a complete description of the data acquisition primitives. DEFBLK is the implementation of Primitive 50.

### FILL

The FILL subroutine is used to fill a number of contiguous data elements starting at a specified address. FILL is the implementation of Primitive 32.

### GATHER

The GATHER subroutine is used to gather data from previously defined data acquisition blocks (see the DEFBLK subroutine). The GATHER subroutine returns the data in the 500/505 data format. Notice that data from multiple blocks can be returned in the same call, so multiple data formats may be returned in the same buffer. GATHER is the implementation of Primitive 51.

---

**GETLEN**

The GETLEN subroutine obtains and displays the maximum primitive length and the maximum number of data acquisition blocks supported by the specified Secondary. GETLEN is the implementation of Primitive 04.

---

**NOTE:** GETLEN is the only TIWAY Primitive routine that does not have a Secondary status (sstat) argument.

---

**NATIVE**

The NATIVE subroutine allows you to bypass the normal primitive processing and execute a native Task Code directly. Primitives are high-level commands that are not specific to any controller model; instead, NIMs use the Task Code commands that are native to the attached controller to perform the function requested by the primitive.

The NATIVE subroutine accepts a Task Code command buffer, appends it to the Host Adapter Command Code and TIWAY Primitive length and Primitive 01, and passes it to the Secondary. The response, without Host Adapter and TIWAY primitive overhead, is returned in the response buffer. This is similar to the XPAR subroutine, with this exception: in addition to the NITP protocol, Host Adapter Command Code 01, and TIWAY Primitive 01 are added to the command and stripped from the response. See the documentation for the individual controllers for information concerning the Task Codes they support.

---

**NOTE:** NATIVE is recommended for use only by the experienced programmer.

---

**RDSTS**

The RDSTS subroutine reads the status of the specified Secondary and attached device. RDSTS is the implementation of Primitive 02.

**TIGET**

The TIGET subroutine is used to read consecutive memory locations from a specified NIM-based attached device and to convert the data from the 500/505 format to a format compatible with the host. TIGET is an implementation of Primitive 20.

**TIPUT**

The TIPUT subroutine is used to convert data from the host format to the 500/505 format and to write to consecutive memory locations in a specified NIM-based attached device. TIPUT is an implementation of Primitive 30.

## TIWAY Primitive Subroutines (continued)

---

TIREAD	The TIREAD subroutine reads consecutive memory locations from a specified NIM-based attached device. Data is returned in the NATIVE 500/505 format as an array of bytes. TIREAD is an implementation of Primitive 20.
TIWRIT	The TIWRIT subroutine writes to consecutive memory locations in a specified NIM-based attached device. The data is passed in the NATIVE 500/505 format as an array of bytes. TIWRIT is an implementation of Primitive 30.
WRBUF	The WRBUF subroutine is used to write to consecutive memory locations in NIM-based attached devices that support buffered program memory. It stores data temporarily in the NIM and downloads it from the NIM to the Unilink upon request. WRBUF is the implementation of 5TI Primitive 33.

---

**NOTE:** The 5TI makes use of this facility for program downloads. See the *5TI NIM User Manual* for a detailed description of this facility.

---

 <b>WARNING</b>
---

<p>Before transferring a new program by calling WRBUF, clear L-memory completely.</p>
---

<p>If this is not done, parts of residue programs may be executed. This could cause unexpected controller behavior that could result in death or serious injury, and/or equipment damage.</p>
---

<p>Use FILL with pattern = 0000 to entirely clear L-memory.</p>
---

WRTGAT	The WRTGAT subroutine combines the functions of the TIWRIT and GATHER subroutines. It writes consecutive memory locations in a specific NIM-based attached device and then gathers data. The data are sent and returned in the 500/505 format. WRTGAT is the implementation of Primitive 52.
--------	--

## 4.6 CIM Functional Command Subroutines

Each CIM Functional Command Subroutine corresponds to a CIM Functional Command. The intent is to provide the applications programmer with a natural interface to the Functional Command. The arguments correspond either to information required to build the request buffer or to data returned in the response. These subroutines can only be used on PM550 controllers using the CIM network interface. Table 4-3 shows the CIM Functional Command subroutines. All CIM Functional Command subroutines use Host Adapter Command Code 01. Before attempting to use these subroutines, become familiar with the Functional Commands used. Refer to the *CIM User Guide* for more information.

The CIM Functional Command subroutines are only supported when the Unilink Host Adapter is in the EHA mode.

**NOTE:** All CIM functional command subroutines use host adapter command code 01. Before attempting to use these subroutines, you should be familiar with the functional commands you intend to use. Refer to the *CIM User Guide* for more information.

The INIT subroutine must be called before any other subroutine can be executed.

Table 4-3 CIM Functional Command Code Subroutines

Subroutine	CIM Functional Command (hex)	Description
CCUSTS	69	Reads the attached PM550 status.
CIMDNL	65	Downloads instructions or data into a PM550.
CIMRD	62 (image register) 60 (V-, C-, A-memory)	Reads consecutive memory locations or V-, C-, A-memory locations from the specified CIM-based PM550.
CIMPUL	66	Uploads instructions or data to the host computer from a PM550.
CIMWR	63 (image register) 61 (V-, C-, A-memory)	Writes to consecutive memory locations for V-, C-, A-memory locations in the specified attached device.
RDLOOP	6A	Reads loop parameters from a PM550.
RNDRD1	6D	Defines a data acquisition block in a PM550.
RNDRD2	(subcommand 00) RNDRD2 (subcommand 01)	6D gathers a block or group of blocks previously defined using RNDRD1 or RNDRD3.
RNDRD3	6D (subcommand 02)	Gathers data from a block of data acquisition blocks specified by a mask and defines new data acquisition blocks.
RNDRD4	6D (subcommand 03, V-, C-, A-memory) (subcommand 04, image registers)	Writes to a set of sequential memory locations and gathers data from one or more data acquisition blocks specified by a mask.



## CIM Functional Command Subroutines (continued)

---

**CCUSTS** The CCUSTS subroutine reads the attached PM550 Command Control Unit's (CCU) status. CCUSTS is the implementation of CIM functional command 69.

**CIMDNL** The CIMDNL subroutine downloads instructions or data into a PM550. Refer to the *CIM User Manual* for more information on the procedures necessary to do this. CIMDNL is the implementation of CIM functional command 65.

---

**NOTE:** The PM550 must be in startup or remote mode in order for the download operation to be enabled. If the PM550 is in any other mode, a facility 8 message 67 (hex) Invalid Download Sequence error is returned.

---

**CIMRD** The CIMRD subroutine reads consecutive memory locations from the specified CIM-based attached device. CIMRD is the implementation of CIM functional commands 60 and 62.

**CIMUPL** The CIMUPL subroutine uploads one or more memory areas from a PM550. You should refer to the *CIM User Manual* for more information on the procedures necessary to do this. CIMUPL is the implementation of CIM functional command 66.

**CIMWR** The CIMWR subroutine writes to consecutive memory locations in the specified CIM-based attached device. CIMWR is the implementation of CIM functional commands 61 and 63.

**RDLOOP** The RDLOOP subroutine reads loop parameters from a PM550. RDLOOP is the implementation of CIM functional command 6A.

**RNDRD1** The RNDRD1 subroutine defines a data acquisition block in a PM550. The data acquisition block can then be activated to gather the specified data using an RNDRD2, RNDRD3, or RNDRD4 subroutine. RNDRD1 is the implementation of CIM functional command 6D00.

---

**NOTE:** Unlike the NIM DEFBLK subroutine, any new definition starts with the first block, and all blocks defined after the first are numbered sequentially from that number.

---

---

**RNDRD2**                    The RNDRD2 subroutine gathers a block or group of blocks previously defined using either an RNDRD1 or an RNDRD3 subroutine. The buffer returned contains only the data specified by the blocks, and those data are in 500/505 format. RNDRD2 is the implementation of CIM functional command 6D01.

**RNDRD3**                    The RNDRD3 subroutine gathers data from assigned data acquisition blocks specified by a mask and defines new data acquisition blocks. RNDRD3 is the implementation of CIM functional command 6D02.

---

**NOTE:** Unlike the NIM DEFBLK subroutine, any new definition starts with the first block, and all blocks defined after the first are numbered sequentially from that number.

---

**RNDRD4**                    The RNDRD4 subroutine performs two functions in the same call and same Host Adapter transaction. It writes to a set of sequential memory locations (see the CIMWR subroutine, for more detail) and also gathers data from one or more data acquisition blocks specified by a mask. RNDRD4 is the implementation of CIM functional commands 6D03 and 6D04.

Chapter 5

# TIWAY Support Routines

---

- 5.1 **Overview** ..... 5-2
- 5.2 **Descriptions of Support Subroutines** ..... 5-4
  - BLDMSK Subroutine ..... 5-4
  - GETMSG Subroutine ..... 5-4
  - HST2TI Subroutine ..... 5-4
  - LKUFMT Subroutine ..... 5-4
  - LKUTGL Subroutine ..... 5-4
  - LKUTGS Subroutine ..... 5-4
  - PUTMSG Subroutine ..... 5-4
  - T12HST Subroutine ..... 5-4
  - TIXTN and TIXTNW Subroutines ..... 5-5

The TIWAY Support Subroutines do not correspond directly to any single TIWAY network operation. Instead, these are general purpose support subroutines. These subroutines do not require communication with the Host Adapter. The support subroutines include:

- A tag conversion subroutine for returning a list of address parameters for a tag specification. (LKUTGL, LKUTGS, LKUFMT)
- Subroutines for converting formats. (TI2HST, HST2TI)
- A subroutine for building a mask. (BLDMSK)
- Subroutines for interpreting error messages. (GETMSG, PUTMSG)
- Subroutines for calling the second half of a subroutine when using asynchronous completion. (TIXTN, TIXTNW)

Although the TIWAY Support Subroutines use many of the same arguments that are used in the TIWAY Subroutines, arguments unique to the support subroutines also occur. Therefore, the arguments used by each subroutine are defined with that subroutine, and the declaration is included in full detail in the Reference Section (Chapter 10). Table 5-1 shows the support subroutines.

---

**NOTE:** You must call the INIT subroutine before any of these subroutines can be executed.

---

Table 5-1 Reference List of Support Subroutines

<b>Subroutine</b>	<b>Arguments</b>	<b>Descriptions</b>
BLDMSK	mask,list	Builds a mask from a list of block numbers.
GETMSG*	istat,rsplen,str80	Returns the text of an error message.
HST2TI	istat,xtn,tt,nnnn, hstbuf, tibuf	Performs conversions from host format to 500/505 format.
LKUFMT	istat,xtn,tt,fmt,tilen, hostlen,cimtyp	Looks up a data element type; returns format information.
LKUTGL	istat, xtn, tag, hwy, secadd, tt, aaaa	Looks up a tag name; returns an address parameter list (long format).
LKUTGS	istat,xtn,tag,,addlst	Looks up a tag name; returns an address parameter list (short format).
PUTMSG*	istat	Displays an error message on a your terminal.
TI2HST	istat,xtn,tt,nnnn,tibuf, hstbuf	Performs conversions from 500/505 format to host format.
TIXTN	istat,xtn	Calls the second half of a subroutine that does I/O and returns an error message if I/O is not complete.
TIXTNW	istat,xtn	Calls the second half of a subroutine that does I/O and waits to respond when I/O is complete

\* These subroutines are not part of the main TIWAY subroutine library. In order to reduce the size of applications programs, they are located in a separate library module (TIMSG) which may be linked if needed.

## 5.2 Descriptions of Support Subroutines

---

BLDMSK Subroutine	The BLDMSK subroutine builds a mask from a list of block numbers.
GETMSG Subroutine	The GETMSG subroutine returns an error message to the caller in an 128-character string. GETMSG is called with the composite status (returned in the <code>istat</code> field) obtained from a previous call to a TIWAY Subroutine.
HST2TI Subroutine	The HST2TI subroutine converts data from host format to 500/505 format. The subroutine can convert from one buffer into another, or the same buffer can be specified for both source and destination. The HST2TI subroutine is the complement of the TI2HST subroutine.
LKUFMT Subroutine	The LKUFMT subroutine returns information on data element format.
LKUTGL Subroutine	The LKUTGL subroutine returns address information that describes the address specification passed to it. The address specification can be any of the available types (tag name, ASCII, or binary). LKUTGL looks up a tag name and determines the address information from the address specifier. (Long format).
LKUTGS Subroutine	The LKUTGS subroutine returns address information that describes the address specification passed to it. The address specification can be any of the available types (tag name, ASCII, or binary). LKUTGS looks up a tag name or derives the address information from the address specifier. (Short format).
PUTMSG Subroutine	The PUTMSG subroutine takes one argument, the composite status as returned from a previous call to a TIWAY subroutine. It prints a message containing the facility code and message code and a text (if one exists for that status) on the operator's console.
TI2HST Subroutine	The TI2HST subroutine converts a list of data of a single type from 500/505 format to host format. (The only subroutine that converts a list of returned data is the TIGET subroutine; the CIMRD, GATHER, NATIVE, RNDRD2, RNDRD3, RNDRD4, TIREAD, WRGAT, and XPAR subroutines return data in the 500/505 format.) TI2HST can convert from one buffer into another or the same buffer can be specified for source and destination. The TI2HST subroutine is the complement of the HST2TI subroutine.

Note that the GATHER, WRGAT, and RNDRDx subroutines return buffers that may have to be converted in several segments.

---

**TIXTN and TIXTNW  
Subroutines**

Each subroutine that does I/O is divided into two parts: the first half uses the argument list to build the command buffer and to issue the I/O request; the second half takes the results of the I/O (i.e., the response buffer) and returns pertinent parts of it to the application using the argument list.

The TIXTN and TIXTNW subroutines are used to call the second half of a subroutine that does I/O and are not valid for any others.

If a subroutine is called with a transaction number of 0, it is synchronous: that is, after the first half, the routine waits until the I/O is complete and then executes the second half. In this case (the fully synchronous case), all processing is performed with a single call either to the TIWAY entry point or to the individual subroutine. This is the normal case and is supported in all operating systems. The transaction number is a read-only variable and can be specified with the constant zero in the argument list.

If a subroutine is called with a transaction number of -1, only the first half of the subroutine is executed before control returns to the programmer. The transaction number is overwritten with a new number identifying that transaction, and a copy of the first half argument list is kept by the TIWAY library. To receive the results from the I/O (the second half), the TIXTN or TIXTNW subroutine must be called. Either of these subroutines is called with only an *istat* and the transaction number that was returned from the first half. The results from the I/O in the second half are returned through the copy of the first half argument list; that is, no arguments to return data are required by the TIXTN or TIXTNW entry points.

The only difference between the TIXTN and TIXTNW subroutines occurs when the I/O is not finished when they are called. The TIXTN subroutine returns immediately with a status of **FIRST HALF NOT FINISHED YET** (facility 1, message 10 [hex]) and you should call it back later. The TIXTNW subroutine waits for the I/O to complete before returning (i.e., never returns with the **FIRST HALF NOT FINISHED YET** error).

# Chapter 6

## File Transfer Subroutines

---

<b>6.1</b>	<b>Overview</b> .....	<b>6-2</b>
	Partial Memory Transfers .....	6-2
	Transfers Between Different Types of Devices .....	6-4
	Generic Upload/Download .....	6-4
	Specific Segment Transfers .....	6-5
<b>6.2</b>	<b>Subroutine Descriptions</b> .....	<b>6-6</b>
	<b>UPLOAD</b> .....	6-6
	<b>DNLOAD</b> .....	6-6
	Upload .....	6-6
	Download .....	6-7
<b>6.3</b>	<b>Transfer File Descriptions</b> .....	<b>6-8</b>
	Transfer File Name Specification .....	6-8
	Transfer File Construction .....	6-8
	Transfer File Formats .....	6-8



Two callable procedures, UPLOAD and DNLOAD, are provided to cause transfers of Secondary memory stored within ASCII files.

- Both procedures are callable from the applications program only.
- Both procedures are separate from the rest of the TIWAY Host Software Package.
- Each procedure resides in a separate linkable object module.

Upload and download (file transfer) functionality supported includes the following.

- Partial memory transfers (UPLOAD and DNLOAD).
- Transfers [downloads] between different types of devices (DNLOAD only).
- Generic upload/download (UPLOAD and DNLOAD).
- Specific segment upload/download (UPLOAD and DNLOAD).

**Partial Memory Transfers**

Partial memory transfers permit upload or download of a selected memory type using a transfer code which is the same as the VPU function code for a similar transfer on that device. Table 6-1 shows those codes.

Table 6-1 Partial Memory Transfer Codes

<b>Upload</b>	<b>Download</b>
60 : all memory	90 : all memory except WXY, IR, WF
61 : L-memory	91 : L-memory
62 : V-memory	92 : V-memory
63 : Constant memory	93 : Constant memory
64 : S-memory – loop tables	94 : S-memory – loop tables
65 : S-memory – analog alarms	95 : S-memory – analog alarms
66 : S-memory – SF programs	96 : S-memory – SF programs
67 : S-memory – SF subroutines	97 : S-memory – SF subroutines
	98 : IR, WF memory
	99 : WXY memory

**Selection** Partial memory transfers are performed one memory type at a time; each transfer is placed in a separate file. Table 6-2 and Table 6-3 summarize allowable partial transfers for each device.

Table 6-2 Memory Selectable for Upload

Type	520	530	520C	525/530C	560/565	5TI	550	MRCU
MEM	Full	Full	Full	Full	Full	Full	Full	Full
IOC	N/A	N/A	Full	Full	Full	N/A	N/A	N/A
L	Full Partial	Full Partial	Full Partial	Full Partial	Full Partial	Full	Full Partial	Full Partial
V	Full Partial	Full Partial	Full Partial	Full Partial	Full Partial	N/A	Full Partial	Full Partial
C/K	N/A	N/A	N/A	N/A	Full Partial	N/A	Full Partial	Full Partial
S	N/A	N/A	N/A	N/A	Full	N/A	N/A	N/A
ST	Full	Full	Full	Full	Full	N/A	N/A	N/A
IR	Full	Full	Full	Full	Full	N/A	N/A	N/A
WXY	Full	Full	Full	Full	Full	N/A	N/A	N/A
WF	Full	Full	Full	Full	Full	N/A	N/A	N/A

Table 6-3 Memory Selectable for Download

Type	520	530	520C	525/530C	560/565	5TI	550	MRCU
MEM	Full	Full	Full	Full	Full	Full	Full	Full
IOC	N/A	N/A	Full	Full	Full	N/A	N/A	N/A
L	Full Partial	Full Partial	Full Partial	Full Partial	Full Partial	Full	Full Partial	Full Partial
V	Full Partial	Full Partial	Full Partial	Full Partial	Full Partial	N/A	Full Partial	Full Partial
C/K	N/A	N/A	N/A	N/A	Full Partial	N/A	Full Partial	Full Partial
S	N/A	N/A	N/A	N/A	Full	N/A	N/A	N/A
ST	Full	Full	Full	Full	Full	N/A	N/A	N/A
IR	Partial	Partial	Partial	Partial	Partial	N/A	N/A	N/A
WXY	Partial	Partial	Partial	Partial	Partial	N/A	N/A	N/A
WF	Partial	Partial	Partial	Partial	Partial	N/A	N/A	N/A

## Overview (continued)

---

### Transfers Between Different Types of Devices

Memory configurations of certain devices within the Series 500/505 controllers can fit into other devices within the same series. This occurs when the memory configuration to be transferred is of the same size or smaller than the memory configuration of the target device. When a request is made to transfer between unlike device types, the file-transfer logic performs a check to determine:

- If both devices are within the Series 500/505,
- And, if so, if transfers between the devices are permitted. Table 6-4 summarizes allowable transfers.

Table 6-4 Allowable Cross-Device Memory Transfers

		Source				
		520	530	520C	525/530C	560/565
<b>Target</b>	520	Yes	No	No	No	No
	520C	Yes	Yes*	Yes	No	No
	530	Yes	Yes	Yes	No	No
	525/530C	Yes	Yes	Yes	Yes	No
	560/565	Yes	Yes	Yes	Yes	Yes
	* Permitted if memory sizes allow transfer					

### Generic Upload/Download

A special feature of the 560 and 565 sequencers and the Unilink Secondary adapter is generic upload/download, which is supported by the TIWAY Host Software Package. Memory types are grouped together by class and may be transferred totally or partially by class. The following table lists memory types which are supported by generic upload/download.

Table 6-5 Memory Selectable for Generic Upload/Download

Memory Class	Memory Type	Upload/Download Supported
Program Memory	MEM	Full, partial
	IOC	
	L	
	K	
	S	
Data Memory	V	Full, partial
	WF	
	IR	

**Selection** Generic memory transfers are selected by requesting transfers of all memory types, program memory, or data memory. Each upload request is placed in a separate file.

Specific Segment Transfers

Another special feature of the Unilink Secondary adapter is the segmentation of Secondary device memory. The UPLOAD and DNLOAD routines support transfers of either all segments or of specific segments. It is your responsibility to know how the segments are defined. You must also know the Secondary device memory program name. Consult the *SIMATIC Unilink Secondary Adapter User Manual* for further information on the use of specific segments.

**Selection** The transfer of specific segments to and from a Unilink Secondary adapter is selected by setting appropriate bits within a selection mask. Either all or specific segments may be selected for transfer. Each upload request is placed in a separate file. It is your responsibility to know which bits within the mask correspond to the desired segment. (These bits are specific to an application.)

## 6.2 Subroutine Descriptions

---

UPLOAD	The UPLOAD subroutine uploads the contents of a Secondary to a text file on the host system. This procedure is intended as a backup function for Secondary programs.
DNLOAD	The DNLOAD subroutine is used to download a previously uploaded file to a Secondary.
Upload	<p>When using the upload function, keep the following guidelines in mind.</p> <ul style="list-style-type: none"><li>• Uploads are made from many Secondaries, regardless of state or mode. However, because the upload function utilizes native task codes, Series 500/505 NIMs must be in the remote mode prior to the upload request in order for the upload to be performed properly.</li><li>• The UPLOAD routine terminates on any network error except 8003, which means a particular memory type does not exist in the Secondary. For example, if an upload is requested and no K-memory is programmed, the network returns an error to the UPLOAD routine signifying such; however, the routine ignores this particular error and continue uploading.</li><li>• The UPLOAD routine is designed to support future products, but that support is limited to generic uploads. That is, UPLOAD permits an upload attempt from an unrecognized device type. However, only a generic upload is attempted. If the device does not support generic uploads, then an error condition occurs.</li><li>• If a specific segment upload has been requested through the TIUSER UPLOAD command, an attempt is made to upload all requested segments. If an error occurs while uploading a segment, an error message appears and an attempt is made to upload the next segment.</li><li>• If a specific segment upload has been requested through the UPLOAD subroutine, an attempt is made to upload all requested segments until successfully completed or an error occurs. If an error occurs while uploading a segment, the upload is terminated and the appropriate <code>istat</code> is returned to you.</li><li>• If a specific segment upload has been requested from a device that does not support program names, the program name entered in the TIUSER UPLOAD command or passed to the UPLOAD subroutine should be a string containing one or more blanks.</li></ul>

---

## Download

When using the download function, keep the following guidelines in mind.

- All Secondary memory is cleared prior to a download.
- Downloads of previously uploaded transfer files are permitted only to the same device or between like devices; i.e., 520 to 520, 565 to 565, etc.
- The target device (the Secondary to receive the download) must be in the program mode before a DOWNLOAD is allowed. Series 500/505 NIMs must be in the remote mode as well.
- A memory check is made prior to download to insure that the configuration contained within the transfer file fits into the target device. If the transfer file contains 2 kbytes of logic memory, the target device must be configured for at least 2 kbytes of logic memory as well.
- Downloads to the 560/565 cause the memory configuration table to be rewritten. The NIM, however, only updates this table upon powerup or reset. In order to achieve this table rewrite, the Secondary is disconnected (reset) and then reconnected after the table has been rewritten.
- The DNLOAD routine terminates upon any network error.
- The DNLOAD routine is also designed to support future products, but that support is limited to generic downloads. That is, DNLOAD permits a download attempt to an unrecognized device type. However, only a generic download is attempted. If the device does not support generic downloads, then an error condition occurs.

---

**NOTE:** If a download involving an unrecognized device type is attempted, then only a download from an unrecognized device type to a like type is permitted. Any combination of unrecognized device type and recognized device type (or a different unrecognized type) is not permitted. For example, type X to type X is allowed. Type X to 565, 565 to type X, and type X to type Y are not allowed.

---

---

**NOTE:** You must change from RUN mode to PROG mode before you attempt a download. Use the CHNGST function option 02 to change the state of the controller from RUN to PROG. Use CHNGST option 00 to return to RUN after you have performed the download.

---

## 6.3 Transfer File Descriptions

---

**Transfer File Name Specification** Any valid VMS file specification may be used in identifying a transfer file.

**Transfer File Construction** Transfer files are constructed as sequential text files with a maximum record length of 128 characters.

**Transfer File Formats** There are two types of transfer files: a conventional transfer file and a generic transfer file.

**Conventional Transfer File Format** Conventional transfer files are produced by requesting either full or partial uploads from a Secondary.

**Generic Transfer File Format** The generic transfer file, created when a generic upload is requested, has a file format very similar to the conventional transfer file. Some differences do exist, however.

- The memory types are repositioned to adhere to the order in which the Series 500/505 NIM expects to find data for the generic download operation.

Memory types are organized for program and data memory.

- Records within each data type are defined to include additional data to promote faster downloads.

Task codes and associated parameters are included in addition to data.

Figure 6-1 through Figure 6-18 show the formats and examples of conventional and generic transfer files for each of the various Secondaries supported by this package.

---

**Specific Segment Transfer File Format** The specific segment transfer file, created when a specific segment upload from a Unilink Secondary Adapter is requested, has a format similar to the generic transfer file format for the Unilink. The following differences should be noted.

- The Secondary device memory program name has its own record.

---

**NOTE:** Some devices do not support a program name. In this event, the record exists but the name is blank.

---

- Two header records exist for each segment:
  - A record containing the segment selection mask.
  - A record containing the segment name.
- Records containing data returned for each requested segment is organized sequentially after the two segment header records.
- As with the generic transfer file, data records for each segment contain additional data to promote faster downloads.



## Transfer File Descriptions (continued)

---

```
TAG NAME: #hhaa
HWY/STA : hh aa
TIME      : dd-mmm-yyyy hh:mm:ss.ss
MODEL     : 00
CONFIG    : llll 0000 0000 iiii

L 0001: nnnn nnnn nnnn nnnn nnnn nnnn nnnn nnnn
L 0009: nnnn nnnn nnnn nnnn nnnn nnnn nnnn nnnn
      .
      .
L llll: nnnn nnnn nnnn nnnn nnnn nnnn nnnn nnnn
```

Figure 6-1 5TI Upload File Format

```
TAG NAME: #0101
HWY/STA : 01 01
TIME:    18-JUN-1995 09:29:27.97
MODEL    : 00
CONFIG   : 0400 0000 0000 0400

L 0001: E5B4 8401 B9A0 89A9 EDB3 9000 9801 8401
L 0009: BDA0 8A01 9802 8401 B5A0 89A9 9400 8A03
      .
      .
L 03E9: 8000 8000 8000 8000 8000 8000 8000 8000
L 03F1: 8000 8000 8000 8000 8000 8000 8000 8000
L 03F9: 8000 8000 8000 8000 8000 8000 8000 8000
```

Figure 6-2 5TI Upload File Example

```

TAG NAME: #hhaa
HWY/STA : hh aa
TIME    : dd-mmm-yyyy hh:mm:ss.ss
MODEL   : 20[30]
CONFIG  : llll vvvv 0000 iiii

L 0001:  mmmm mmmm mmmm mmmm mmmm mmmm mmmm mmmm
L 0009:  mmmm mmmm mmmm mmmm mmmm mmmm mmmm mmmm
.
L 1111:  mmmm mmmm mmmm mmmm mmmm mmmm mmmm mmmm
V 0001:  nnnn nnnn nnnn nnnn nnnn nnnn nnnn nnnn
V 0009:  nnnn nnnn nnnn nnnn nnnn nnnn nnnn nnnn
.
V vvvv:  nnnn nnnn nnnn nnnn nnnn nnnn nnnn nnnn
WXY 0001: nnnn nnnn nnnn nnnn nnnn nnnn nnnn nnnn
.
WXY iiii: nnnn nnnn nnnn nnnn nnnn nnnn nnnn nnnn
ST nnnn: nn
IR aaaa: dddd 0j
IR aaaa: dddd dddd 0j
           where   aaaa = start address
                   dddd = returned address
                   dddd dddd = returned address
                   0j = status value

WF aaaa: rrrr
WF aaaa: aaaa rrrr
           where   aaaa = address 1-1024
                   aaaa aaaa =address 1025-8192
                   rrrr = word force status value

```

Figure 6-3 520/530 Upload File Format

## Transfer File Descriptions (continued)

---

```
TAG NAME: #0753
HWY/STA:  07 53
TIME:     18-JUN-1995 09:29:27.97
MODEL:    30
CONFIG:   1FFF 0800 0000 03FF
L 0001:  E5B4 8401 B9A0 89A9 EDB3 9000 9801 8401
L 0009:  BDA0 8A01 9802 8401 B5A0 89A9 9400 8A03
.
L 1FF1:  8000 8000 8000 8000 8000 8000 8000 8000
L 1FF9:  8000 8000 8000 8000 8000 8000 8000 8000
V 0001:  AAAA BBBB CCCC DDDD 0010 3890 2904 0E70
V 0009:  2CEC 4C2D 309F 3D38 3A12 46AB 483E FC18
.
V 07F1:  0000 0000 0000 0000 0000 0000 0000 0000
V 07F9:  0000 0000 0000 0000 0000 0000 0000 0000
ST 0000: 96
```

Figure 6-4 530 Upload File Example

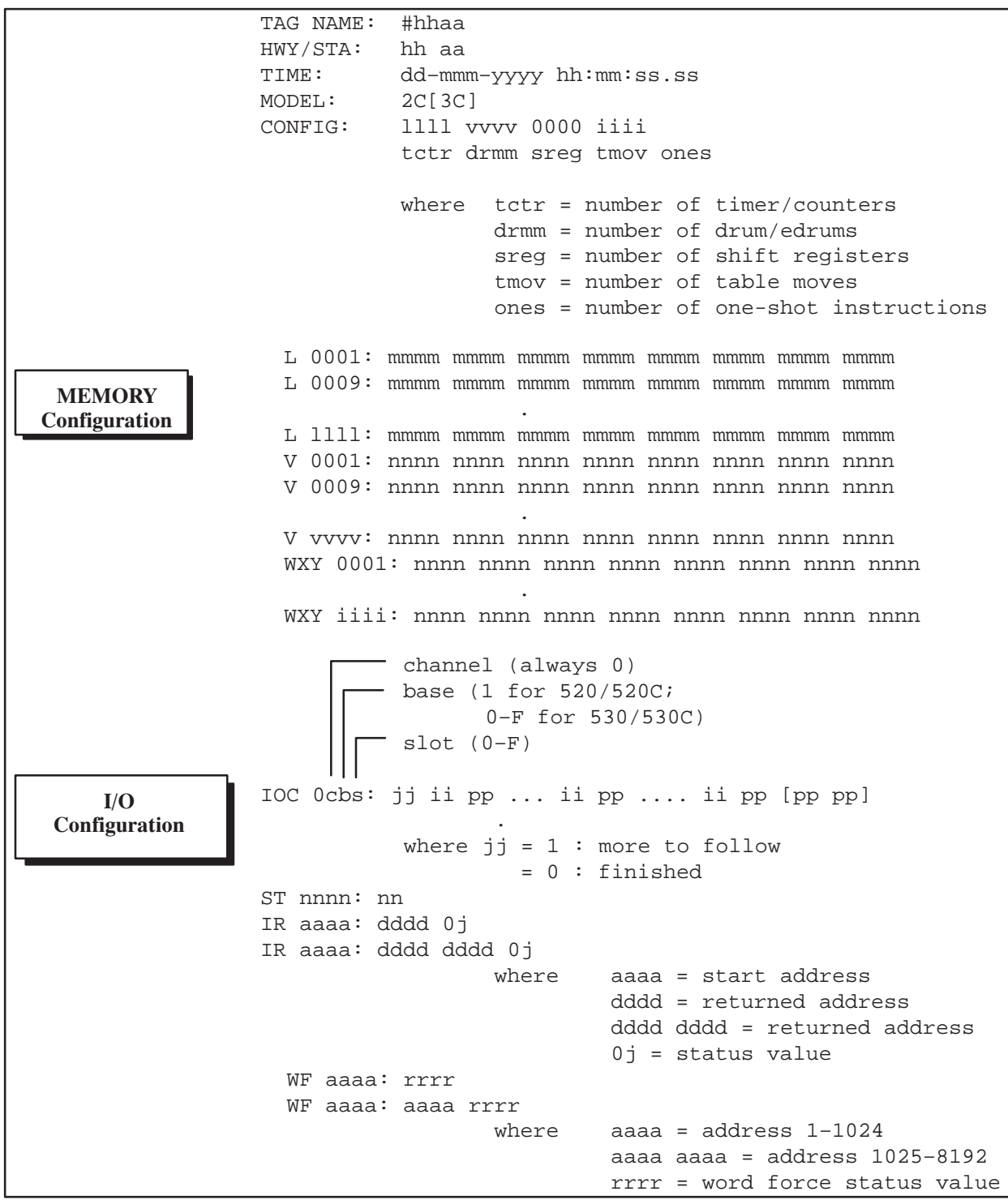


Figure 6-5 520C/530C Upload File Format

## Transfer File Descriptions (continued)

---

```
TAG NAME: /072c
HWY/STA: 07 2C
TIME:    18-JUN-1995 09:28:43.77
MODEL:   2C
CONFIG:  0800 0400 0000 0080
          0100 001F 001E 001E 0080
L 0001:  E5B4 8401 B9A0 89A9 EDB3 9000 9801 8401
L 0009:  BDA0 8A01 9802 8401 B5A0 89A9 9400 8A03
L 0011:  EBD1 EBD6 99A0 8402 8402 A001 0064 9803

          .
L 07E9:  8000 8000 8000 8000 8000 8000 8000 8000
L 07F1:  8000 8000 8000 8000 8000 8000 8000 8000
L 07F9:  8000 8000 8000 8000 8000 8000 8000 8000
V 0001:  0011 0022 0003 0004 0550 0606 0007 0008
V 0009:  0009 4C2D 000B 000C 000D 000E 000F 0010
V 0011:  0011 0012 0013 501D 0015 0016 0017 0018

          .
V 03E9:  0001 0002 0003 0004 0005 0006 0007 0008
V 03F1:  0009 000A 0000 0000 0000 0000 0002 0000
V 03F9:  0000 0000 0000 0000 0000 0000 0000 0000
IOC 0000: 00 00 60 01 60 02 07 03 60 04 07 05 60 06 60 07 60
IOC 0008: 00 08 60 09 60 0A 60 0B 67 0C 67 0D 67 0E 67 0F 67
IOC 0010: 00 10 FF 11 FF 12 FF 13 FF 14 FF 15 FF 16 FF 17 FF
IOC 0018: 00 18 FF 19 FF 1A FF 1B FF 1C FF 1D FF 1E FF 1F FF
ST 0000: 00
```

Figure 6-6 520C Upload File Example

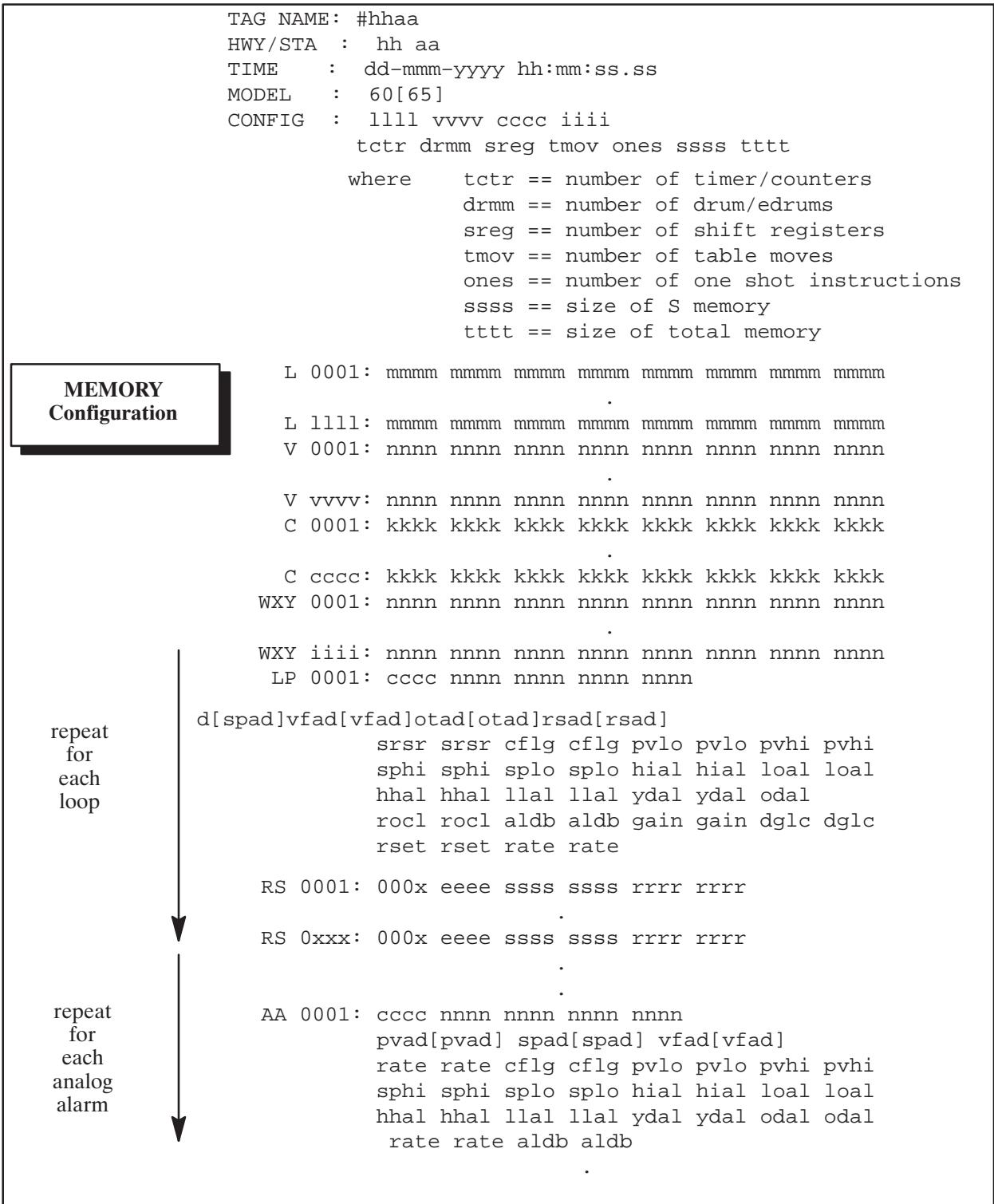


Figure 6-7 560/565 Non-extended Upload File Format

## Transfer File Descriptions (continued)

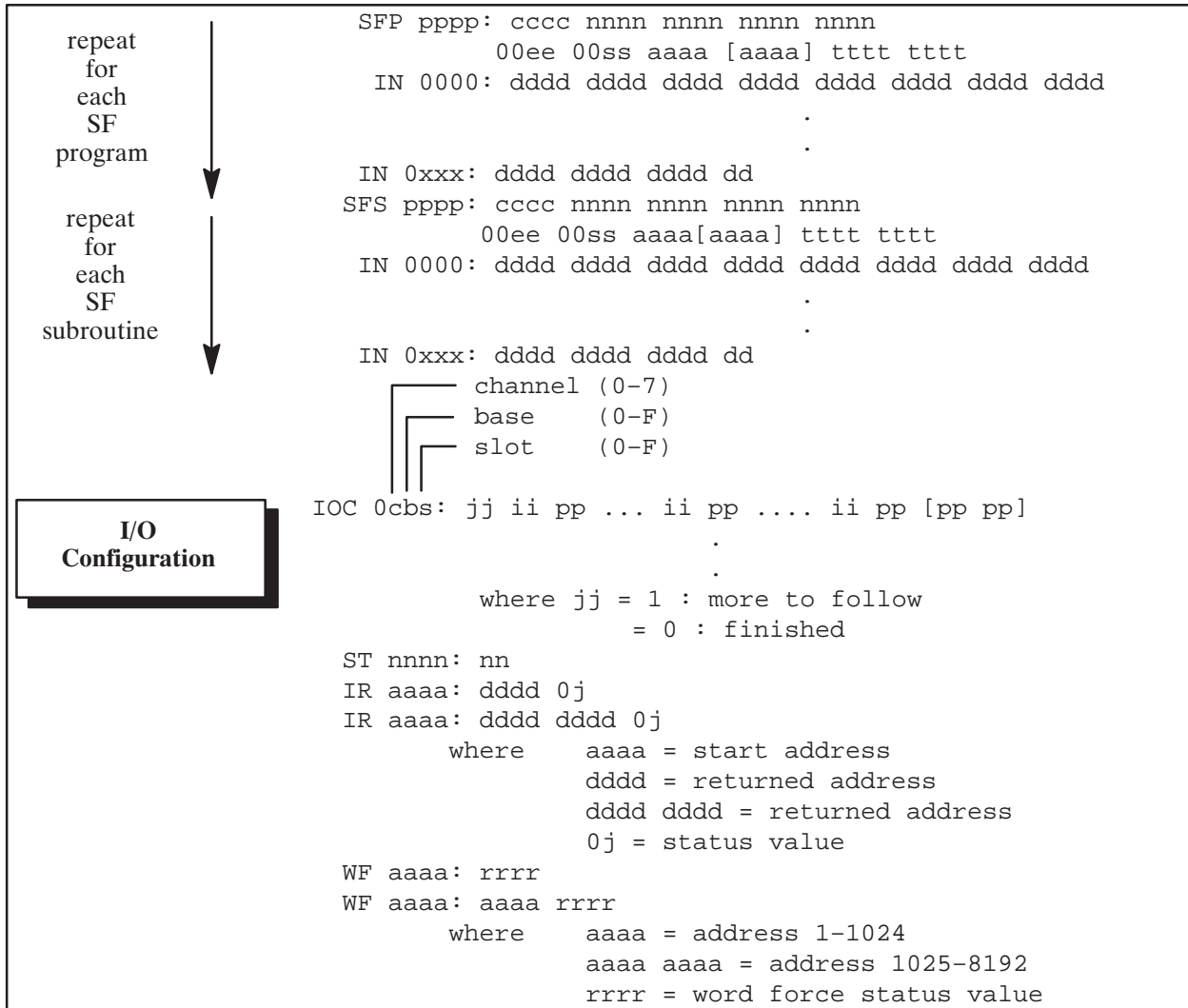


Figure 6-7 560/565 Non-extended Upload File Format (continued)

```

TAG NAME: #0722
HWY/STA: 07 22
TIME: 22-JUL-1995 10:54:53.07
MODEL: 65
CONFIG: 3000 1800 0400 02C8
        0400 0083 0800 0400 0800 2800 8000
L 0001: 8206 048C 8206 64AA 8205 64A0 B101 0001
L 0009: 0064 0064 00C8 014D 01BC 7001 002B 7001
L 0011: 009A 7001 0109 7001 0178 7001 01E7 7002
        .
        .
        .
L 2FE9: 8000 8000 8000 8000 8000 8000 8000 8000
L 2FF1: 8000 8000 8000 8000 8000 8000 8000 8000
L 2FF9: 8000 8000 8000 8000 8000 8000 8000 8000
V 0001: 335A 0000 0100 0000 0000 0000 0600 0001
V 0009: 0700 0102 0304 0506 0708 0000 0000 0000
V 0011: 0000 0000 0159 0003 002D 01B3 0038 0000
        .
        .
        .
V 17E9: 0238 0038 0007 0004 0000 0000 0038 0000
V 17F1: 01C9 0000 0038 0004 0038 0000 002D 0006
V 17F9: 01C9 0237 002D 0017 8000 7FFF FFFF 270F
C 0001: 19AD B0CF 0159 0159 0159 11B4 0216 0285
C 0009: 02FD 223D 036C 0285 0159 0D7B 11D2 11AE
C 0011: 223C 223C 1DE5 11AE 1490 1DD3 0312 1934
        .
        .
        .
C 03E9: 0004 0000 3C23 D70A 378B CF65 0005 0000
C 03F1: 3C23 D70A 378B CF65 E006 0000 0000 0258
C 03F9: 3C23 D70A E007 0000 0000 0258 3C23 D70A
LP 0001: 4001 3135 2052 2D53 0000
        6064 7000 0064 7064 0065
        4000 0000 3E23 6064 0000 0000 4049 0FF8
        4049 0FF8 0000 0000 4049 0FF8 4000 0057
        4049 0FF8 3F80 0057 4049 0FF8 4049 0FF8
        F80 0000 3F80 0057 2141 1BC7 3F80 0000
        452B CEFF 3C23 D70A
RS 0000: 0000 0001 3F80 0000 3F80 0000
RS 0001: 0000 0002 3F80 0000 3F80 0000
RS 0002: 0000 0003 3F80 0000 3F80 0000
RS 0003: 0000 0004 3F80 0000 3F80 0000
RS 0004: 0000 0005 3F80 0000 3F80 0000
RS 0005: 0001 6006 3F80 0000 3F80 0000
RS 0006: 0001 6007 3F80 0000 3F80 0000

```

Figure 6-8 565 Non-extended Upload File Example



## Transfer File Descriptions (continued)

```

RS 0007: 0001 6008 3F80 0000 3F80 0000
RS 0008: 0001 6009 3F80 0000 3F80 0000
RS 0009: 0001 600A 3F80 0000 3F80 0000
RS 000A: 0000 000B 3F80 0000 3F80 0000
RS 000B: 0000 000C 3F80 0000 3F80 0000
RS 000C: 0000 000D 3F80 0000 3F80 0000
RS 000D: 0000 000E 3F80 0000 3F80 0000
RS 000E: 0000 000F 3F80 0000 3F80 0000
.
.
LP 0040: 4040 3135 2052 2D53 0000
        6064 7000 0064 7064 0065
        400 0000 3E23 6064 0000 0000 42C8 0000
        42C8 0000 0000 0000 42C4 0000 4000 0000
        42C6 0000 3F80 0000 40A0 0000 4120 0000
        F80 0000 3F80 0000 3F80 0000 3F80 0000
        4479 C000 3C23 D70A
RS 0000: 0000 0001 3F80 0000 3F80 0000
RS 0001: 0000 0002 3F80 0000 3F80 0000
RS 0002: 0000 0003 3F80 0000 3F80 0000
RS 0003: 0000 0004 3F80 0000 3F80 0000
RS 0004: 0000 0005 3F80 0000 3F80 0000
RS 0005: 0001 6006 3F80 0000 3F80 0000
RS 0006: 0001 6007 3F80 0000 3F80 0000
RS 0007: 0001 6008 3F80 0000 3F80 0000
RS 0008: 0001 6009 3F80 0000 3F80 0000
RS 0009: 0001 600A 3F80 0000 3F80 0000
RS 000A: 0000 000B 3F80 0000 3F80 0000
RS 000B: 0000 000C 3F80 0000 3F80 0000
RS 000C: 0000 000D 3F80 0000 3F80 0000
RS 000D: 0000 000E 3F80 0000 3F80 0000
RS 000E: 0000 000F 3F80 0000 3F80 0000
AA 0001: 0401 4F52 2020 4E4F 2020
        60C8 6000 00C8
        4000 0000 7E00 08C8 3F80 0000 44FA 0000
        44BB 8000 4348 0000 44C8 0067 4348 0684
        44E0 FF34 42C8 06A4 0000 0000 0000 0000
        0000 0000 3FFF DF3C
.
.
AA 0080: 0480 4F52 2020 4E4F 2020
        60C8 6000 F803 01AB
        4000 0000 7E00 08C8 0000 0000 447A 0000
        0000 0000 0000 0000 4479 8000 4000 0000
        4479 C000 3F80 0000 4248 0000 42C8 0000
        4120 0000 3F80 0000

```

Figure 6-8 565 Non-extended Upload File Example (continued)

	SFP 0001:	0801	4D41	5448	2020	2020									
		0000	0002	8063	3F00	0000									
4E20	IN 0000:	002E	414C	4C20	5448	4953	2049	5320	4120	5041	494E	2049			
	IN 0000:	5448	4520	4953	4348	4941	4C20	5455	4245	524F	5349	5459			
2000	IN 0001:	1425	1E00	001F	0001	1F00	021B	221F	0003	1F00	0420	0002			
	IN 0001:	0318	1C19	1F80	0520	0002	181A	24							
0C1C	IN 0002:	1423	1E00	011F	8002	011F	8000	101C	1F80	0009	1C1F	8299			
	IN 0002:	1F80	2C05	1C1F	80DE	081C	24								
0B19	IN 0003:	1439	1E00	021F	8025	0E1F	8015	0719	1F08	DD0A	191F	8100			
	IN 0003:	1F00	D90D	191F	6012	0619	1F70	B10F	191F	00F2	0319	1F81	3302		
	IN 0003:	1921	453B	E000	0419	24									
1F7F	IN 0004:	1441	1E01	C91F	7FEE	0080	1F7F	E900	801A	1F7F	EA00	801A			
1F7F	IN 0004:	EF00	801A	1F7F	E300	801A	1F7F	EB00	801A	1F7F	E400	801A			
	IN 0004:	EC00	801A	1F7F	E500	801A	1F7F	CB00	801A	24					
1F7F	IN 0005:	1423	1E01	511F	7FCC	0080	1F7F	E800	801B	1F7F	E700	801B			
	IN 0005:	CD00	801B	1F7F	E600	801B	24								
1F7F	IN 0006:	1441	1E01	651F	7804	108F	1F78	0450	261C	1F78	0450	081C			
1F7F	IN 0006:	E200	401C	1F7F	DD00	401C	1F7F	DE00	401C	1F7F	C500	401C			
	IN 0006:	D700	401C	1F7F	DF00	401C	1F7F	D800	401C	24					
1F7F	IN 0007:	1447	1E01	631F	7FE0	0040	1F7F	C300	401C	1F7F	C400	401C			
1F7F	IN 0007:	D900	401C	1F7F	C100	401C	1F7F	CA00	401C	1F7F	C200	401C			
	IN 0007:	DC00	401C	1F7F	DB00	401C	1F7F	C900	401C	1F7F	DA00	401C	24		
	IN 0008:	1417	1E81	2E1F	20B0	1F7C	101C	1F41	001C	1F78	0740	801C	24		
1FFF	IN 0009:	1447	1E71	911F	FFD4	0080	1FFF	D300	801C	1FFF	ED00	801C			
1FFF	IN 0009:	D600	801C	1FFF	C600	401C	1FFF	D200	401C	1FFF	D100	401C			
	IN 0009:	E100	401C	1FFF	C700	401C	1FFF	C800	401C	1FFF	D500	401C	24		
	IN 000A:	1819	1E00	091F	000D	1F00	0F1F	0011	191F	0013	1B1F	0015			

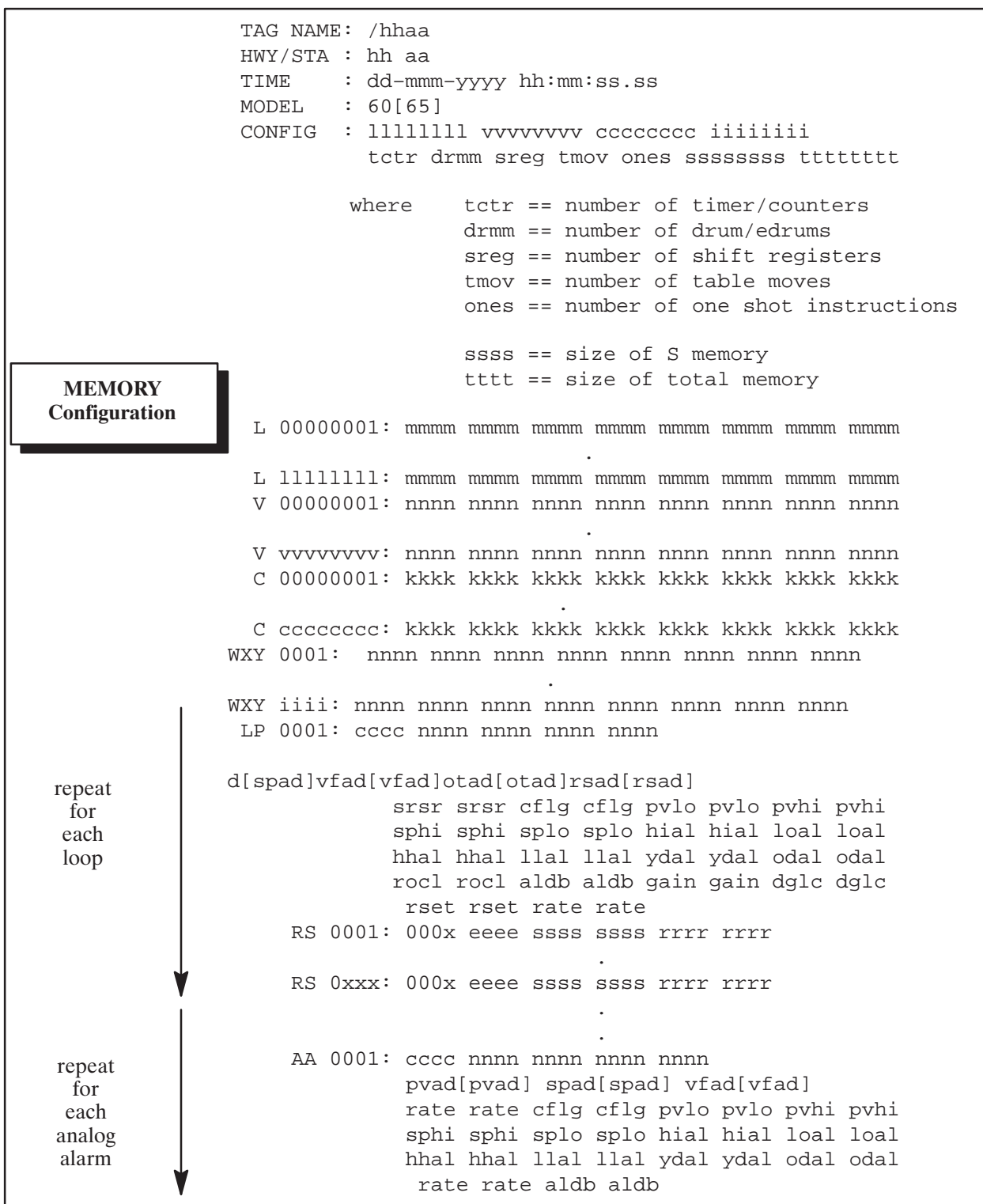
Figure 6-8 565 Non-extended Upload File Example (continued)

## Transfer File Descriptions (continued)

```
1A1C
IN 000A: 24
IN 000B: 180D 1E00 091F 000A 2000 0226 24
IN 000C: 180D 1E00 091F 000A 2000 0225 24
IN 000D: 1816 1E00 091F 0009 1F60 0A1F 700A 1F00 0B14 1312 1124
IN 000E: 180D 1E00 131F 0014 1F00 1615 24
IN 000F: 0806 600C 7070
IN 0010: 0406 090D 0117
SFS 0001: 0C01 5449 4D20 4455 4E4E
IN 0000: 1809 1E00 001F 0001 24
SFS 0002: 0C02 5449 4D20 4455 4E4E
IN 0000: 1809 1E00 001F 0001 24
SFS 0003: 0C03 5449 4D20 4455 4E4E
IN 0000: 1809 1E00 001F 0001 24
.
.
.
SFS 03F7: 0FF7 5449 4D20 4455 4E4E
IN 0000: 1809 1E00 001F 0001 24
SFS 03FE: 0FFE 5449 4D20 4455 4E4E
IN 0000: 1809 1E00 001F 0001 24
SFS 03FF: 0FFF 5449 4D20 4455 4E4E
IN 0000: 1809 1E00 001F 0001 24

IOC 8000: 01 00 FF 00 FF 01 07 00 FF 00 FF 00 FF 46 30 55 55 00
FF...
IOC 800E: 00 00 FF 00 FF
IOC 8010: 01 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
FF...
IOC 801F: 00 00 FF
IOC 8020: 01 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
FF...
IOC 802F: 00 00 FF
.
.
.
IOC 85D0: 01 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
FF...
IOC 85DF: 00 00 FF
IOC 85E0: 01 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
FF...
IOC 85EF: 00 00 FF
IOC 85F0: 01 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
FF...
IOC 85FF: 00 00 FF
ST 0000: 64
```

Figure 6-8 565 Non-extended Upload File Example (continued)



## Transfer File Descriptions (continued)

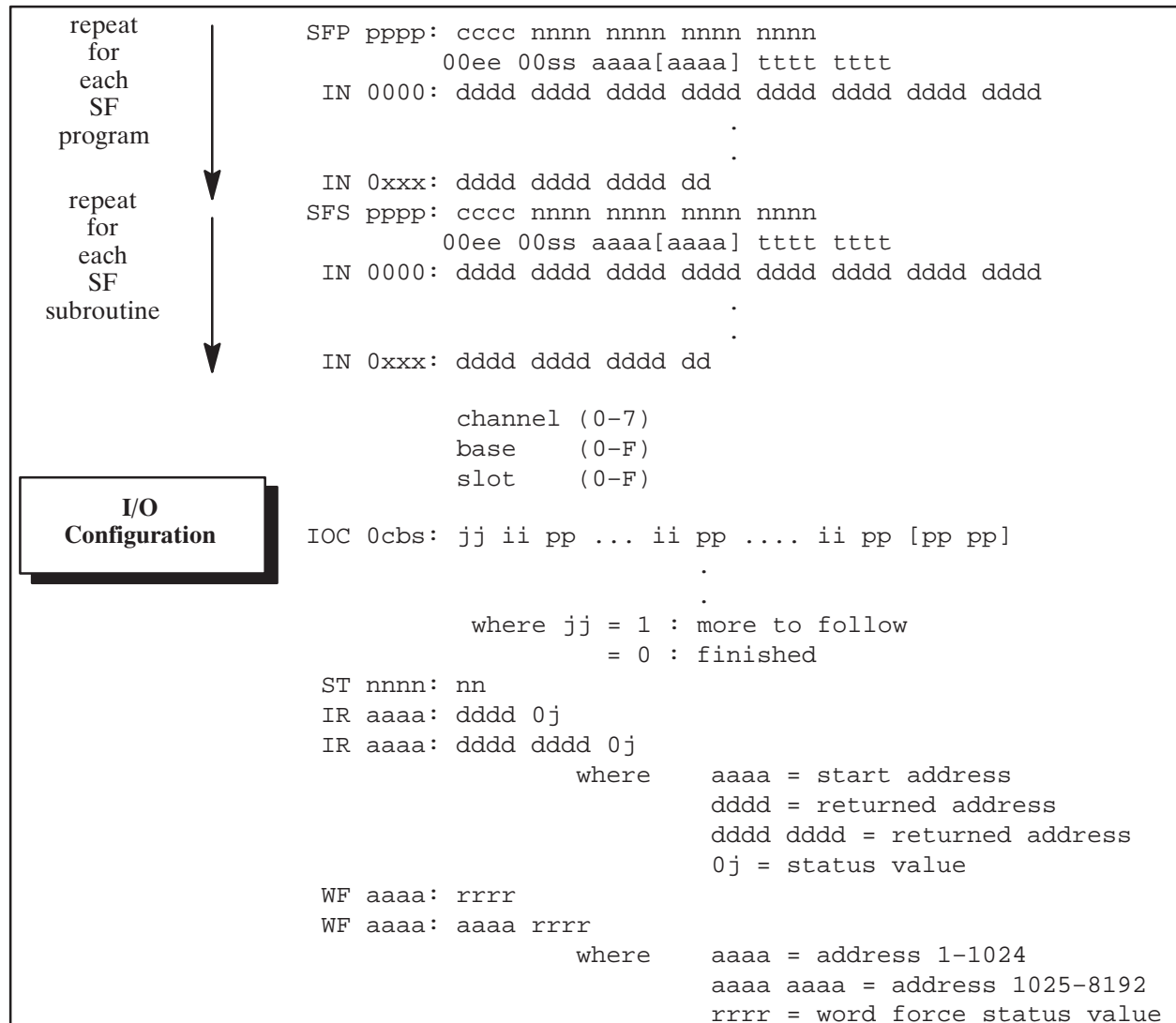


Figure 6-9 560/565 Extended Upload File Format (continued)

```

TAG NAME:565
HWY/STA: 07 22
TIME: 22-JUL-1995 10:54:53.07
MODEL: 65
CONFIG: 00004800 00001400 00002000 000003E8
        00000800 00000106 00001000 00000800 00001000 0000C800 00040000
L 00000001: 8206 048C 8206 64AA 8205 64A0 B101 0001
L 00000009: 0064 0064 00C8 014D 01BC 7001 002B 7001
L 00000011: 009A 7001 0109 7001 0178 7001 01E7 7002
        .
        .
        .
L 000047E9: 8000 8000 8000 8000 8000 8000 8000 8000
L 000047F1: 8000 8000 8000 8000 8000 8000 8000 8000
L 000047F9: 8000 8000 8000 8000 8000 8000 8000 8000
V 00000001: 335A 0000 0100 0000 0000 0000 0600 0001
V 00000009: 0700 0102 0304 0506 0708 0000 0000 0000
V 00000011: 0000 0000 0159 0003 002D 01B3 0038 0000
        .
        .
        .
V 000013E9: 0238 0038 0007 0004 0000 0000 0038 0000
V 000013F1: 01C9 0000 0038 0004 0038 0000 002D 0006
V 000013F9: 01C9 0237 002D 0017 8000 7FFF FFFF 270F
C 00000001: 19AD B0CF 0159 0159 0159 11B4 0216 0285
C 00000009: 02FD 223D 036C 0285 0159 0D7B 11D2 11AE
C 00000011: 223C 223C 1DE5 11AE 1490 1DD3 0312 1934
        .
        .
        .
C 00001FE9: 0004 0000 3C23 D70A 378B CF65 0005 0000
C 00001FF1: 3C23 D70A 378B CF65 E006 0000 0000 0258
C 00001FF9: 3C23 D70A E007 0000 0000 0258 3C23 D70A
LP 0001: 4001 3135 2052 2D53 0000
        6064 7000 0064 7064 0065
        4000 0000 3E23 6064 0000 0000 4049 0FF8
        4049 0FF8 0000 0000 4049 0FF8 4000 0057
        4049 0FF8 3F80 0057 4049 0FF8 4049 0FF8
        3F80 0000 3F80 0057 2141 1BC7 3F80 0000
        452B CEFF 3C23 D70A
RS 0000: 0000 0001 3F80 0000 3F80 0000
RS 0001: 0000 0002 3F80 0000 3F80 0000
RS 0002: 0000 0003 3F80 0000 3F80 0000
RS 0003: 0000 0004 3F80 0000 3F80 0000
RS 0004: 0000 0005 3F80 0000 3F80 0000
RS 0005: 0001 6006 3F80 0000 3F80 0000
RS 0006: 0001 6007 3F80 0000 3F80 0000

```

Figure 6-10 565 Extended Upload File Example

## Transfer File Descriptions (continued)

```

RS 0007: 0001 6008 3F80 0000 3F80 0000
RS 0008: 0001 6009 3F80 0000 3F80 0000
RS 0009: 0001 600A 3F80 0000 3F80 0000
RS 000A: 0000 000B 3F80 0000 3F80 0000
RS 000B: 0000 000C 3F80 0000 3F80 0000
RS 000C: 0000 000D 3F80 0000 3F80 0000
RS 000D: 0000 000E 3F80 0000 3F80 0000
RS 000E: 0000 000F 3F80 0000 3F80 0000
      .
      .
      .
LP 0040: 4040 3135 2052 2D53 0000
      6064 7000 0064 7064 0065
      4000 0000 3E23 6064 0000 0000 42C8 0000
      42C8 0000 0000 0000 42C4 0000 4000 0000
      42C6 0000 3F80 0000 40A0 0000 4120 0000
      3F80 0000 3F80 0000 3F80 0000 3F80 0000
      4479 C000 3C23 D70A
RS 0000: 0000 0001 3F80 0000 3F80 0000
RS 0001: 0000 0002 3F80 0000 3F80 0000
RS 0002: 0000 0003 3F80 0000 3F80 0000
RS 0003: 0000 0004 3F80 0000 3F80 0000
RS 0004: 0000 0005 3F80 0000 3F80 0000
RS 0005: 0001 6006 3F80 0000 3F80 0000
RS 0006: 0001 6007 3F80 0000 3F80 0000
RS 0007: 0001 6008 3F80 0000 3F80 0000
RS 0008: 0001 6009 3F80 0000 3F80 0000
RS 0009: 0001 600A 3F80 0000 3F80 0000
RS 000A: 0000 000B 3F80 0000 3F80 0000
RS 000B: 0000 000C 3F80 0000 3F80 0000
RS 000C: 0000 000D 3F80 0000 3F80 0000
RS 000D: 0000 000E 3F80 0000 3F80 0000
RS 000E: 0000 000F 3F80 0000 3F80 0000
AA 0001: 0401 4F52 2020 4E4F 2020
      60C8 6000 00C8
      4000 0000 7E00 08C8 3F80 0000 44FA 0000
      44BB 8000 4348 0000 44C8 0067 4348 0684
      44E0 FF34 42C8 06A4 0000 0000 0000 0000
      0000 0000 3FFF DF3C
      .
      .
AA 0080: 0480 4F52 2020 4E4F 2020
      60C8 6000 F803 01AB
      4000 0000 7E00 08C8 0000 0000 447A 0000
      0000 0000 0000 0000 4479 8000 4000 0000
      4479 C000 3F80 0000 4248 0000 42C8 0000
      4120 0000 3F80 0000

```

Figure 6-10 565 Extended Upload File Example (continued)

```

SFP 0001: 0801 4D41 5448 2020 2020
          0000 0002 8063 3F00 0000

IN 0000: 002E 414C 4C20 5448 4953 2049 5320 4120 5041 494E 2049 4E20
          IN 0000: 5448 4520 4953 4348 4941 4C20 5455 4245 524F 5349 5459
          IN 0001: 1425 1E00 001F 0001 1F00 021B 221F 0003 1F00 0420 0002
2000
          IN 0001: 0318 1C19 1F80 0520 0002 181A 24
          IN 0002: 1423 1E00 011F 8002 011F 8000 101C 1F80 0009 1C1F 8299
0C1C
          IN 0002: 1F80 2C05 1C1F 80DE 081C 24
          IN 0003: 1439 1E00 021F 8025 0E1F 8015 0719 1F08 DD0A 191F 8100
0B19
          IN 0003: 1F00 D90D 191F 6012 0619 1F70 B10F 191F 00F2 0319 1F81
3302
          IN 0003: 1921 453B E000 0419 24
          IN 0004: 1441 1E01 C91F 7FEE 0080 1F7F E900 801A 1F7F EA00 801A
1F7F
          IN 0004: EF00 801A 1F7F E300 801A 1F7F EB00 801A 1F7F E400 801A
1F7F
          IN 0004: EC00 801A 1F7F E500 801A 1F7F CB00 801A 24
          IN 0005: 1423 1E01 511F 7FCC 0080 1F7F E800 801B 1F7F E700 801B
1F7F
          IN 0005: CD00 801B 1F7F E600 801B 24
          IN 0006: 1441 1E01 651F 7804 108F 1F78 0450 261C 1F78 0450 081C
1F7F
          IN 0006: E200 401C 1F7F DD00 401C 1F7F DE00 401C 1F7F C500 401C
1F7F
          IN 0006: D700 401C 1F7F DF00 401C 1F7F D800 401C 24
          IN 0007: 1447 1E01 631F 7FE0 0040 1F7F C300 401C 1F7F C400 401C
1F7F
          IN 0007: D900 401C 1F7F C100 401C 1F7F CA00 401C 1F7F C200 401C
1F7F
          IN 0007: DC00 401C 1F7F DB00 401C 1F7F C900 401C 1F7F DA00 401C 24
          IN 0008: 1417 1E81 2E1F 20B0 1F7C 101C 1F41 001C 1F78 0740 801C 24
          IN 0009: 1447 1E71 911F FFD4 0080 1FFF D300 801C 1FFF ED00 801C
1FFF
          IN 0009: D600 801C 1FFF C600 401C 1FFF D200 401C 1FFF D100 401C
1FFF
          IN 0009: E100 401C 1FFF C700 401C 1FFF C800 401C 1FFF D500 401C 24
          IN 000A: 1819 1E00 091F 000D 1F00 0F1F 0011 191F 0013 1B1F 0015

```

Figure 6-10 565 Extended Upload File Example (continued)



## Transfer File Descriptions (continued)

```
1A1C
IN 000A: 24
IN 000B: 180D 1E00 091F 000A 2000 0226 24
IN 000C: 180D 1E00 091F 000A 2000 0225 24
IN 000D: 1816 1E00 091F 0009 1F60 0A1F 700A 1F00 0B14 1312 1124
IN 000E: 180D 1E00 131F 0014 1F00 1615 24
IN 000F: 0806 600C 7070
IN 0010: 0406 090D 0117
SFS 0001: 0C01 5449 4D20 4455 4E4E
IN 0000: 1809 1E00 001F 0001 24
SFS 0002: 0C02 5449 4D20 4455 4E4E
IN 0000: 1809 1E00 001F 0001 24
SFS 0003: 0C03 5449 4D20 4455 4E4E
IN 0000: 1809 1E00 001F 0001 24
.
.
.
SFS 03F7: 0FF7 5449 4D20 4455 4E4E
IN 0000: 1809 1E00 001F 0001 24
SFS 03FE: 0FFE 5449 4D20 4455 4E4E
IN 0000: 1809 1E00 001F 0001 24
SFS 03FF: 0FFF 5449 4D20 4455 4E4E
IN 0000: 1809 1E00 001F 0001 24

IOC 8000: 01 00 FF 00 FF 01 07 00 FF 00 FF 00 FF 46 30 55 55 00
FF...
IOC 800E: 00 00 FF 00 FF
IOC 8010: 01 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
FF...
IOC 801F: 00 00 FF
IOC 8020: 01 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
FF...
IOC 802F: 00 00 FF
.
.
.
IOC 85D0: 01 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
FF...
IOC 85DF: 00 00 FF
IOC 85E0: 01 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
FF...
IOC 85EF: 00 00 FF
IOC 85F0: 01 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00
FF...
IOC 85FF: 00 00 FF
ST 0000: 64
```

Figure 6-10 565 Extended Upload File Example (continued)

```

TAG NAME:#aaaa | /aaaa
HWY/STA: nn nn
TIME:    dd-mmm-yyyy hh:mm:ss.ss
MODEL:   65
CONFIG:  llll vvvv cccc iiii
          tctr drmm sreg tmov ones ssss tttt

          where      tctr == number of timer/counters
                    drmm == number of drum/edrums
                    sreg == number of shift registers
                    tmov == number of table moves
                    ones == number of one shot instructions
                    ssss == size of S memory
                    tttt == size of total memory

P :  yyyy ww zz dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ..dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ..dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ..dd
      .
      .
P :  yyyy ww zz dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ..dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ..dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ..dd

D :  yyyy ww zz dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ..dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ..dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ..dd
      .
      .
D :  yyyy ww zz dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ..dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd .. dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ..dd

```

Figure 6-11 560/565 Generic Upload File Format

## Transfer File Descriptions (continued)

```

TAG NAME:    /0202
HWY/STA:    02 02
TIME:       10-SEP-1995 14:11:42.02
MODEL:      65
CONFIG:     00002000 00000800 00000800 00000028
            00000400 00000083 00000800 00000400 00000800 00002C00 00040000
P : 0000 00 00 15 55 01 FF 00 10 00 04 00 04 00 05 00 06 00 02 ...
      03 07 00 FF 04 07 02 39 00 05 57 40 01 F0 01 1B ...
      84 0C 8C 0E 8A 0A 8A 0B 1B 1E 00 0C 98 0E 84 0B ...
      12 1B 1E 00 18 8C 08 8A 10 8A 11 98 08 84 11 8C ...
      95 1F 98 21 84 01 D6 01 01 BE 01 B7 00 20 00 20 ...
P : 0001 00 00 1B 1E 00 30 A0 25 00 3C 98 22 84 22 F5 01 01 B7 ...
      20 98 1F 84 03 D6 02 01 BF 01 B8 00 20 00 20 98 ...
      01 B8 01 BF 00 00 98 14 84 01 D0 01 01 BE 1B 1E ...
      25 E4 A8 84 31 E8 A9 1B 1E 00 60 94 00 E4 A9 84 ...
      1B 1E 00 6C E4 A3 E8 AF 94 00 90 00 E0 A3 82 02 ...
      65 86 66 A0 1E 00 05 98 66 84 66 88 6A 84 66 8A ...
      98 6B 98 1E 84 67 84 6A 8A 68 94 00 98 6A
      .
      .
P : 0006 00 00 1B 51 0B E9 42 C8 00 00 00 00 00 00 42 C8 00 00 ...
      00 00 00 00 00 42 C8 00 00 00 00 00 00 43 96 00 ...
      43 96 00 00 00 00 00 00 43 96 00 00 00 00 1B 51 ...
      00 43 96 00 00 00 00 1B 51 78 01 08 19 43 96 00 ...
      FF 51
D : 0007 00 01 01 40 02 40 03 40 04 40 05 40 06 40 07 40 08 40 ...
      40 19 40 20 40 21 40 99 99 AA AA BB BB CC CC DD ...
      05 00 05 00 05 00 05 00 05 00 05 00 05 00 05 00 03 00 ...
      00 05 00 05 00 05 00 22 22 00 00 00 00 00 00 00 ...
      05 00 05 00 05 00 FF FF FF FF FF FF FF FF FF ...
      00 05 00 05 00 03 00 00 00 00 00 00 00 05 00 05 ...
      06 41 07 41 08 41 09 41 10 41 05 00 05 00 05 00 ...
      00 00 00 00 00 00 00 05 00 0A 00 00 00
      .
      .
D : 0020 00 01 04 49 05 49 06 49 07 49 08 49 09 49 10 49 00 00 ...
      04 00 00 FF FF 00 04 00 00 00 00 00 04 00 00 00 ...
      39 49

```

Figure 6-12 560/565 Generic Upload File Example

```

TAG NAME: #hhaa
HWY/STA : hh aa
TIME    : dd-mmm-yyyy hh:mm:ss.ss
MODEL   : 80 | 208
CONFIG  : 1111 vvvv cccc 0000

L 0001: mmmm mmmm mmmm mmmm mmmm mmmm mmmm mmmm
L 0009: mmmm mmmm mmmm mmmm mmmm mmmm mmmm mmmm
.
L 1111: mmmm mmmm mmmm mmmm mmmm mmmm mmmm mmmm
V 0001: nnnn nnnn nnnn nnnn nnnn nnnn nnnn nnnn
V 0009: nnnn nnnn nnnn nnnn nnnn nnnn nnnn nnnn
.
V vvvv: nnnn nnnn nnnn nnnn nnnn nnnn nnnn nnnn
C 0001: kkkk kkkk kkkk kkkk kkkk kkkk kkkk kkkk
C 0009: kkkk kkkk kkkk kkkk kkkk kkkk kkkk kkkk
.
C cccc: kkkk kkkk kkkk kkkk kkkk kkkk kkkk kkkk

```

Figure 6-13 PM550 and IT-160 MRCU Upload File Format

```

TAG NAME: #0B56
HWY/STA: 0B 56
TIME:    19-JUN-1995 14:47:23.47
MODEL:   80
CONFIG:  1000 0400 0800 0000
L 0001: 0000 0000 0000 0000 0000 0000 0000 0000
L 0009: 0000 0000 0000 0000 0000 0000 0000 0000
.
L 0FF9: 0000 0000 0000 0000 0000 0000 0000 0000
V 0001: 0000 0000 0000 0000 0000 0000 0000 0000
V 0009: 0000 0000 0000 0000 0000 0000 0000 0000
.
V 03F9: 0000 0000 0000 0000 0000 0000 0000 0000
C 0001: 0000 0000 0000 0000 0000 0000 0000 0000
C 0009: 0000 0000 0000 0000 0000 0000 0000 0000
.
C 07F9: 0000 0000 0000 0000 0000 0000 0000 0000

```

Figure 6-14 PM550 Upload File Example

## Transfer File Descriptions (continued)

---

```
TAG NAME:#aaaa
HWY/STA: nn nn
TIME:    dd-mmm-yyyy hh:mm:ss.ss
MODEL:   100
CONFIG:  llll vvvv 0000 0000 0000 tttttttt

                where tttttttt == size of total memory

P : yyyy ww zz dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ...dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ...dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ...dd
      .
      .
P : yyyy ww zz dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ...dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ...dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd .. dd
D : yyyy ww zz dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd .. dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ...dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd .. dd
      .
      .
D : yyyy ww zz dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ...dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ...dd
      dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd ...dd

(The example above depicts the binary format)
---or---
P : yyyy ww zz aaaaa ... aaaaa
      .
      .
P : yyyy ww zz aaaaa ... aaaaa
D : yyyy ww zz aaaaa ... aaaaa
      .
      .
D : yyyy ww zz aaaaa ... aaaaa

(The example above depicts the ASCII format)
```

Figure 6-15 Unilink Generic Upload File Format

```

TAG NAME: #0201
HWY/STA: 02 01
TIME: 22-SEP-1995 12:44:28.49
MODEL: 100
04E0 03AB 0000 0000 0000 00001000
P : 0000 00 00 00 04 00 00 00 00 01 04 01 00 01 10 02 04 02 00 ...
    50 06 04 06 00 06 60 07 04 07 00 07 70 08 04 08 ...
    00 00 00 04 00 00 00 00 00 04 00 00 00 00 00 04 ...
    00 00 00 00 04 00 00 00 00 00 04 00 00 00 00 00 ...
    00 00 00 00 00 04 00 00 00 00 00 04 00 00 00 00 ...
    04 00 00 00 00 00 04 00 00 00 00 00 04 00 00 00 ...
    00 04 00 00 00 00 00 04 00 00 00 00 00 04 00 00 ...
    00 00 04 00 00 00 00 00 04 00 00 00 00
P : 0001 00 00 00 04 00 00 00 00 04 00 00 00 00 04 00 00 ...
    00 00 04 00 00 00 00 00 04 00 00 00 00 00 04 00 ...
    00 00 00 04 00 00 00 00 00 04 00 00 00 00 00 04 ...
    00 00 00 00 04 00 00 00 00 00 04 00 00 00 00 00 ...
    00 00 00 00 00 04 00 00 00 00 00 04 00 00 00 00 ...
    04 00 00 00 00 00 04 00 00 00 00 00 04 00 00 00 ...
    00 04 00 00 00 00 00 04 00 00 00 00 00 04 00 00 ...
    00 00 04 00 00 00 00 00 04 00 00 00 00
.
.
P : 0009 00 00 00 04 00 00 00 00 04 00 00 00 00 04 00 00 ...
    00 00 04 00 00 00 00 00 04 00 00 00 00 00 04 00 ...
    00 00 8F 15 8F 81 8F F1 90 15 90 91 90 01 91 15 ...
    91 94 41 95 15 95 91 95 51 96 15 96 91 96 61 97 ...
    9A 91 9A A1 9B 15 9B 91 9B B1 9C 15 9C 91 9C C1 ...
D : 000A 00 01 01 40 02 40 03 40 04 40 05 40 06 40 07 40 08 40 ...
    40 19 40 20 40 21 40 99 99 AA AA BB BB CC CC DD ...
    05 00 05 00 05 00 05 00 05 00 05 00 05 00 03 00 ...
    00 05 00 05 00 05 00 22 22 00 00 00 00 00 00 00 ...
    05 00 05 00 05 00 FF FF FF FF FF FF FF FF FF ...
    00 05 00 05 00 03 00 00 00 00 00 00 00 05 00 05 ...
    06 41 07 41 08 41 09 41 10 41 05 00 05 00 05 00 ...
    00 00 00 00 00 00 00 05 00 0A 00 00 00
.
.
D : 0011 00 01 04 49 05 49 06 49 07 49 08 49 09 49 10 49 00 00 ...
    04 00 00 FF FF 00 04 00 00 00 00 00 04 00 00 00 ...
    39 49

```

Figure 6-16 Unilink Generic Upload File Example

## Transfer File Descriptions (continued)

```

TAG NAME: #aaaa
HWY/STA : nn nn
TIME    : dd-mmm-yyyy hh:mm:ss.ss
MODEL   : 100
CONFIG  : llll vvvv 0000 0000 0000 tttttttt

                where      tttttttt == size of total memory

PROGRAM : AAA .. AAA { program name }
SS      : mmmm       { specific segment mask, where

                +-----+
                |e|1|C|D|E|F|G|H|I|J|K|L|M|N|O|P|
                +-----+
                | | | | | | | | | | | | | | | |
e = 0 : non-extended mask  --+ +-----+
    1 : extended mask      Set to select
                           specific segment }

AAA .. AAA { segment name }
YYYY ww zz dd dd dd dd dd dd dd dd dd dd dd dd dd dd .. dd
                dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd .. dd
                dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd .. dd
                .
                .
YYYY ww zz dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd .. dd
                dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd .. dd
                dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd dd .. dd

(The example above depicts the binary format)
                --or--
AAA .. AAA { segment name }
YYYY ww zz aaaaa ... aaaaa
YYYY ww zz aaaaa ... aaaaa
                .
                .
AAA .. AAA { segment name }
YYYY ww zz aaaaa ... aaaaa
YYYY ww zz aaaaa ... aaaaa

(The example above depicts the ASCII format)

```

Figure 6-17 Unilink Specific Segment Upload File Format

```

TAG NAME: #0101
HWY/STA:  01 01
TIME:     4-NOV-1995 10:08:05.99
MODEL:    100
          0000 09D7 0000 0000 0000 000009D7
PROGRAM :
  SS : 4001
      Spindle Map
      0000 01 00 SMAP CM=RWL TD=RWL TR=RWL TH=RWL TL=RWL TP=RWL TO=RWL
      0001 01 00 SMAP TPK=R** JT=RWL AH=RWL AL=RWL AC=RWL AO=RWL APK=R**
      0002 01 00 SMAP SI=RWL SH=RWL SL=RWL YP=RWL SS=RWL ACK=RWL
  SS : 4002
      N+1 MAP
      0000 01 01 NMAP TM=RWL DT=RWL YR=RWL MCH(01)=RWL MCH(02)=RWL
      0001 01 01 NMAP MCH(05)=RWL MCH(06)=RWL MCH(07)=RWL MCH(08)=RWL
      0002 01 01 NMAP MCH(10)=RWL ACK=RWL
  SS : 4004
      SPINDLE VALUES
  SS : 4008
      N+1 VALUES
  SS : 4010
      VALUESLUES N+1 VALUESL L MCH(07)=RWL MCH(08)=RWL MCH(09)=
  SS : 4020
      VALUESLUES N+1 VALUESL L MCH(07)=RWL MCH(08)=RWL MCH(09)=
  SS : 4040
      VALUESLUES N+1 VALUESL L MCH(07)=RWL MCH(08)=RWL MCH(09)=
  SS : 4080
      VALUESLUES N+1 VALUESL L MCH(07)=RWL MCH(08)=RWL MCH(09)=
  SS : 4100
      VALUESLUES N+1 VALUESL L MCH(07)=RWL MCH(08)=RWL MCH(09)=
  SS : 4200
      VALUESLUES N+1 VALUESL L MCH(07)=RWL MCH(08)=RWL MCH(09)=
  SS : 4400
      VALUESLUES N+1 VALUESL L MCH(07)=RWL MCH(08)=RWL MCH(09)=
  SS : 4800
      VALUESLUES N+1 VALUESL L MCH(07)=RWL MCH(08)=RWL MCH(09)=
  SS : 5000
      VALUESLUES N+1 VALUESL L MCH(07)=RWL MCH(08)=RWL MCH(09)=
  SS : 6000
      VALUESLUES N+1 VALUESL L MCH(07)=RWL MCH(08)=RWL MCH(09)=

```

Figure 6-18 Unilink Specific Segment Upload File Example



# Interactive Operator Utilities

---

<b>7.1</b>	<b>Using TIUSER and TIPROG</b> .....	<b>7-2</b>
<b>7.2</b>	<b>TIUSER Utility</b> .....	<b>7-3</b>
	ACTIVATE Command .....	7-5
	CHANGESTATE Command .....	7-6
	DEACTIVATE Command .....	7-7
	DOWNLOAD Command .....	7-8
	FILESTATUS Command .....	7-11
	LIST Command .....	7-11
	STATISTICS Command .....	7-11
	UPLOAD Command .....	7-12
	VERIFY Command .....	7-15
	QUIT Command .....	7-15
<b>7.3</b>	<b>TIPROG Utility</b> .....	<b>7-16</b>
	Session Control Commands .....	7-20
	Host Adapter Commands .....	7-21
	Base HIU Commands .....	7-21
	TIWAY Primitive Commands .....	7-26
	CIM Functional Commands .....	7-32
	Support Commands .....	7-37

## 7.1 Using TIUSER and TIPROG

---

Two Interactive Operator Utilities, TIUSER and TIPROG, are generic to the TIWAY Host software package. These utilities allow a standard interface to common network functions and require no applications programming. TIUSER and TIPROG commands are presented in alphabetical order. Examples of user responses to utility prompts are also included. Table 7-1 provides basic information on using the two utilities.

Table 7-1 Using TIUSER and TIPROG

	<b>TIUSER</b>	<b>TIPROG</b>
To run, issue	TIUSER	TIPROG
The utility prompt is	TIUSER>	TIPROG>
To exit utility, issue	QUIT	Q
To exit command, issue	<CTRL>Z	<CTRL>Z

- TIUSER (Section 7.2) provides network management functions and can be used to simplify procedures such as connecting or disconnecting Secondaries from the network.
- TIPROG (Section 7.3) provides an interactive programmer interface to all TIWAY library subroutines. Its primary use is as a learning tool that allows you to become familiar with the software package. It is also useful in testing your applications before coding them.
- You can type HELP within TIUSER or HE within TIPROG at the utility prompt to receive a listing of all options in that utility.
- The <RETURN> key is used to carry out the command or display the next command prompt.
- <CTRL>Z exits the command and returns you to the utility prompt.
- In TIPROG, you must issue the initialize command (IN) before any other subroutine can be used.

---

**NOTE:** In TIUSER, you are prompted to Enter Port Setup String when you request the TIUSER utility. Enter a port, baud rate, data bits, stop bit, even or odd parity, retry count, and timeout (for example: P1,19200,7,1,E,3,A).

In TIPROG, you must issue the initialize command (IN) before any other subroutine can be used.

---

For more information on any of these commands, refer to the detailed corresponding subroutine descriptions given in Chapter 10 and to the appropriate manuals listed in the Preface.

## 7.2 TIUSER Utility

---

Table 7-2 shows the ten functions that the TIUSER utility provides. This utility requires a configured network, since this utility does not support UHA configuration commands.

Table 7-2 Alphabetized List of TIUSER Commands

<b>TIUSER Command</b>	<b>Function</b>
ACTIVATE	Connects an attached device to the highway.
CHANGESTATE	Changes attached device state.
DEACTIVATE	Disconnects an attached device from the highway.
DOWNLOAD	Downloads file to attached device memory.
FILESTATUS	Shows attached device memory file status.
LIST	Lists all connected Secondaries.
STATISTICS	Lists attached device TIWAY statistics.
UPLOAD	Uploads attached device memory to file.
VERIFY	Describes how to compare two uploaded files.
QUIT	Terminates utility.

All TIUSER commands operate on either a CIM- or NIM-based attached devices, and HIUs or Secondary adapters.

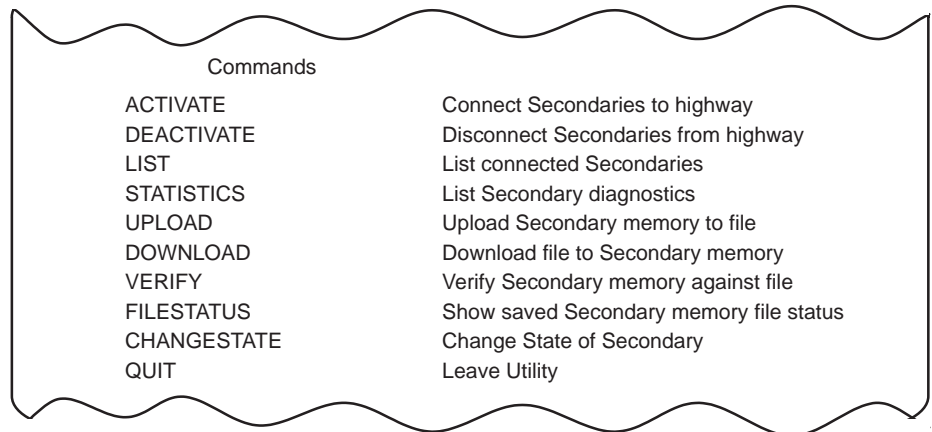
All addresses are specified in the standard addressing format (see Section 3.5), either by tag name, direct ASCII specification, or direct binary specification. Many of the commands require only the specification of the highway or the highway and Secondary address.

---

**NOTE:** You must type at least the first four letters of each command at the utility prompt (TIUSER>) to receive the command prompts.

---

When running TIUSER and you are trying to decide which command to use, type in HELP and the screen shown in Figure 7-1 appears.

The image shows a screenshot of the TIUSER Help Menu Screen. It is enclosed in a rectangular frame with wavy, decorative borders on the top and bottom. The word "Commands" is centered at the top. Below it, there are two columns of text. The left column lists the commands: ACTIVATE, DEACTIVATE, LIST, STATISTICS, UPLOAD, DOWNLOAD, VERIFY, FILESTATUS, CHANGESTATE, and QUIT. The right column provides a brief description for each command: Connect Secondaries to highway, Disconnect Secondaries from highway, List connected Secondaries, List Secondary diagnostics, Upload Secondary memory to file, Download file to Secondary memory, Verify Secondary memory against file, Show saved Secondary memory file status, Change State of Secondary, and Leave Utility.

Commands	
ACTIVATE	Connect Secondaries to highway
DEACTIVATE	Disconnect Secondaries from highway
LIST	List connected Secondaries
STATISTICS	List Secondary diagnostics
UPLOAD	Upload Secondary memory to file
DOWNLOAD	Download file to Secondary memory
VERIFY	Verify Secondary memory against file
FILESTATUS	Show saved Secondary memory file status
CHANGESTATE	Change State of Secondary
QUIT	Leave Utility

Figure 7-1 TIUSER Help Menu Screen

---

**ACTIVATE  
Command**

The ACTIVATE command is used to connect a Secondary or Secondaries to the TIWAY network.

The utility attempts to connect the specified Secondary to the highway and prompts for another Secondary to connect. This is repeated until all desired Secondaries are connected. To terminate the command, enter RETURN and the TIUSER> prompt is returned.

The address specification requires only the highway and Secondary address (e.g., #0102 would specify the first highway and Secondary number 02). Any further address specification is ignored.

The LIST command can be used to verify that the Secondary has been connected.

Prompt: TIUSER> ACTI  
Secondary activation –  
Input tag for Secondaries to activate, one per line  
<RETURN> to terminate

Tag specifier:

Response: Tag specifier: #0102 {to activate a highway 1 Secondary #2}  
Tag specifier: <RETURN> {to terminate the command}  
TIUSER>

## TIUSER Utility (continued)

---

### CHANGESTATE Command

The CHANGESTATE command is used to change the state of an attached device. This command has two sets of state options depending on whether the attached device is NIM- or CIM-based.

```
Prompt:  TIUSER> CHAN
          Tag specifier: #0102
          Hex operating state values
            00) execute loops and logic
            01) to execute loops but not logic
            02) to not execute either
          Current state: 02
          Are you sure you want to change state?: Y
          Desired state: 00
```

```
Response: Return state: 00
          TIUSER>
```

The utility prompts you for a Secondary address. Depending on whether the attached device is CIM- or NIM-based, you are presented with one of two lists of options. The option list for a NIM-based attached device is shown in the example above. If the attached device is CIM-based, the option list is

```
Enter    1) remote mode,      2) non-remote mode,
          3) program mode,    4) non-program mode
```

Type in a number representing the desired option, and the Secondary state is changed to that option.

### **WARNING**

A configuration switch (/CHANGESTATE) is provided to disallow (default) or allow remote change state operations.

In many applications, to remotely start and stop machines or processes could cause unpredictable operation that could result in death or serious injury and/or damage to equipment.

Remote change state operations should only be allowed at sites where no hazard would result. (The /CHANGESTATE option is only checked prior to this subroutine. TIPROG or user-written applications are not disallowed.)

---

**DEACTIVATE  
Command**

The DEACTIVATE command is used to disconnect a Secondary or Secondaries from the TIWAY network. You are prompted for a Secondary address. The utility attempts to disconnect the specified Secondary from the highway and prompts for another Secondary to disconnect. This is repeated until all desired Secondaries are disconnected. To terminate the command, enter RETURN and the TIUSER> prompt is returned.

The address specification requires only the highway and Secondary address (e.g., #0102 would specify the first highway and Secondary number). Any further address specification is ignored.

The LIST command can be used to verify that the Secondary has been disconnected.

Prompt: TIUSER> DEAC  
Secondary deactivation –

Input tag for Secondaries to deactivate, one per line  
<RETURN> to terminate

Tag specifier:

Response: Tag specifier: #0102 {to deactivate a Secondary}  
Tag specifier:<RETURN> {to terminate the command}  
TIUSER>

## TIUSER Utility (continued)

---

### DOWNLOAD Command

The **DOWNLOAD** command is used to download a file to an attached device. The file that is downloaded is one that was previously uploaded from a Secondary. Before you download to the Secondary, you should put the Secondary into the proper state for downloading. This command supports:

- Full memory download
- Partial memory download
- Generic download
- Specific segment download (Unilink only)
- Download between different devices

After you request **DOWNLOAD**, you are prompted for the file specification (filename). The file characteristics, including the creation date, highway, Secondary address, and attached device model number, are displayed for verification, which you are asked to supply. You are then prompted for the Secondary address to receive the download. If the highway or Secondary address does not match the file characteristics, you are asked to verify that this mismatch is your intention. The specified attached device is checked to see if it is NIM- or CIM-based. If it is not, the download cannot proceed. Finally, the download is attempted.

Only memory types contained within the upload file are downloaded. (If the file does not contain V-memory, for example, **DOWNLOAD** proceeds to constant memory; no error message is displayed.) If any **TIWAY** error occurs during **DOWNLOAD**, the operation stops and an error message is displayed. Chapter 6 contains more information on transfer files.

In the example on the next two pages, a file that was previously uploaded from highway 1, Secondary 12, is downloaded to highway 1, Secondary 9. You were requested to verify the download to a different Secondary. Because the response was yes, the download was performed.

---

**NOTE:** If the upload file contains extended addressing, the extended version of the tag specifier must be used (i.e., /hhaa, not #hhaa).

If you are downloading to a NIM-based attached device, you can edit the file, and download only the portions of it that you want.

---



---

Prompt: TIUSER> DOWN  
File specification: C:\TIWY6911\MYFILE.EXT  
File created on 10-JUN-95 17:34:19  
Uploaded from highway 1 station 12 Model 550  
Is this the desired file: Y

>>> If answer = "yes" ---

Prompt: Tag specifier: #0109  
Highway or station mismatch, proceed? Y

>>> If answer = "yes" {and the transfer is permitted by the package} ---

Prompt: Select specific segment download (y/n) ? : {displayed for  
Unilink only}

>>> If answer = "yes" ---

Prompt: Select segment mask (hex) : HHHH

Response: Begin specific segment download . . .  
Downloading segment 1 mask value = HHHH ...  
.  
Downloading segment n mask value = HHHH ...  
Specific segment download completed  
TIUSER>

Response: No segments specified for specific transfer {no bits set in  
mask}  
TIUSER>

Response: WARNING: Not all specific segments transferred {error in  
downloading requested segment}  
TIUSER>

>>> If answer (to specific segment download request) = "no" ---

Prompt: Select Generic Download (y/n) ? :

>>> If answer = "yes" ---

Prompt: Select Generic Download configuration:  
1 - Download all memory  
2 - Download Program Memory only  
3 - Download Data Memory only  
Enter selection (1-3) : 3

Response: Begin Generic Download . . .  
Downloading Program Memory ...  
Downloading Data Memory ...  
Generic Download completed  
TIUSER>

## TIUSER Utility (continued)

---

>>> If answer (to generic download request) = "no" ---

Select transfer code ---

90 : download all except WXY, IR, and WF	94 : download S-memory – loop tables
91 : download L-memory	95 : download S-memory – analog alarms
92 : download V-memory	96 : download S-memory – SF programs
93 : download constant memory	97 : download S-memory – SF subroutines
	98 : download IR, WF memory
	99 : download WXY memory

Enter transfer code : 90

Response: Downloading program . . .  
Download [memory type] ...  
Download completed

TIUSER>

>>> If you attempt to download from a device type unrecognized by the package to the same device or another of the same type, the following series of prompts/messages appears. (Note that only a generic download is attempted.)

Prompt: TIUSER> DOWN  
File specification: C:\TIWY6911\MYFILE.EXT  
File created on 10-JUN-95 17:34:19  
Uploaded from highway 1 station 12 Model ???  
Is this the desired file: Y

>>> If answer = "yes" ---

Prompt: Tag specifier: #0109

>>> If the transfer is between like devices ---

Response: \*\*\* WARNING. UNKNOWN DEVICE TYPE: xxx

Select Generic Download configuration:

- 1 – Download all memory
- 2 – Download Program Memory only
- 3 – Download Data Memory only

Enter selection (1-3) : 3

Response: Begin Generic Download . . .  
Downloading Program Memory ...  
Downloading Data Memory ...  
Generic Download completed

TIUSER>

---

**FILESTATUS  
Command**

The FILESTATUS command is used to list the parameters of a previously uploaded file. These include the date and time of upload, the highway and Secondary address, and the attached device model. This command allows you to check which file is being downloaded to an attached device. You are prompted for the file specification (filename).

Prompt: TIUSER> FILE  
File specification: C:\TIWY6911\MYFILE.EXT  
File created on 20-DEC-95 10:26:11  
Uploaded from highway 1 station 2 model 530

Response: TIUSER>

**LIST Command**

The LIST command is used to list all of the Secondaries connected to a highway.

Prompt: TIUSER> LIST  
Response: Highway 1 has 1 Secondaries  
02  
TIUSER>

The number of Secondaries and a list of Secondaries connected to the highway are displayed.

**STATISTICS  
Command**

The STATISTICS command is used to list the attached device diagnostics statistics that report the TIWAY network usage (reported as an unformatted hex list).

Prompt: TIUSER> STAT  
Tag specifier: #0102

Response: Secondary statistics for tag #

Number of times polled	1
Number of I-frames transmitted	0
Number of timeouts	0
Number of I-frames received	0
Number of I-frames re-transmitted	0
Number of received errors from Secondary	0
Number of transmitted errors to Secondary	0
Number of times Secondary initialized	1

TIUSER>

## TIUSER Utility (continued)

---

**UPLOAD Command** The **UPLOAD** command is used to upload the contents of a Secondary to a file on the host system. This command is intended as a backup for Secondary programs, and supports these functions:

- Full memory upload
- Partial memory upload
- Generic memory upload
- Specific segment upload (Unilink)

You are prompted for a Secondary address and a file specification (filename). Any further address specification is ignored. The filename should take the form of any valid operating system filename specification. The file is created with parameters of the Secondary and the upload session and the memory dump from the Secondary.

Edited (shortened) examples of files uploaded from the Secondaries supported by this package are shown in Chapter 6. To display a file, exit the utility and use the operating system's **TYPE** command or display utility.

Chapter 6 contains more information on transferring files. However, unlike the **UPLOAD** procedure, the **UPLOAD** command treats one occurrence differently. If a **TIWAY** error occurs, the error message is displayed, the message "... upload continued" appears, and an attempt is made to upload the next memory type. As with the **UPLOAD** procedure, an attempt to upload a memory type that is not configured in the Secondary does not halt the operation, which continues to the next memory type. However, **UPLOAD** displays a warning message that the memory type was not found.

```
Prompt:  TIUSER> UPLO
          Tag specifier: #0102
          File specification: C:\TIWY6911\MYFILE.EXT
```

```
Response: *** WARNING. UNKNOWN DEVICE TYPE: xxx
```

This response appears if the device type is not recognized by the package. It is added for support of future products. In this case, the package attempts a generic upload. You can select the Generic Upload configuration.

---

Prompt: Select specific segment upload (y/n) ? : {is displayed only for Unilink}

>>> If answer = "yes" ----

Prompt: Enter program name : (blanks if not supported) ..  
Program name : AAA .. AAA  
Enter segment mask (hex) : HHHH

Response: Begin specific segment upload . . .  
Uploading segment 1 mask value = HHHH ...  
Uploading segment n mask value = HHHH ...  
Specific segment upload completed

TIUSER>

Response: No segments specified for specific transfer {no bits set in mask}

TIUSER>

Response: WARNING: Not all specific segments transferred {error in uploading requested segment}

TIUSER>

>>> If answer (to specific segment upload request) = "no" ----

Prompt: Select Generic Upload (y/n) ? :

>>> If answer = "yes" ----

Prompt: Select Generic Upload configuration:  
1 - Upload all memory  
2 - Upload Program Memory only  
3 - Upload Data Memory only  
Enter selection (1-3) : 1

Response: Begin Generic Upload . . .  
Uploading Program Memory ...  
Uploading Data Memory ...  
Generic Upload completed

TIUSER> (continued on next page)

## TIUSER Utility (continued)

---

>>> If answer (to generic upload request) = "no" ---

Prompt:           Select transfer code --

60 : upload all memory	64 : upload S-memory – loop tables
61 : upload L-memory	65 : upload S-memory – analog alarms
62 : upload V-memory	66 : upload S-memory – SF programs
63 : upload constant memory	67 : upload S-memory – SF subroutines

Enter transfer code : 60

Response:   Uploading program . . .  
              Uploading [memory type] ...  
              Upload completed

TIUSER>

---

**VERIFY Command**

The VERIFY command explains how to verify that the contents of two previously uploaded files are the same. When you request VERIFY at the TIUSER> prompt, a screen explaining the command appears. VERIFY is used to verify that a download of the same file was correct before starting the Secondary, or to ensure that the correct program revision is currently running on a machine. The screen is similar to the one shown below.

Prompt: TIUSER> VERI

Response: To verify the contents of a file with controller memory, upload controller memory using the TIWAY utility TIUSER, UPLOAD command into a second file. The contents of these files can be compared using a file comparison utility.

The syntax of the MS-DOS command and the format of the list of differences is explained in the MS-DOS documentation. The command syntax is:

```
comp file1.spc file2.spc
```

**QUIT Command**

The QUIT command stops running the TIUSER program.

The TIPROG utility provides an interactive programmer interface to all of the subroutines in the TIWAY Subroutine Library. It provides more functions than are available through the TIUSER utility, but at a lower level. TIPROG provides a convenient tool for learning the facilities available in the TIWAY Host Software Package. It is also valuable for manually prototyping application programs or for simulating application programs for diagnostic purposes.

The TIPROG commands are interactive and self-descriptive, and a HELP command (HE) lists the individual commands available. Each command is presented in the following pages with a brief description and an example.

In general, each command prompts for the necessary information, carries out the command, and displays the results and status. The commands available are shown in Table 7-3 and Figure 7-2. Table 7-3 lists the commands in alphabetical order, divided into subroutine types. Figure 7-2 shows the Help Menu Screen given when you access TIPROG and type HE.

### **WARNING**

The TIPROG utility is intended primarily as a tool for you, the TIWAY applications programmer. It provides you with interactive access to the TIWAY Subroutine Library. It prompts you for the required arguments and displays the returned data and status messages.

It should be noted that TIPROG gives full access to all TIWAY functions and you must exercise caution if you are accessing Secondaries which are controlling operating machines and processes. Remote state changes could cause unpredictable controller behavior that could result in death and/or serious injury, or damage to equipment.

Do not use this routine without thoroughly understanding how it might impact operations directed by the controller.



Table 7-3 Alphabetized List of TIPROG Commands with Subroutines and Primitives

<b>TIPROG Command</b>	<b>Function</b>	<b>Equivalent Subroutine</b>	
<b>Session Control Subroutines</b>			
FI	Finish TIWAY Subroutines	FIN – Session control	
IN	Initialize TIWAY Subroutines	INIT – Session control	
<b>TIWAY Primitive Subroutines</b>			
CH	Change attached device state	CHNGST	TIWAY Primitive 10
CN	Get configuration	CONFIG	TIWAY Primitive 03
DE	Define data blocks	DEFBLK	TIWAY Primitive 50
FL	Fill block	FILL	TIWAY Primitive 32
GA	Gather blocks	GATHER	TIWAY Primitive 51
GL	Get Primitive length	GETLEN	TIWAY Primitive 04 (Rel. 1.0)
NA	NIM native Task Code	NATIVE	TIWAY Primitive 01
RS	Read status	RDSTS	TIWAY Primitive 02
TG	Read data (host format)	TIGET	TIWAY Primitive 20
TP	Write data (host format)	TIPUT	TIWAY Primitive 30
TR	Read data (500/505 format)	TIREAD	TIWAY Primitive 20
TW	Write data (500/505 format)	TIWRIT	TIWAY Primitive 30
WB	Buffered write	WRBUF	TIWAY Primitive 33
WG	Write and gather block	WRTGAT	TIWAY Primitive 52
<b>Host Adapter Command Code Subroutines</b>			
AC	Activate Secondary	ACTVAT	Host adapter command code 04
AD	Get adapter diagnostics	RADIAG	Host adapter command code 08
BR	Broadcast message	BRDCST	Host adapter command code 02
DA	Deactivate Secondary	DEACT	Host adapter command code 05
PO	Poll response	POLL	Host adapter command code 03
SD	Secondary diagnostics	RSDIAG	Host adapter command code 07
SL	Secondary log	SECLOG	Host adapter command code 06
XP	General purpose transparency command	XPAR	User formatted I/O

## TIPROG Utility (continued)

Table 7-3 Alphabetized List of TIPROG Commands with Subroutines and Primitives (continued)

<b>TIPROG Command</b>	<b>Function</b>	<b>Equivalent Subroutine</b>	
<b>CIM Functional Command Subroutines</b>			
CC	CIM CCU status	CCUSTS	CIM Functional Command 69
CR	CIM read	CIMRD	CIM Functional Command 62
CW	CIM write	CIMWR	CIM Functional Command 63
RL	Read CIM loop data	RDLOOP	CIM Functional Command 6A
R1	CIM download random	RNDRD1	CIM Functional Command 6D
R2	CIM retrieve random	RNDRD2	CIM Functional Command 6D
R3	CIM define and gather	RNDRD3	CIM Functional Command 6D
R4	CIM write and gather	RNDRD4	CIM Functional Command 6D
<b>Status and Support Subroutines</b>			
BL	Build block masks	BLDMSK	Build mask for GATHER
GE	Return error message	GETMSG	Return error message string
HT	Convert host to 500/505 format	HST2TI	Data format conversion
LF	Look up format	LKUFMT	Look up format
LL	Look up tag (long format)	LKUTGL	Look up tag (long format)
LS	Look up tag (short format)	LKUTGS	Look up tag (short format)
PM	Display error message	PUTMSG	Error message display
TH	Convert 500/505 to host format	TI2HST	Data format conversion
<b>Miscellaneous</b>			
HE	Help Menu listing all commands available in TIPROG		
<RETURN>	Help Menu scrolling		
Q	Exit TIPROG		
<CTRL>Z	Exit command		

Figure 7-2 shows the TIPROG Help Menu screen as it appears when you type HE.

AC	ACTVAT	Activate Secondary	SD	SDIAG	Secondary diagnostics
AD	ADIAG	Get adapter diagnostics	SL	SECLOG	Secondary log
BR	BRDCST	Broadcast message	TG	TIGET	Read data (host format)
CH	CHNGST	Change Secondary state	TH	TI2HST	Convert TI to host format
CN	CONFIG	Get configuration	TP	TIPUT	Write data (host format)
DA	DEACT	Deactivate Secondary	TR	TIREAD	Read Data (TI format)
DE	DEFBLK	Define data blocks	WB	WRBUF	Buffered write
FI	FIN	Finish TIWAY routines	WG	WRTGAT	Write and gather blocks
FL	FILL	Fill block	TW	TIWRIT	Write data (TI format)
GA	GATHER	Gather blocks	XP	XPAR	Transparency command
GL	GETLEN	Get primitive length			
HT	HST2TI	Convert host to TI format	IN	INIT	Initialize TIWAY routines
CR	CIMRD	CIM read			
LF	LKUFMT	Lookup format	CW	CIMWR	CIM write
LL	LKUTGL	Lookup tag (long format)	R1	RNDRD1	CIM download random
LS	LKUTGS	Lookup tag (short format)	R2	RNDRD2	CIM retrieve random
NA	NATIVE	NIM native task code	R3	RNDRD3	CIM define and gather
PO	POLL	Poll response	R4	RNDRD4	CIM write and gather
RS	RDSTS	Read status	RL	RDLOOP	Read CIM loop data
	Q	Exit utility (Also ^Z)	HE	-	This message

Figure 7-2 TIPROG Help Menu

## TIPROG Utility (continued)

---

### Session Control Commands

The IN command (initialize) must be issued at the beginning of each TIPROG session. To end the session, issue the Q command.

**IN Command** When you first enter TIPROG, you must initialize the session by typing IN at the function prompt. If you attempt to execute any other command before you have initialized the session, you receive an error message, WRONG INITed STATE.

Prompt: TIPROG> IN  
Enter port setup string (e.g. – P1, 19200, 7, 1, E, 3, A) *portstr*

Response: TIPROG>

*portstr* – Used to select and initialize the appropriate communications port. This must be an ASCII string specifying the port number, baud rate, number of data bits, number of stop bits, parity number of retries, and length of time-out in hex. The following is an example of a port set up string:  
“P1, 19200, 7, 1, E, 3, A”

**FI Command** The FI command (finish) releases the system resources. It should be used to end a TIPROG session. You do not need to issue the FI command if your session has exited or aborted.

TIPROG> FI

TIPROG>

**Q Command** The Q command ends the TIPROG session.

Prompt: TIPROG> Q

Response: c:\TIWY6911>

**CTRL Z** The CTRL Z key sequence cancels the TIPROG operation in progress.

---

Host Adapter  
Commands

The Host Adapter commands are used to configure the network and to diagnose the operation of the network. These commands can be used with all Secondaries, but implementation is dependent on the the UHA mode and type of Host Adapter.

Base HIU  
Commands

**AC Command** The activate command is used to connect a Secondary or Secondaries to the specified highway, allowing communication between the host computer and the Secondary.

Prompt: TIPROG> AC  
List of Secondaries (free format hex)  
LIST: 2

Response: List of connected Secondaries 1 Secondaries connected  
02  
TIPROG>

or

Prompt: TIPROG> AC  
List of Secondaries (free format hex)  
LIST: 01020304

Response: List of connected Secondaries 4 Secondaries connected  
01 02 03 04  
TIPROG>

## TIPROG Utility (continued)

---

**AD Command** The adapter diagnostics command, the equivalent of the ADIAG subroutine, requests a Host Adapter diagnostics list. The statistics that are returned have been collected by the Host Adapter about its network usage. The AD command is the implementation of command code 08.

Prompt: TIPROG> AD

Response: Host Adapter diagnostics (decimal)

Invalid host commands	0
Send information commands	531
Broadcasts	0
Poll Secondaries	0
Connects	3
Disconnects	1
Read Secondary logs	1
Read Secondary diagnostics	2
Read adapter diagnostics	1
Reset adapters	0
Total poll cycles	0
Total frames transmitted	531
Timeouts	0
I-frames received	531
Retransmissions	0
Receiver errors	0
Transmit errors	0
Secondary initializations	5
Current clock (high/low)	0

TIPROG>

---

**BR Command** The broadcast command simultaneously sends a command buffer that you specify to all Secondaries on the highway you specify. Before using BR, be aware that the command sends a message, but does not guarantee or acknowledge that the message has been received. Review the cautions listed in the description of the BRDCST subroutine. The BR command is the implementation of command code 02.

At the CMD: prompt, you supply a command buffer that contains a TIWAY Primitive. The format is length of Primitive, Primitive code, and Primitive data. For more information refer to the XPAR subroutine, and to the Secondary user guides.

Use the PO command to solicit responses to the BR message.

Prompt: TIPROG> BR  
Command buffer (free format hex)  
CMD: 0006200100100001

Response: TIPROG>

**DA Command** The deactivate command is used to disconnect a Secondary from the specified highway, preventing any communication between the host computer and the Secondary. The DA command is the implementation of command code 05.

Prompt: TIPROG> DA  
List of Secondaries (free format hex)  
LIST: 2

Response: List of disconnected Secondaries 1 Secondaries disconnected  
02  
TIPROG>

or

Prompt: TIPROG>DA  
List of Secondaries (free format hex)  
LIST: 010203040506

Response: List of disconnected Secondaries 6 Secondaries disconnected  
01 02 03 04 05 06  
TIPROG>

## TIPROG Utility (continued)

---

**PO Command** The poll command solicits a response from a single specified Secondary to the most recent broadcast command that Secondary has received. The TH command is useful in processing the results of the poll command. The PO command is the implementation of command code 03.

Prompt: TIPROG> PO  
Tag specifier: #0102

Response: Response buffer: 38 bytes received  
03 02 00 22 20 00 AA BB BC DE CD EF 07 08 09 0A  
0B 0C 0D 0E 0F 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00  
TIPROG>

**SD Command** The Secondary diagnostics command, the equivalent of the RSDIAG subroutine, returns a Secondary diagnostics list, collected by the Host Adapter, that contains data about the Secondary's network usage and errors. The SD command is the implementation of command code 07.

Prompt: TIPROG> SD  
Tag specifier: #0102

Secondary diagnostics (decimal)

Number of polls	25
I-frames transmitted	5
Timeouts	0
I-frames received	5
I-frames re-transmitted	0
Received errors	0
Transmit errors	0
Secondary initialization	4

TIPROG>



---

**SL Command** The Secondary log command, the equivalent of the SECLOG subroutine, returns a list of the Secondaries logically connected to the specified highway when the request is issued. The SL command is the implementation of command code 06.

Prompt: TIPROG> SL

Response: List of connected Secondaries 1 Secondaries connected  
02  
TIPROG>

**XP Command** The XP command, the equivalent of the XPAR subroutine, provides the lowest level of access in TIPROG. It allows you to pass commands that are embedded in the appropriate host-UHA protocol but not available in the TIPROG utility as a string of bytes. In the example shown below, the UHA command code FF is used to reset the Host Adapter.

---

**NOTE:** No more than 128 characters may be entered for a single command buffer. If an odd number of characters are entered, the package inserts a 0 preceding the last character. For example, if FFE were entered, the package converts this entry to FF0E.

---

 <b>WARNING</b>
--

Resetting the Unilink can be dangerous in some installations.

Resetting the Unilink causes a loss of communication which could cause unpredictable controller behavior that could result in death and/or serious injury, or damage to equipment.

Do not use this routine without thoroughly understanding how it might impact operations directed by the controller.

The following example shows how to reset the Unilink from TIPROG.

---

**NOTE:** After issuing a reset to the Unilink Host Adapter, you must follow these steps:

1. Issue a dummy command (such as command code 21) to the Unilink Host Adapter using the BDLC host protocol, because the first response following a UHA reset returns.
  2. Reconfigure the Unilink Host Adapter.
  3. Reconnect all Secondary devices.
  4. Reallocate all Source IDs.
  5. Redefine all macros.
-

## TIPROG Utility (continued)

---

Prompt: TIPROG> XP  
Command buffer (free format hex)  
CMD: FF  
Error checking types  
0 – No checking  
1 – Check for Host Adapter Command Code errors  
2 – Check for NIM Primitive/Host Adapter errors  
4 – Check for CIM Functional Command/Host Adapter errors  
Error checking type (0,1,2,4): 1  
Response: Response buffer: 1 bytes received  
FF  
TIPROG>

### TIWAY Primitive Commands

The TIWAY Primitive commands determine and assess the performance of NIM-based attached devices on a network. You should refer to the associated user manuals for the attached devices for further information on the Primitive commands supported by the devices.

**CH Command** The change state command is used to change the operational state of a specified Secondary. The Secondary must be in a specific state before it is able to perform certain functions. The CH command shows you a list of Secondary states and a list of operational states. Fill in the code for the Secondary state you wish to change to, selecting from the codes in the Secondary state list. You are also shown a code for previous operational status; this code is taken from the operational status list. The CH command is the implementation of Primitive 10.

TIPROG> CH  
Tag specifier: #0102  
Secondary state (DD)      Operational status (HH)

00 – Execute logic and loops	00 – Executing logic and loops
01 – Execute loops only	01 – Executing w/non-fatal error
02 – No execution	02 – Executing loops only
	03 – No execution
	04 – Executing loops w/non-fatal error
	05 – No execution w/non-fatal error
	80 – Fatal error

DD – Secondary state (hex): 02  
Returned operational status (hex) 03

### **WARNING**

Remote state changes can be dangerous in some installations.  
Remote state changes could cause unpredictable controller behavior that could result in death and/or serious injury, or damage to equipment.  
Do not use this routine without thoroughly understanding how it might impact operations directed by the controller.

---

**CN Command** The configuration command displays the configuration (total memory sizes) of the specified Secondary. The CN command is the implementation of Primitive 03.

TIPROG> CN  
Tag specifier: #0102

Configuration parameters (decimal)

Device type (hex)	40
Instruction data memory size	16384
Variable data type memory size	26624
Constant data type memory size	0
Local input/output memory size	16
Global input/output memory size	2032

**DE Command** The define block command defines a data acquisition block and specifies data that are to be returned from the block. NIMs have 32 blocks that are definable by memory location and length. Using the DE command, independent blocks can be defined.

Note that the DE command only defines the block; to retrieve the data, use the GA command. The DE command is the implementation of Primitive 50.

TIPROG> DE  
Tag specifier: #0102  
Number of blocks (decimal): 5  
Arbitrary limit of 4 blocks in TIPROG  
Number of blocks (decimal): 2

Definition for first data acquisition block  
Block number (decimal): 32  
Tag specification: #0102000001  
Number of data elements (decimal): 20

Definition for second data acquisition block  
Block number (decimal): 1  
Tag specification: #0102010004  
Number of data elements (decimal): 2

## TIPROG Utility (continued)

---

**FL Command** The fill command can be used to fill a consecutive block of memory in a 5TI or PM550 attached device with a specified pattern.

To view the results of the fill command, use the TG command. The FL command is the implementation of Primitive 32.

```
TIPROG> FL
Tag specifier: #0101010001
Number of data elements (decimal): 10
Pattern (hex – four digits): C3C3
```

**GA Command** The gather command is used to gather data from the data acquisition blocks defined using the DE command.

The TH and LF commands are useful in processing the results of the gather command. The GA command is the implementation of Primitive 51.

```
TIPROG> GA
Tag specifier: #0102

(MSB is block 32, LSB is block 1)
Mask of data acquisition blocks (hex – eight digits): 8000000
Response buffer 40 bytes received
85 29 86 01 B0 01 00 01 01 F4 60 21 60 22 60 23
60 24 60 25 60 26 60 27 60 28 00 00 00 00 00 00
00 00 00 00 00 00 00 00
```

**GL Command** The get length command obtains and displays the maximum Primitive length and number of data acquisition blocks supported by the specified Secondary. The GL command is the implementation of Primitive 04 (Rel. 1.x).

```
TIPROG> GL
Tag specifier: #0102
```

Primitive format configuration	(decimal)
Maximum Primitive length	270
Number of data acquisition blocks	32

---

**NA Command** The NATIVE command allows you to bypass the normal Primitive processing and execute a native Task Code directly. Each machine has a machine-specific command set, including specific functions that are not mapped into the Primitives, and the NATIVE command gives you direct access to this command set.

---

**NOTE:** No more than 128 characters may be entered for a single command buffer.

---

The NA command is the implementation of Primitive 01.

```
TIPROG> NA
Tag specifier: #0102010002
Task Code command buffer (free format hex)
CMD: 38
Response buffer 2 bytes received
38 00
```

**RS Command** The read status command simply returns the status of the specified Secondary. The RS command is the implementation of Primitive 02.

```
TIPROG> RS
Tag specifier: #0102
Secondary status (hex)

Operational status:           00
Auxiliary power status:      80
NIM status:                   00
```

## TIPROG Utility (continued)

---

**TG Command** The TIGET command reads consecutive memory locations from a specific Secondary and converts the data from the 500/505 format. The TG command is an implementation of Primitive 20.

```
TIPROG> TG
Tag specifier: #0102010001
Number of data elements (decimal): 10
Data element      1 (decimal and hex)  1      0001
Data element      2 (decimal and hex)  2      0002
Data element      3 (decimal and hex)  3      0003
Data element      4 (decimal and hex)  4      0004
Data element      5 (decimal and hex)  5      0005
Data element      6 (decimal and hex)  6      0006
Data element      7 (decimal and hex)  7      0007
Data element      8 (decimal and hex)  8      0008
Data element      9 (decimal and hex)  9      0009
Data element     10 (decimal and hex) 10     000A
TIPROG>
```

**TP Command** The TIPUT command writes consecutive memory locations in a specified Secondary and converts the data from the host format to 500/505 format. The TP command is an implementation of Primitive 30.

```
TIPROG> TP
Tag specifier: #010201000B
Number of data elements (decimal): 2
Next 16 bit data element (decimal or hexH): C3C3H
Next 16 bit data element (decimal or hexH): 0001H
TIPROG>
```

**TR Command** The 500/505 read command reads consecutive memory locations from a specified Secondary. Data are returned in 500/505 format. The TR command is an implementation of Primitive 20.

```
TIPROG> TR
Tag specifier: #0102010001
Number of data elements (decimal): 10
Response buffer 20 bytes received
00 01 00 02 00 03 00 04 00 05 00 06 00 07 00 08
00 09 00 0A
TIPROG>
```

---

**TW Command** The 500/505 write command, the dual of TR, writes consecutive memory locations in a specified Secondary. Data is expected in 500/505 format.

---

**NOTE:** No more than 128 characters of data may be entered in a single TW command.

---

The TW command is an implementation of Primitive 30.

```
TIPROG> TW
Tag specifier: #010201000D
Number of data elements (decimal): 2
Data (free format hex)
DATA: 03 04 05 06
```

**WB Command** The write buffer command writes consecutive memory locations in Secondaries that support buffered L-memory. This command supports 5TI attached devices, and the data element locations specified must be contiguous. The WB command is the implementation of Primitive 33.

```
TIPROG> WB
Tag specifier: #0101
```

Command (CC)

00 – Clear storage RAM and start temporary storage of data  
01 – Continue storage of data in temporary RAM  
02 – Replace attached device memory with stored data  
03 – Abort temporary storage

```
CC – Command (hex): 00
Number of data elements (decimal): 4
Data (free format hex)
DATA: 0000 1111 2222 3333
Primitive status (hex) 00
```

**WG Command** The write and gather command combines the functions of the TIWRITE command and the gather command. It is particularly useful in data acquisition applications.

To use the WG command, you must first issue the DE command to define the blocks. The WG command is the implementation of Primitive 52.

```
TIPROG> WG
Tag specifier: #0102010020
Number of data elements (decimal): 4
Data (free format hex)
DATA: AAAA BBBB CCCC DDDD

(MSB is block 32, LSB is block 1)
Mask of data acquisition blocks (hex – eight digits): 00000001
Response buffer: 4 bytes received
00 04 00 05
```

### CIM Functional Commands

The CIM Functional Commands determine and assess the performance of CIM-based attached devices on a network (valid for PM550s only).

**CC Command** The CIM command control unit status command reads the status of the command control unit (valid for PM550s only).

```
TIPROG> CC
Tag specifier: #0102
CCU status (hex) 0050
```

**CR Command** The CIM read command reads consecutive image register locations from the specified Secondary. The first example shown below identifies the Secondary using an ASCII string; the second uses a tag name (located in the tag table). (Valid for PM550s only).

```
TIPROG> CR
Tag specifier: #0102010001
Number of data elements (decimal): 10
Response buffer: 20 bytes received
10 02 11 04 12 06 13 08 14 0A 00 00 00 00 00 00
00 00 00 00
```

```
TIPROG> CR
Tag specifier: SETPT
Number of data elements (decimal): 1
Response buffer: 2 bytes received
44 FE
```



---

**CW Command** The CIM write command writes to consecutive image register locations in the specified Secondary (valid for PM550s only).

```
TIPROG> CW
Tag specifier: #0102010001
Number of data elements (decimal): 5
Data element ( 1) (hex): 1
Data element ( 2) (hex): 2
Data element ( 3) (hex): 3
Data element ( 4) (hex): 4
Data element ( 5) (hex): 5
Data element ( 6) (hex): 6
Data element ( 7) (hex): 7
Data element ( 8) (hex): 8
Data element ( 9) (hex): 9
Data element (10) (hex): 10
```

**RL Command** The read loop command reads loop parameters. You are prompted for a loop number that determines the type of data returned (valid for PM550s only). Possible responses are:

```
01      display only
02      tuning only
03      display and tuning
```

Notice that the combination of all loops and display and tuning combined is not allowed.

The first example below uses an ASCII string to respond to the tag specifier prompt; the second uses a tag name.

```
TIPROG> RL
Tag specifier: #0102
Loop number (decimal): 1
Requested loop data (decimal): 33
10 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 04
26 40 00 00 00 00 00 00
```

```
TIPROG> RL
Tag specifier: SETPT
Loop number (decimal): 1
Requested loop data (decimal): 03
10 03 EC 80 00 00 00 00 00 01 00 00 00 00 00 04
26 40 00 00 00 00 00 00 00 00 00 00 04 11 00 00 00
00 00 00 04 33 E7 FD 80
```

## TIPROG Utility (continued)

---

**R1 Command** The R1 command, the equivalent of the RNDRD1 subroutine, defines a data acquisition block (valid for PM550s only).

The block can be activated to gather the specified data using the R2, R3, or R4 command.

TIPROG> R1

Tag specifier: #0102

Number of blocks (decimal): 3

Defining the first data acquisition block

Tag specifier: #0102010001

Number of data elements (decimal): 10

Defining the second data acquisition block

Tag specifier: #0102020001

Number of data elements (decimal): 10

Defining the third data acquisition block

Tag specifier: #0102010064

Number of data elements (decimal): 2

**R2 Command** The R2 command, the equivalent of the RNDRD2 subroutine, gathers a block of data or a group of blocks previously defined using R1 or R3. The buffer returned contains only the data specified by the blocks, and those data are in 500/505 format (valid for PM550s only).

TIPROG> R2

Tag specifier: #0102

(MSB is block 16, LSB is block 1)

Mask of data acquisition blocks (hex – four digits): 0005

01 02 03 04 05 06 07 08 09 0A 00 00 00 00 00 00

00 00 00 00 00 00 10 00

---

**R3 Command** The R3 command, the equivalent of the RNDRD3 subroutine, gathers data from a block of data acquisition blocks specified by a mask and defines new data acquisition blocks (valid for PM550s only).

TIPROG> R3

Tag specifier: #0102

Number of blocks (decimal): 3

Defining the first data acquisition block

Tag specifier: #0102010001

Number of data elements (decimal): 10

Defining the second data acquisition block

Tag specifier: #0102020001

Number of data elements (decimal): 10

Defining the third data acquisition block

Tag specifier: #0102010064

Number of data elements (decimal): 2

(MSB is block 16, LSB is block 1)

Mask of data acquisition blocks (hex – four digits): 0007

01 02 03 04 05 06 07 08 09 0A 00 00 00 00 00 00

00 00 00 00 E0 C8 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 10 00

## TIPROG Utility (continued)

---

**R4 Command** The R4 command, the equivalent of the RNDRD4 subroutine, performs two functions: it writes to a set of sequential memory locations and also gathers data from one or more data acquisition blocks specified by a mask (valid for PM550s only).

```
TIPROG> R4
Tag specifier: #0102010001
Number of data elements (decimal): 10
Data element ( 1) (hex): 1
Data element ( 2) (hex): 2
Data element ( 3) (hex): 3
Data element ( 4) (hex): 4
Data element ( 5) (hex): 5
Data element ( 6) (hex): 6
Data element ( 7) (hex): 7
Data element ( 8) (hex): 8
Data element ( 9) (hex): 9
Data element (10) (hex): 10
Data element (11) (hex): 11
Data element (12) (hex): 12
Data element (13) (hex): 13
Data element (14) (hex): 14
Data element (15) (hex): 15
Data element (16) (hex): 16
Data element (17) (hex): 17
Data element (18) (hex): 18
Data element (19) (hex): 19
Data element (20) (hex): 20

(MSB is block 16, LSB is block 1)
Mask of data acquisition blocks (hex – four digits): 0007
01 00 02 00 03 00 04 00 05 00 00 00 00 00 00 00
00 00 00 00 E0 C8 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 10 00
```

---

Support  
Commands

The support commands are general purpose commands that are useful in performing other commands.

**HT Command** The host-to-TI command, the complement of the TI-to-host command, converts data from the host format to the 500/505 format. The 500/505 format data are required to construct buffers for XP, NA, BR, GA, and WG.

```
TIPROG> HT
TT - data element type (hex): 20
Number of data elements (decimal): 3
Next real data element: 0.0
Next real data element: 99.999
Next real data element: -1.0
Response buffer (hex):
00 00 00 00 42 63 FF BE C1 10 00 00
```

**LF Command** The look-up format command returns information on format, returning all elements as integers.

The format can be decoded using the following list:

A = 01, 01, null, null ; bytes, no conversion  
B = 02, 02, null, null ; words, no conversion  
C = 02, 02, swap, swap ; integer, swap bytes  
D = 04, 04, vtff, vtff ; real\*4, TI-to-host format  
E = 03, 03, null, null ; triplets, no conversion  
F = 01, 01, rev8, rev8 ; # bytes, bits reversed in field  
G = 04, 04, vtff, vtff ; real, TI-to-host format  
H = 04, 04, null, null ; # bytes, no conversion  
I = 13, 13, vtff, vtff ; real, TI-to-host format

```
TIPROG> LF
TT - data element type (hex): 20
Format D 500/505 length 4 Host length 4
Equivalent CIM type (hex) 08
```

## TIPROG Utility (continued)

---

**LL Command** The look up tag (long format) command returns address information that describes the address specification passed to it. The address specification can be any one of the available types. The example given below uses an ASCII string.

```
TIPROG> LL
Tag specifier: #0102010001
Address list (in hex)
0000
0001
0002
0001
0001

Address elements (hex)

Highway number:          01
Secondary address:       02
Data element type:      01
Address offset:          0001
```

**LS Command** The look-up tag (short format) command returns address information that describes the address specification passed to it. The address specification can be any one of the available types. The example given below uses a tag name as the tag specifier.

```
TIPROG> LS
Tag specifier: MyTagName
Address list (in hex)
0000
0001
0002
0001
0003

Address elements (hex)

Highway number: 01
Secondary address: 02
Data element type: 01
Address offset: 0003
```

---

**TH Command** The TI-to-host command, the complement of the host-to-TI command, converts data from 500/505 format to host format. This command is particularly useful in decoding the data that are returned by XP, PO, NA, GA, and WG.

```
TIPROG> TH
TT - data element type (hex): 20
Number of data elements (decimal): 3
TI DATA 00 00 00 00 42 63 FF BE C1 10 00 00
Data element 1 (real) 0.0000 .0000000E+00
Data element 2 (real) 99.9990 .9999899E+02
Data element 3 (real) -1.0000 -.1000000E+01
```

# Chapter 8

## Tag Table

---

8.1	Defining Tag Names .....	8-2
8.2	Building the Tag Table .....	8-3



## 8.1 Defining Tag Names

---

As discussed in Section 3.5, Secondaries are addressed through an argument mechanism known as a tag, which can be specified either as a string, for instance, “#hhssttaaaa” or “/hhssttaaaaaaaa”, or as an array of integers. Either method can become cumbersome with repetitive use. Therefore, a means of referencing the same information symbolically is provided. This is called the tag name.

As an example, consider the following TIPROG prompt:

```
TIPROG> TG
Tag specifier: #0101010001FFFF
```

Continually entering this long tag when executing several successive commands could become tedious. However, using symbolic representation of this tag simplifies the operation. An example follows.

```
TIPROG> TG
Tag specifier: 565V
```

The operation is simplified.

There are no limitations on the number of tag names that can be defined. The collection of all tag name definitions is contained within a file called the tag table. The tag table filename is TAG.TBL. Each entry within the tag table is a unique tag name definition which takes the form:

```
nnnnnnnnnnnnnnnn hh ss tt aaaa aaaa ; comments
```

where nn...nn is the symbolic tag name  
hh is the highway  
ss is the Secondary address  
tt is the data type to be accessed (optional)  
aa... is the data address to be accessed (optional) (Hex)

Either the short form or the long form of the tag may be configured. Either extended or non-extended addressing may be used. The tag name may be up to 12 characters in length. Either upper- or lower-case hexadecimal characters may be used in specifying the remaining fields. The tag name may be separated from the “hh” field by an alphanumeric character. The remaining fields may be delimited by any character other than lower-case “a-f”. If extended addressing is chosen, the extra two bytes in the address field “aa...” must be used.

## 8.2 Building the Tag Table

Once constructed, the tag table must be converted from the editable text file version to a binary version that can be used by the TIWAY library or TIPROG and TIUSER interactive utilities. This is done by executing the tag table builder. This procedure can either be automatically run upon system startup by inserting the line `c:\TIWY6911\tag` into your AUTOEXEC.BAT file, or by executing the DOS command:

```
c:\TIWY6911\tag
```

Figure 8-1 shows a sample tag table.

```
{
{ Tag table definitions
{ NOTE: Fields in the tag table must be separated by tabs, not spaces!
{
000C   Length of a tag name
{
{-----
{ FILENAME: TAG.TBL
{ PRODUCT: TIWAY HOST SOFTWARE FOR PC
{
{ DESCRIPTION:
{ Tag table definitions. Up to 12 characters may be used in a tag name. The tag name field may be
{ separated from the rest of the fields by any non-hexadecimal character and the remaining
{ fields may be delimited by any non-hexadecimal character except lower-case a-f. Lower-case
{ a-f characters may be used in any hexadecimal field. The extra 2 byte address field must be
{ used for ALL extended addresses regardless of whether those tags are long or short.
{
{ Tagname HH SS TT AAAA AAAA comments
{ First tag is maximum length CCCCCCCC
{ Upper or lower case hexadecimal may be used
AAAAAAAAAAAA      01,02,03,00000004      First tag
BBBBBBBBBBBBBB   01,02,03,00000004
CCCCCCCCCCCC     01,02,03,0000ABCD      A new tag
DDDDD            01,02,03,00000001      And a little variety
MARK             01,02,01,00000002      A little vanity
MyTagName        01,02,01,00000003      An example
{ (hwy 1, sec2, v memory, address 1)
a                01,02,03,00000004      1 character name
mark             01,02,02,00000003      it is different from MARK
typef            01,01,06,00000001      packed x
typev            01,01,01,00000001      V memory
typea            01,01,03,00000001      discrete x
hwy0             00,01,01,00000001      bad highway number
hwy2             02,01,01,00000001      bad highway number
short            01,01                                Short tag
```

Figure 8-1 Sample Tag Table

# Network Autoconfiguration File

---

<b>9.1</b>	<b>Introduction</b> .....	<b>9-2</b>
<b>9.2</b>	<b>Network Autoconfigure File</b> .....	<b>9-3</b>
	Template Record .....	9-3
	Data Record .....	9-3
	Comments .....	9-7
	Record Editing .....	9-7
<b>9.3</b>	<b>Network Autoconfiguration Process</b> .....	<b>9-8</b>

The Network Autoconfigure File is a convention that permits you to start up the network without operator intervention. Each Unilink Host Adapter on the network(s) can be identified and configured as follows:

- If the UHA is selected as a Host Interface Unit (HIU), make required definitions and select its bandwidth.
- If the UHA is selected as a Network Manager (NM), make required definitions. For HIUs on the network, allocate NM buffers and select its preferred channel.
- If the UHA is selected as a Master Host Interface Unit (MHIU), make required definitions, allocate NM buffers, and select its bandwidth and preferred channel. Network autoconfiguration is achieved by:

Editing a text file that defines the network configuration.

Executing a program that reads this file and performs the necessary network definitions. The following paragraphs discuss this feature in greater detail.

## 9.2 Network Autoconfigure File

---

The Network Autoconfigure File is a text editable file that contains all information necessary to configure a Unilink Host Adapter (UHA) as a Host Interface Unit (HIU), a Network Manager (NM), or a Master Host Interface Unit (MHIU). This information is contained within file records. There are two types of records within the file:

- A template record, which contains information which aids in configuring a UHA.
- A data record, which contains the actual configuration data. The Network Autoconfigure File is found in the NETAUTOC.DAT file.

### Template Record

Several template records are used together to describe the UHA configuration data field that is edited within a data record. The template record describes the field to be edited, defines the valid range for the field entry, and delineates the field within the record. Each template record must start with an exclamation point in column 1. Figure 9-1 shows the format of this record.

!	field	field	field
!	description	description	description
!	(range)	(range)	(range)
!			
!	VV	VV	VVV

Figure 9-1 Network Autoconfiguration File Template Record Format

### Data Record

The data record contains an identifier field and data fields. The identifier field (columns 1–6) is used to select the specific configuration activity. The data fields are used to contain the configuration-specific data. The position of the data fields vary from record to record. The user is aided in placing field data in the correct columns by a template. The generic data record format is shown in Figure 9-2.

AAAAA	nn	nn	nnn
-------	----	----	-----

Figure 9-2 Network Autoconfiguration File Data Record Format

Data records must start with one of the following keywords: ALCHBW, is used primarily to assist you in ALCNMB, CFGHIU, CNFGHA, CNFGNM, END, EOF and HWY. Each keyword correlates to a specific data record type.

## Network Autoconfigure File (continued)

**HWY Record** The HWY record begins the definition for a particular UHA and contains a flag that determines whether the UHA should be reset or not. See Figure 9-3.

```

!   Host Adapter
!   Identifier Reset?
!   (1-32)    (Y/N)
!   ||        |
!   VV        V
!-----+-----+-----+-----+-----+-----+-----
-----+-----
HWY      nn      a

```

Figure 9-3 HWY Record Format

**CNFGHA Record** The CNFGHA data record configures the UHA, specifies the HIU address and a maximum secondary address, and allocates memory. See Figure 9-4.

```

!
! *****
! ***** UHA CONFIGURATION
! *****
!
! UHA Configuration:
!   0 = MHIU in
!       EHA mode
!   1 = HIU in
!       EHA mode      Maximum      Memory
!       2 = HIU      HIU Secondary  Allocation
!       3 = NM onlyaddressaddress  0 = 16K
!       4 = MHIU    (0-254)(1-254)  1 = 24K
!                                     2 = as much as possible
!       |           | |           | |           |
!       V           V V           V V           V
!-----+-----+-----+-----+-----+-----
-----+-----
CNFGHA  n           nnn      nnn           n

```

Figure 9-4 CNFGHA Record Format

**CFGHIU and ALCHBW Records** The CFGHIU and ALCHBW records are used to configure either a HIU or a MHIU. Figure 9-5 shows CFGHIU and ALCHBW records.

```

!
! *****
! ***** UHA CONFIGURATION
! *****
!
! ===== HIU Configuration : HIU
!
!   A = async
!   S = sync
!   F = full duplex
!   H = half duplex
!   0 = NRZ
!   1 = NRZI
!           Baud
!           Delay
!           Delay
!   A/S F/H  0/1  Rate      (0-1000) (0-1000)
!   |   |   |   |   |   |   |
!   V   V   V   V   V   V   V
!-----+-----+-----+-----+-----+-----+-----+-----+-----+
CFGHIU  a   a           n   nnnnnn           nnnn           nnnn
!
! ===== HIU Configuration : HIU bandwidth
!
!           max #           #
!           outstanding      outstanding
!           repetitive      requests per
!           macros          secondary
!           (0-254)         (0-254)
!           |   |           |   |
!           V   V           V   V
!-----+-----+-----+-----+-----+-----+-----+
ALCHBW  nnn           nnn

```

Figure 9-5 CFGHIU and ALCHBW Record Formats





---

**END Record** The END record (Figure 9-7) completes the definition for a particular UHA.

```
!
END
!
```

Figure 9-7 END Record Format

**EOF Record** The EOF record (Figure 9-8) marks the end of the Network Autoconfiguration File.

```
!
EOF
```

Figure 9-8 EOF Record Format

**Comments**

You can add comments to the Network Autoconfiguration File. This is achieved by placing a “!” in column 1 of the record. As with the template records, these records are treated as non-data records by the network auto-configuration process and ignored.

**Record Editing**

Any VMS Editor may be used to edit the Network Autoconfiguration File. Data records must have the identifier field in columns 1 – 6. Data field entries must be confined to the columns delineated by the template and must be within the specified range. Be careful not to disturb the template records. These records must contain a “!” in column 1. If comment records are added, a “!” must be placed in column 1. Any deviation from the above rules results in abnormal termination of the network auto-configuration process. Error messages are displayed to aid in identifying the error condition(s).

### 9.3 Network Autoconfiguration Process

---

Once edited, the Network Autoconfiguration File is ready for the actual configuration process. This process can be invoked two ways:

- Interactively
- From a command file

The interactive version is invoked by typing

```
$ RUN SYSSYSTEM:NETCONFIG
```

from DCL. If the process is to be executed from a command file, then the following line should be added:

```
$ RUN SYSSYSTEM:NETCONFIG
```

*Chapter 10*  
**Subroutine Library**

---

10.1	ACTVAT .....	10-4
10.2	ADIAG .....	10-5
10.3	BLDMSK .....	10-7
10.4	BRDCST .....	10-8
10.5	CCUSTS .....	10-10
10.6	CHNGST .....	10-11
10.7	CIMDNL .....	10-12
10.8	CIMRD .....	10-14
10.9	CIMUPL .....	10-15
10.10	CIMWR .....	10-16
10.11	CONFIG .....	10-17
10.12	DEACT .....	10-18
10.13	DEFBLK .....	10-20
10.14	DNLOAD .....	10-22
10.15	FILL .....	10-24
10.16	FIN .....	10-25
10.17	GATHER .....	10-26
10.18	GETLEN .....	10-27
10.19	GETMSG .....	10-28
10.20	HST2TI .....	10-29
10.21	INIT .....	10-30
10.22	LKUFMT .....	10-31
10.23	LKUTGL .....	10-32
10.24	LKUTGS .....	10-33

---

10.25	NATIVE	10-34
10.26	POLL	10-35
10.27	PUTMSG	10-36
10.28	RDLOOP	10-37
10.29	RDSTS	10-38
10.30	RNDRD1	10-40
10.31	RNDRD2	10-41
10.32	RNDRD3	10-42
10.33	RNDRD4	10-44
10.34	SDIAG	10-45
10.35	SECLOG	10-46
10.36	TI2HST	10-47
10.37	TIGET	10-48
10.38	TIPUT	10-50
10.39	TIREAD	10-51
10.40	TIWRIT	10-52
10.41	TIXTN and TIXTNW	10-53
10.42	UPLOAD	10-54
10.43	WRBUF	10-56
10.44	WRTGAT	10-58
10.45	XPAR	10-60

---

This chapter lists the syntax and argument detail for each subroutine available in the TIWAY Host Software Package. The subroutines are listed in alphabetical order. The overview information on using these subroutines is presented in Chapters 4 and 5.

---

**NOTE:** Chapter 3 describes all calling arguments used in the TIWAY Interface Subroutines (the subroutines contained in Chapter 4). However, the TIWAY Support Routines (described in Chapter 5) use arguments that do not occur in the Interface Subroutines. Therefore, the arguments used in the Support Subroutines are defined in the Notes on Call Format section in this Chapter.

---

## 10.1 ACTVAT

---

Before any communication is allowed with a Secondary, the Secondary must be logically connected to the network, which is done using the ACTVAT subroutine. After the Secondary has been connected, the host computer can communicate freely with it. If a Secondary is connected to the highway when the call is issued, it can be included in *wi4*, and is listed as successfully connected. ACTVAT is the implementation of command code 04.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("ACTVAT", *istat*, *xtn*, *hwy*, *wilen*, *wi4*, *rilen*, *ri4*)  
CALL ACTVAT (*istat*, *xtn*, *hwy*, *wilen*, *wi4*, *rilen*, *ri4*)

Pascal: TIWAY ("ACTVAT", *istat*, *xtn*, *hwy*, *wilen*, *wi4*, *rilen*, *ri4*);  
ACTVAT (*istat*, *xtn*, *hwy*, *wilen*, *wi4*, *rilen*, *ri4*);

C: *tiway* ("*actvat*", &*istat*, &*xtn*, &*hwy*, &*wilen*, *wi4*, &*rilen*, *ri4*);  
*actvat* (&*istat*, &*xtn*, &*hwy*, &*wilen*, *wi4*, &*rilen*, *ri4*);

### Notes on Call Format

Explanations of the terms used in the call format follow.

*istat* – The status is zero (success) even if none of the Secondaries is connected. To verify that Secondaries were connected, the application program should be designed to inspect *rilen*.

*xtn* – Specifies synchronous (*xtn*=0) or asynchronous (*xtn*=1) completion.

*hwy* – Specifies the logical highway number which is used to communicate to the device setup for communications. For the PC version of the TIWAY package, *hwy* is always set to 1.

*wilen* – The *wilen* argument specifies the number of Secondary(s) to be connected to the highway.

*wi4* – The *wi4* argument specifies the list of Secondary(s) to be connected to the highway.

*rilen* – The *rilen* argument is returned to the application program to indicate the number of entries in *ri4* (the number of Secondaries upon which a connection was performed).

*ri4* – The *ri4* argument is returned to the application program containing the list of Secondaries upon which a connection was attempted. A valid Secondary address indicates a successful connection. A value of zero indicates that the Secondary was not connected.

### Related Calls

The DEACT subroutine is used to disconnect Secondaries from the network.

## 10.2 ADIAG

---

The ADIAG subroutine returns a list of statistics (diagnostics) collected by the Host Adapter about the Host Adapter's network usage and network activity statistics. ADIAG is the implementation of command code 08.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("ADIAG", istat, xtn, hwy, rilen, ri32)  
CALL ADIAG (istat, xtn, hwy, rilen, ri32)

Pascal: TIWAY ("ADIAG", istat, xtn, hwy, rilen, ri32);  
ADIAG (istat, xtn, hwy, rilen, ri32);

C: tiway ("adiag", &istat, &xtn, &hwy, &rilen, ri32);  
adiag (&istat, &xtn, &hwy, &rilen, ri32);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- hwy – Specifies the logical highway number which is used to communicate to the device setup for communications. For the PC version of the TIWAY package, hwy is always set to 1.
- rilen – The number of elements returned within ri32.

---

ri32 – This integer array is used to return the Host Adapter diagnostics list in host format. Response values follow:

[aaaa] = number of invalid commands from host

[bbbb] = number of send information commands processed

[cccc] = number of broadcast commands processed

[dddd] = number of poll Secondary commands

[eeee] = number of connect Secondary commands processed

[ffff] = number of disconnect Secondary commands processed

[gggg] = number of read Secondary log commands processed

[hhhh] = number of read Secondary diagnostic commands processed

[iiii] = number of read adapter diagnostic commands processed

[jjjj] = number of reset adapter commands processed

[kkkk] = total number of poll cycles

[llll] = total number of I-frames transmitted to Secondaries

[mmmm] = total number of timeouts

[nnnn] = total number of I-frames received from Secondaries

[oooo] = total number of I-frames re-transmitted (null for UHA)

[pppp] = total number of receive errors

[qqqq] = total number of transmit errors (null for UHA)

[rrrr] = total number of Secondary initializations

[sssssss] = current system clock value in 256 microsecond intervals

Rel. 1.0 TIWAY Host Adapters return a reserved field (set to zero) in elements ssssss. The system clock for these models is returned in the two elements immediately following the ssssss field.



## 10.3 BLDMSK

---

The BLDMSK subroutine builds a mask from a list of block numbers.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("BLDMSK", mask, list)  
FORTRAN: CALL BLDMSK (mask, list)

Pascal: TIWAY ("BLDMSK", mask, list);  
BLDMSK (mask, list);

C: tiway ("bldmsk", mask, list);  
bldmsk (&mask, list);

### Notes on Call Format

Explanations of the terms used in the call format follow.

mask – The mask contains the bit mask taken from the list argument.  
Declaration: integer\*4 (FORTRAN); packed array [1..32] of Boolean (Pascal).

list – The list argument is a 32-word integer array. Each element of the array contains either a valid block number (in the range of 1 to 32) or a zero. For each valid block number, the corresponding bit in the mask argument is set. Declaration: integer (FORTRAN); array [1..32] of integer (Pascal).

### Example

```
initialize LIST to 0
      .
      .
      .
LIST(1)=3           ! include block 3
LIST(2)=5           ! include block 5
LIST(3)=11          ! include block 11
CALL BLDMSK (MASK, LIST) ! build mask
CALL GATHER (ISTAT, 0, TAG, MASK, RSPLN, RSP275)
                        ! Use it
```

## 10.4 BRDCST

---

The BRDCST subroutine broadcasts the user-specified command buffer to all Secondaries on the specified highway. The command buffer must be one that all the Secondaries can interpret. Using BRDCST lowers the overhead associated with long command buffers that are issued to several Secondaries. BRDCST is the implementation of command code 02.

---

**NOTE:** The Secondaries do not immediately respond to the broadcast; the individual responses must be solicited using POLL.

PM550 CIMs (PPX:PM550–502 and PPX:PM550–503) do not support the BRDCST subroutine and ignore all broadcast messages. PM550 NIMs (PPX:PM550–5038, PPX:PM550–5039, and PPX:PM550–5040) support BRDCST.

The BRDCST routine is intended to be immediately followed by the POLL routine. Calling another TIWAY routine after BRDCST and before POLL causes an invalid response (i.e., the Secondary attempts to answer the new routine as if it were responding to the POLL request).

When using BRDCAST and POLL through a UHA, refer to the *SIMATIC Unilink Host Adapter User Manual*, PPX:TIWAY–8121–x, for the implementation of these commands for your specific UHA mode.

---

### Call Format

BASIC and FORTRAN: CALL TIWAY ("BRDCST", istat, xtn, hwy, cmdlen, cmd275)  
CALL BRDCST (istat, xtn, hwy, cmdlen, cmd275)

Pascal: TIWAY ("BRDCST", istat, xtn, hwy, cmdlen, cmd275);  
BRDCST (istat, xtn, hwy, cmdlen, cmd275);

C: tiway ("brdcst", &istat, &xtn, &hwy, &cmdlen, cmd275);  
brdcst (&istat, &xtn, &hwy, &cmdlen, cmd275);

---

**Notes on Call Format**

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- hwy – Specifies the logical highway number which is used to communicate to the device setup for communications. For the PC version of the TIWAY package, hwy is always set to 1.
- cmdlen – The cmdlen argument specifies the length of the command buffer.
- cmd275 – The cmd275 argument contains a command consisting of the following fields: Primitive length, Primitive code, and Primitive data, if applicable. The command is formatted for the Host Adapter to read. For more information, refer to the *SIMATIC Unilink Host Adapter User Manual*, PPX:TIWAY-8121-x.

**Related Calls**

The POLL subroutine must be used to obtain responses to BRDCST.

## 10.5 CCUSTS

---

The CCUSTS subroutine reads the attached PM550's Command Control Unit's (CCU) status. CCUSTS is the implementation of CIM functional command 69.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("CCUSTS", istat, xtn, tag, retsts)  
FORTRAN: CALL CCUSTS (istat, xtn, tag, retsts)

Pascal: TIWAY ("CCUSTS", istat, xtn, tag, retsts);  
CCUSTS (istat, xtn, tag, retsts);

C: tiway ("ccusts", &istat, &xtn, &tag, &retsts);  
ccusts (&istat, &xtn, &tag, &retsts);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- tag – The short form of the address (highway and Secondary) can be used.
- retsts – The CCU status is returned in this argument in the form 00XY,

where X is the state of the CCU:

- 2 = start up
- 3 = hold
- 5 = run
- 6 = remote

and Y is the current CCU mode:

- 0 = not program
- 1 = program

## 10.6 CHNGST

---

The CHNGST subroutine is used to change the operational state of a specified NIM-based attached device. CHNGST is the implementation of Primitive 10. This changes the operational state of a controller, either from STOP to RUN or vice versa, and must be selected carefully. See warning below.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("CHNGST", istat, xtn, tag, state, sstat)  
FORTRAN: CALL CHNGST (istat, xtn, tag, state, sstat)

Pascal: TIWAY ("CHNGST", istat, xtn, tag, state, sstat);  
CHNGST (istat, xtn, tag, state, sstat);

C: tiway ("chngst", &istat, &xtn, &tag, &state, &sstat);  
chngst (&istat, &xtn, &tag, &state, &sstat);

### Notes on Call Format

Explanations of the terms used in the call format follow.

istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).

xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.

tag – The short form of the address (highway and Secondary) can be used.

state – The state argument (an integer argument in which the high byte of the integer is ignored) is the code for the desired state. Each NIM allows a specific subset; refer to the appropriate NIM user manuals. Generally, those states are:

00 = execute logic and loops  
01 = execute loops only  
02 = no execution

sstat – The Secondary status argument contains the returned state of the specified attached device for Series 500/505 NIMs, Release 1.1 and earlier. For later releases, sstat contains the current state.

### WARNING

Remote state changes can be hazardous in some installations.

Remote state changes could cause unpredictable operation that could result in death or serious injury to personnel, and/or damage to equipment.

Do not use this routine without thoroughly understanding how it might impact equipment controlled by the programmable logic controller.

## 10.7 CIMDNL

---

The CIMDNL subroutine downloads instructions or data into a PM550. Refer to the *CIM User's Manual* for more information on the procedures necessary to do this. CIMDNL is the implementation of CIM functional command 65.

**Call Format**

BASIC and FORTRAN: CALL TIWAY ("CIMDNL", istat, xtn, tag, ss, wblen, wb275, cstat, mm)  
CALL CIMDNL (istat, xtn, tag, ss, wblen, wb275, cstat, mm)

Pascal: TIWAY ("CIMDNL", istat, xtn, tag, ss, wblen, wb275, cstat, mm);  
CIMDNL (istat, xtn, tag, ss, wblen, wb275, cstat, mm);

C: tiway ("cimdnl", &istat, &xtn, &tag, &ss, &wblen, &wb275, &cstat, &mm);  
cimdnl (&istat, &xtn, &tag, &ss, &wblen, &wb275, &cstat, &mm);

### Notes on Call Format

Explanations of the terms used in the call format follow.

istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).

xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.

tag – The short form of the address (highway and Secondary) can be used.

ss – The ss argument is a code used to describe the operation. Valid values (in hex) are:

<u>ss</u>	<u>function</u>
00	initial block
01	continue
02	terminate
08	request download

On return, ss contains the code describing the operation that was performed. It contains an 02 if the operation was automatically terminated.

wblen – The length of the write buffer. (Note that wblen must equal zero when download – ss = 08 – is requested.)

wb275 – The wb275 argument contains the instructions or data to be downloaded.

---

cstat – Contains the CCU status code on return. Possible values (in hex) are

<u>cstat</u>	<u>status</u>
01	CCU not in start-up
02	C- and L-memory in ROM
04	extended C-memory present
08	extended L-memory present

mm – The data element type argument defines the type of PM550 memory:

<u>mm</u>	<u>memory type</u>
01	L-memory
02	V-memory
04	C-memory

Note that mm is cumulative (i.e., mm = 3 defines L- and V-memory; mm = 7 defines L-, V-, and C-memory).

#### Related Calls

The TIUSER DOWNLOAD function (Chapter 6) can be used instead of writing programs to download Secondary memory.

---

**NOTE:** The PM550 must be in Startup or Remote mode in order for the download operation to be enabled. If the PM550 is in any other mode, a facility 8 message 67 (hex) Invalid Download Sequence error is returned.

---

The CIMRD subroutine reads consecutive memory locations from the specified CIM-based attached device.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("CIMRD", istat, xtn, tag, nnnn, rsplen, rsp275)  
CALL CIMRD (istat, xtn, tag, nnnn, rsplen, rsp275)

Pascal: TIWAY ("CIMRD", istat, xtn, tag, nnnn, rsplen, rsp275);  
CIMRD (istat, xtn, tag, nnnn, rsplen, rsp275);

C: tiway ("cimrd", &istat, &xtn, &tag, &nnnn, &rsplen, &rsp275);  
cimrd (&istat, &xtn, &tag, &nnnn, &rsplen, &rsp275);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- tag – The short form of the address (highway and Secondary) can be used.
- nnnn – The nnnn argument specifies the number of data element locations to be filled. Each data element is two bytes long.
- rsplen – The response buffer length argument specifies a maximum length for the data to be returned (and the actual length on return in bytes).
- rsp275 – The rsp275 argument contains the gathered data in 500/505 format.

### Related Calls

The CIMWR subroutine can be used to write the same set of locations.

The RNDRDx series of subroutines can be used to gather multiple consecutive sets of the same data accessible through CIMRD.



## 10.9 CIMUPL

---

The CIMUPL subroutine uploads one or more memory areas from a PM550. Refer to the *CIM User Manual* for more information on the procedures.

**Call Format**

BASIC and FORTRAN: CALL TIWAY ("CIMUPL", istat, xtn, tag, ss, rsplen, rsp275, cstat, mm)  
FORTRAN: CALL CIMUPL (istat, xtn, tag, ss, rsplen, rsp275, cstat, mm)

Pascal: TIWAY ("CIMUPL", istat, xtn, tag, ss, rsplen, rsp275, cstat, mm);  
CIMUPL (istat, xtn, tag, ss, rsplen, rsp275, cstat, mm);

C: tiway ("cimupl", &istat, &xtn, &tag, &ss, &rsplen, &rsp275, &cstat, &mm);  
cimupl (&istat, &xtn, &tag, &ss, &rsplen, &rsp275, &cstat, &mm);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11.
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- tag – The short form of the address (highway and Secondary) can be used.
- ss – The ss argument is a code used to describe the operation. Valid values (in hex) are:
- | <u>ss</u> | <u>function</u> |
|-----------|-----------------|
| 00        | initial request |
| 01        | continue        |
| 02        | terminate       |

On return, ss contains the code describing the operation that was performed. It contains an 02 if the operation was automatically terminated.

- rsplen – The response buffer length argument specifies a maximum length for the data to be returned (and the actual returned length in bytes).
- rsp275 – The rsp275 argument contains the gathered data in 500/505 format.
- cstat – Contains the CCU status code on return. Possible values (in hex):
- | <u>cstat</u> | <u>status</u>             |
|--------------|---------------------------|
| 01           | CCU not in start-up       |
| 02           | C- and L-memory in ROM    |
| 04           | extended C-memory present |
| 08           | extended L-memory present |
- mm – The data element type argument defines the type of PM550 memory.
- | <u>mm</u> | <u>memory type</u> |
|-----------|--------------------|
| 01        | L-memory           |
| 02        | V-memory           |
| 04        | C-memory           |

Note that mm is cumulative (i.e., mm = 3 defines L- and V-memory; mm = 7 defines L-, V-, and C-memory).

### Related Calls

The TIUSER UPLOAD function can be used instead of writing programs to upload Secondaries.

## 10.10 CIMWR

---

The CIMWR subroutine writes to consecutive memory locations in the specified CIM-based attached device.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("CIMWR", istat, xtn, tag, nnnn, wb275)  
CALL CIMWR (istat, xtn, tag, nnnn, wb275)

Pascal: TIWAY ("CIMWR", istat, xtn, tag, nnnn, wb275);  
CIMWR (istat, xtn, tag, nnnn, wb275);

C: tiway ("cimwr", &istat, &xtn, &tag, &nnnn, &wb275);  
cimwr (&istat, &xtn, &tag, &nnnn, &wb275);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- tag – The short form of the address (highway and Secondary) can be used.
- nnnn – The nnnn argument specifies the number of data element locations to be filled. Each data element is two bytes long.
- wb275 – The write buffer contains the data to be written. The data should be in the 500/505 format.

### Related Calls

The CIMRD subroutine can be used to read from the same set of locations.

The RNDRD4 subroutine can be used to write the same data simultaneously with the gathering of a data acquisition block.

## 10.11 CONFIG

The CONFIG subroutine returns the configuration (memory sizes) of the specified NIM-based attached device. CONFIG is the implementation of Primitive 03.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("CONFIG", istat, xtn, tag, rilen, ri32, sstat)

FORTRAN: CALL CONFIG (istat, xtn, tag, rilen, ri32, sstat)

Pascal: TIWAY ("CONFIG", istat, xtn, tag, rilen, ri32, sstat);  
CONFIG (istat, xtn, tag, rilen, ri32, sstat);

C: tiway ("config", &istat, &xtn, &tag, &rilen, ri32, &sstat);  
config (&istat, &xtn, &tag, &rilen, ri32, &sstat);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11.
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- tag – The short form of the address (highway and Secondary) can be used.
- rilen – The number of values returned in ri32.
- ri32 – ri32 contains the attached device configuration data and is an integer array of words returned in host format. Values are listed below.

<u>ri32[x]</u>	<u>value</u>	<u>device type</u>	
[1]	0000	5TI	
	0020	520	
	002C	520C/525	} Series 500 Family
	0030	530	
	003C	530C/525	
	0060	560	
	0065	565	
	007E	Unilink Host Adapter	
	0080	PM550	
	0100	Unilink Secondary Adapter	
	0200	IT-111/IT-121 Tank Transmitters	
	0204	IT-150 Hydrostatic Tank Transmitter	
	0208	IT-160 Micro Remote Control Unit	

[2]integer instruction data type memory size

[3]integer variable data type memory size

[4]integer constant data type memory size

[5]integer local input/output memory size

[6]integer global input/output memory size

[7]\*integer total user memory — high order

[8]\*integer total user memory — low order

\*Series 500/505 NIM, Rel. 2.0 and later, and Unilink only.

- sstat – The Secondary-status argument contains the code for the current state of a specific NIM-based attached device. Current state means the state the device is in when the response to the subroutine call is returned. Current state is returned for all attached devices except Series 500 NIMs, Release 1.1 and earlier. (For these Series 500 NIMs, the state that is returned is the state that existed before the subroutine call was issued.) Table 3-7 shows the seven possible values.

## 10.12 DEACT

---

The DEACT subroutine logically disconnects one or more Secondaries from the specified highway. Disconnecting a Secondary removes that Secondary's entry from the host adapter's Secondary log, preventing any communications between that Secondary and the Host Adapter until the Secondary is again connected to the Host Adapter with the ACTVAT subroutine. DEACT is the implementation of command code 05. The DEACT command results in the issuing of an HDLC DISC to the Secondary depending on the configuration of the UHA.

---

**NOTE:** A Secondary is successfully disconnected only if it is connected and functioning properly when DEACT is issued.

When a NIM is disconnected, it resets (the test mode is entered and the NIM executes its startup sequence). A Series 500 NIM attached to a 560 or 565 has its memory configuration table updated at this time.

---

### Call Format

BASIC and FORTRAN: CALL TIWAY ("DEACT", istat, xtn, hwy, wilen, wi4, rilen, ri4)  
CALL DEACT (istat, xtn, hwy, wilen, wi4, rilen, ri4)

Pascal: TIWAY ("DEACT", istat, xtn, hwy, wilen, wi4, rilen, ri4);  
DEACT (istat, xtn, hwy, wilen, wi4, rilen, ri4);

C: tiway ("deact", &istat, &xtn, &hwy, &wilen, wi4, &rilen, ri4);  
deact (&istat, &xtn, &hwy, &wilen, wi4, &rilen, ri4);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – The status from DEACT is zero (success) even if none of the Secondaries is disconnected. To verify that Secondaries were disconnected, the application program should be designed to inspect rilen.
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- hwy – Specifies the logical highway number which is used to communicate to the device setup for communications. For the PC version of the TIWAY package, hwy is always set to 1.
- wilen – The wilen argument specifies the number of Secondary(s) to be disconnected from the highway.
- wi4 – The wi4 argument specifies the list of Secondary(s) to be disconnected from the highway.

- 
- ri4n – The ri4n argument is returned to the application program to indicate the number of entries in ri4 (the number of Secondaries upon which a disconnect was performed).
  - ri4 – The ri4 argument is returned to the application program to indicate the list of Secondaries upon which a disconnect has been performed. A valid Secondary address indicates that the disconnect was successful. A zero value indicates that it was not.

**Related Calls**

The ACTVAT subroutine is used to connect Secondaries to the network.

## 10.13 DEFBLK

---

The DEFBLK subroutine is used to define PLC memory locations. See the applicable NIM user manuals for a complete description of the data acquisition Primitives. DEFBLK is the implementation of Primitive 50.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("DEFBLK", istat, xtn, tag, nblk, cclst, taglst, nnnn1st, sstat)  
FORTRAN: CALL DEFBLK (istat, xtn, tag, nblk, cclst, taglst, nnnn1st, sstat)

Pascal: TIWAY ("DEFBLK", istat, xtn, tag, nblk, cclst, taglst, nnnn1st, sstat);  
DEFBLK (istat, xtn, tag, nblk, cclst, taglst, nnnn1st, sstat);

C: tiway ("defblk", &istat, &xtn, &tag, &nblk, cclst, taglst, nnnn1st, &sstat);  
defblk (&istat, &xtn, &tag, &nblk, cclst, taglst, nnnn1st, &sstat);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- tag – The short form of the address (highway and Secondary) can be used.
- nblk – The number of blocks (nblk) argument specifies how many data acquisition blocks are to be defined. The argument is also used in RDRAND and WRRAND to specify the number of data groups to be transferred.
- cclst – An array containing block numbers for each block to be defined.
- taglst – The taglst argument is an array of addresses of the tags that identify the data element type and starting data element location for each of the blocks defined. The array contains an address for each of the blocks defined (nblk elements). Each array element must contain a full (long) tag specification.
- nnnn1st – The nnnn1st argument is a list of bytes. Each array element specifies the number of data elements to be included in a particular block. The array is nblk elements long.

---

sstat – The Secondary-status argument contains the code for the current state of a specific NIM-based attached device. Current state means the state the device is in when the response to the subroutine call is returned. Current state is returned for all attached devices except Series 500 NIMs, Release 1.1 and earlier. (For these Series 500 NIMs, the state that is returned is the state that existed before the subroutine call was issued.) Table 3-7 shows the seven possible values.

**Related Calls**

The GATHER subroutine can be used to read the data defined by DEFBLK.

## 10.14 DNLOAD

---

The DNLOAD subroutine is used to download a previously uploaded file to a Secondary.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("DNLOAD", istat, tag, ftype, flen, fnam)  
CALL DNLOAD (istat, tag, ftype, flen, fnam)

Pascal: TIWAY ("DNLOAD", istat, tag, ftype, flen, fnam);  
DNLOAD (istat, tag, ftype, flen, fnam);

C: tiway ("dnload", &istat, &tag, &ftype, &flen, &fnam);  
dnload (&istat, &tag, &ftype, &flen, &fnam);

---

**NOTE:** For BASIC programmers: Because BASIC implements dynamic strings, force an extra character, preferably a blank, at the end of the string variable containing the tag.

### Example:

```
TAG$="      " (six blanks)
Input TAG$
or
TAG$="#0102 "
```

This step is required because the TIWAY library tests the sixth character in a tag to determine length.

---

### Notes on Call Format

Explanations of the terms used in the call format follow.

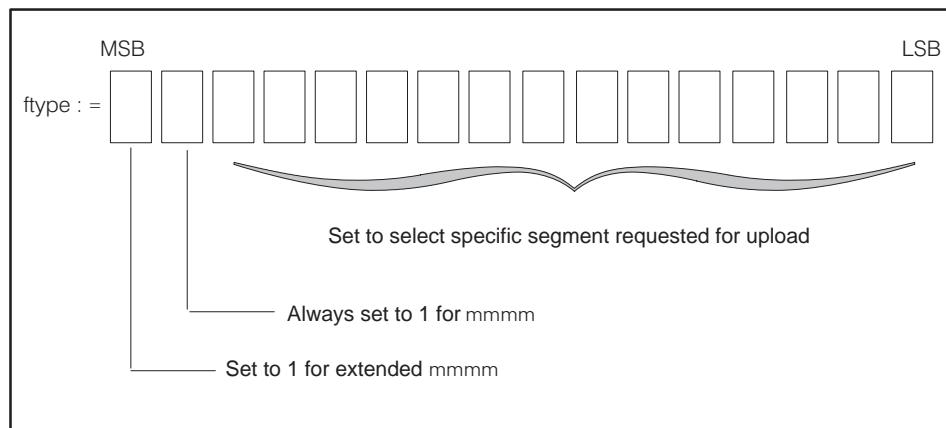
- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- tag – The short form of the tag (highway and address) can be used. Since DNLOAD is written in FORTRAN, the tag argument must be passed by descriptor. See the programming manual for the language you are using for information on passing arguments by descriptor.



---

**f<sub>type</sub>** – Specifies what type of download is requested according to its value.  
Declaration: integer.

- 20 = generic download of all memory
- 21 = generic download of program memory
- 22 = generic download of data memory
- 90 = download all memory except WXY, IR, WF
- 91 = partial download L-memory
- 92 = partial download V-memory
- 93 = partial download Constant memory
- 94 = partial download S-memory – loop tables
- 95 = partial download S-memory – analog alarms
- 96 = partial download S-memory – SF programs
- 97 = partial download S-memory – SF subroutines
- 98 = partial download IR, WF memory
- 99 = partial download WXY memory
- >4000 = Unilink specific segment download, which is translated to mmmm field in primitive where



---

**NOTE:** For f<sub>type</sub> >4000, the passed f<sub>type</sub> value is masked with the mask value in the upload file. For download selection of a specific segment to occur, the bits must match. This implies partial segment download is supported.

---

**f<sub>len</sub>** – Length of file specification (filename). Declaration: integer.

**f<sub>nam</sub>** – File specification (filename) for uploaded data. Declaration: string or character array.

## 10.15 FILL

---

The FILL subroutine is used to fill a number of contiguous data elements starting at a specified address. FILL is the implementation of Primitive 32.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("FILL", istat, xtn, tag, nnnn, pattern, sstat)  
CALL FILL (istat, xtn, tag, nnnn, pattern, sstat)

Pascal: TIWAY ("FILL", istat, xtn, tag, nnnn, pattern, sstat);  
FILL (istat, xtn, tag, nnnn, pattern, sstat);

C: tiway ("fill", &istat, &xtn, &tag, &nnnn, &pattern, &sstat);  
fill (&istat, &xtn, &tag, &nnnn, &pattern, &sstat);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- tag – The long form form of the address (highway and Secondary) must be used.
- nnnn – The nnnn argument specifies the number of data element locations to be filled. Each data element is two bytes long.
- pattern – The pattern argument is an integer field that fills a consecutive block of memory on a specified NIM-based attached device. The pattern argument is always a word in host format, which is converted to 500/505 format before being sent to the attached device. The valid range for pattern is 0 to 65535.
- sstat – The Secondary-status argument contains the code for the current state of a specific NIM-based attached device. Current state means the state the device is in when the response to the subroutine call is returned. Current state is returned for all attached devices except Series 500 NIMs, Release 1.1 and earlier. (For these Series 500 NIMs, the state that is returned is the state that existed before the subroutine call was issued.) Table 3-7 shows the seven possible values.

## 10.16 FIN

---

The FIN subroutine is the complement of the INIT subroutine. It releases system resources reserved during the INIT call. After FIN has been called, none of the other subroutines can be called until an INIT is called.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("FIN", istat, xtn)  
CALL FIN (istat, xtn)

Pascal: TIWAY ("FIN", istat, xtn);  
FIN (istat, xtn);

C: tiway ("fin", &istat, &xtn);  
fin (&istat, &xtn);

### Notes on Call Format

Explanations of some of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – The transaction number must be zero. (There is no valid second half entry point for any subroutine that does not actually do I/O to the Host Adapter.)

## 10.17 GATHER

---

The GATHER subroutine is used to gather data from previously defined data acquisition blocks (see the DEFBLK subroutine, Chapter 4). The GATHER subroutine returns the data in the 500/505 data format. Notice that data from multiple blocks can be returned in the same call, so multiple data formats may be returned in the same buffer. GATHER is the implementation of Primitive 51.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("GATHER", istat, xtn, tag, mask, rsplen, rsp275, sstat)  
CALL GATHER (istat, xtn, tag, mask, rsplen, rsp275, sstat)

Pascal: TIWAY ("GATHER", istat, xtn, tag, mask, rsplen, rsp275, sstat);  
GATHER (istat, xtn, tag, mask, rsplen, rsp275, sstat);

C: tiway ("gather", &istat, &xtn, &tag, &mask, &rsplen, &rsp275, &sstat);  
gather (&istat, &xtn, &tag, &mask, &rsplen, &rsp275, &sstat);

### Notes on Call Format

Explanations of the terms used in the call format follow.

istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).

xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.

tag – The short form of the address (highway and Secondary) can be used.

mask – See Section 3.9 of this manual for an explanation of masks.

rsplen – The response buffer length argument specifies a maximum length for the data to be returned (and the actual length on return in bytes).

rsp275 – The rsp275 argument contains the gathered data in 500/505 format.

sstat – The Secondary-status argument contains the code for the current state of a specific NIM-based attached device. Current state means the state the device is in when the response to the subroutine call is returned. Current state is returned for all attached devices except Series 500 NIMs, Release 1.1 and earlier. (For these Series 500 NIMs, the state that is returned is the state that existed before the subroutine call was issued.) Table 3-7 shows the seven possible values.

### Related Calls

The DEFBLK subroutine is required to define the data acquisition blocks. The data format subroutines, HST2TI and TI2HST, (see Chapter 5) can be called to perform data format conversions. The BLDMSK subroutine can be used to build a mask in non-Pascal applications. The WRTGAT subroutine combines TIWRIT and GATHER into a single call.

## 10.18 GETLEN

---

The GETLEN subroutine obtains and displays the maximum Primitive length and the maximum number of data acquisition blocks supported by the specified Secondary Primitive 04.

---

**NOTE:** GETLEN and GETCON are the only TIWAY Primitive routines that do not have a Secondary status (sstat) argument.

---

### Call Format

BASIC and FORTRAN: CALL TIWAY ("GETLEN", istat, xtn, tag, primlen, nblks)  
CALL GETLEN (istat, xtn, tag, primlen, nblks)

Pascal: TIWAY ("GETLEN", istat, xtn, tag, primlen, nblks);  
GETLEN (istat, xtn, tag, primlen, nblks);

C: tiway ("getlen", &istat, &xtn, &tag, &primlen, &nblks);  
getlen (&istat, &xtn, &tag, &primlen, &nblks);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- tag – The short form of the address (highway and Secondary) can be used.
- primlen – The Primitive length argument contains the maximum primitive length in bytes (integer in host format).
- nblks – The number of blocks available argument contains the maximum number of data acquisition blocks supported by the specified NIM-based attached device (integer in host format).

### Related Calls

The GETCON command performs the same Primitive and returns additional information for Series 500/505, Rel. 2.1 and later NIMs.

## 10.19 GETMSG

---

The GETMSG subroutine returns an error message to the caller in an 128-character string. GETMSG is called with the composite status (returned in the istat field) obtained from a previous call to a TIWAY Subroutine.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("GETMSG", istat, rsplen, str128)  
CALL GETMSG (istat, rsplen, str128)

Pascal: TIWAY ("GETMSG", istat, rsplen, str128);  
GETMSG (istat, rsplen, str128);

C: tiway ("getmsg", &istat, &rsplen, str128);  
getmsg (&istat, &rsplen, str128);

### Notes on Call Format

Explanations of the terms used in the call format follow.

istat – The composite status from a previous call is passed to the GETMSG subroutine. Declaration: integer.

rsplen – GETMSG is called with the response length buffer containing the maximum length of the str128 argument. On return, the actual string length is passed back in rsplen. Declaration: integer (in bytes).

str128 – GETMSG is called with an 128-character string initialized to blanks. This string is returned with the error message text. Declaration: string128.

## 10.20 HST2TI

---

The HST2TI subroutine converts data from host format to 500/505 format. The subroutine can convert from one buffer into another, or the same buffer can be specified for both source and destination.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("HST2TI", istat, xtn, tt, nnnn, hstbuf, tibuf)  
FORTRAN: CALL HST2TI (istat, xtn, tt, nnnn, hstbuf, tibuf)  
Pascal: TIWAY ("HST2TI", istat, xtn, tt, nnnn, hstbuf, tibuf);  
HST2TI (istat, xtn, tt, nnnn, hstbuf, tibuf);  
C: tiway ("hst2ti", &istat, &xtn, &tt, &nnnn, &hstbuf, tibuf);  
hst2ti (&istat, &xtn, &tt, &nnnn, &hstbuf, tibuf);

### Notes on Call Format

Explanations of the terms used in the call format follow.

istat – Error codes are returned in the status field. Declaration: integer.

xtn – The transaction number must be zero. Declaration: integer.

tt – The read only tt argument is the data element type (only low byte significant). Declaration: integer.

nnnn – The read/write nnnn argument is the number of data elements to convert. Declaration: integer.

hstbuf – The read only host buffer argument is the source buffer (host format). Declaration: byte array (FORTRAN); packed array [1..275] of byte (Pascal).

tibuf – The write only 500/505 buffer argument is the destination buffer (500/505 format). Declaration: byte array (FORTRAN); packed array [1..275] of byte (Pascal).

Notice that no buffer lengths are passed. They are implied from the tt and nnnn arguments.

### Related Calls

The HST2TI subroutine is the reverse of the TI2HST subroutine.

## 10.21 INIT

---

INIT reserves system resources used by the software package.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("INIT", istat, xtn, portstr)  
Pascal: TIWAY ("INIT", istat, xtn, portstr);  
INIT (istat, xtn, portstr);  
C: tiway ("init", &istat, &xtn, &portstr);  
init (&istat, &xtn, &portstr);

### Notes on Call Format

Explanations of the terms used in the call format follow.

xtn – The transaction number must be zero. (There is no valid second half entry point for any subroutine that does not actually access the Host Adapter.)

portstr – Used to select and initialize the appropriate communications port. This must be an ASCII string specifying the port number, baud rate, number of data bits, number of stop bits, parity number of retries, and length of time-out in hex. The following is an example of a port set up string:

"P1, 19200, 7, 1, E, 3, A"

This string specifies port 1, 19200 baud; 7 data bits and 1 stop bit (standard for the host adapter); even parity, 3 retries, and a timeout of 10 (hex A) seconds per try.

The maximum timeout period is 15 seconds. The port set-up string must be fully specified in order for the TIWAY host package to communicate with the network.

### Related Calls

The INIT subroutine must have already been called to use all other subroutines in the TIWAY Host Software Package. Before the portstr parameter can be changed, the FIN subroutine must be called.



The LKUFMT subroutine returns information on data element format.

**Call Format**

BASIC and FORTRAN: CALL TIWAY ("LKUFMT", istat, xtn, tt, fmt, tilen, hostlen, cimtyp)  
CALL LKUFMT (istat, xtn, tt, fmt, tilen, hostlen, cimtyp)

Pascal: TIWAY ("LKUFMT", istat, xtn, tt, fmt, tilen, hostlen, cimtyp);  
LKUFMT (istat, xtn, tt, fmt, tilen, hostlen, cimtyp);

C: tiway ("lkufmt", &istat, &xtn, &tt, &fmt, &tilen, &hostlen, &cimtyp);  
lkufmt (&istat, &xtn, &tt, &fmt, &tilen, &hostlen, &cimtyp);

**Notes on Call Format**

Explanations of the terms used in the call format follow.

istat – Error codes are returned in the status field. Declaration: integer.

xtn – The transaction number must be zero. Declaration: integer.

tt – The tt argument, supplied by the caller, is the data element type specified in NIM format. All other arguments contain data corresponding to this data element type. Declaration: integer.

fmt – The format argument is a code used to indicate the format the data are represented in (and therefore the type of conversion required to convert data from the 500/505 format to the host format, and vice versa). Declaration: character. Formats follow:

A	bytes, no conversion
B	16-bit words, no conversion
C	16-bit words, converted to least significant bit first
D	floating point, conversion
E	triplets, 24-bit data, no conversion
F	bytes, converted to least significant bit first
G	floating point, 32-bit data, conversion
H	32-bit data, no conversion
I	floating point, 52-bit data, conversion
J	byte, converted to 32-bit integer

Declaration: character.

tilen – The 500/505 length argument specifies the length of the data element type in the 500/505 format in bytes. Declaration: integer.

hostlen – The host length argument specifies the length of the data element type in the host format in bytes. Declaration: integer.

cimtyp – The CIM type argument is an integer representing the corresponding CIM data specifier. A cimtyp of zero indicates there is no corresponding CIM data type. If the cimtyp is positive, it specifies the bit representing one of the CIM memory types. If negative, it is the two's complement (negative) of the bit representing one of the CIM image registers (X, Y, CR). Declaration: integer.

## 10.23 LKUTGL

---

The LKUTGL subroutine returns address information that describes the address specification passed to it. The address specification can be any of the available types (tag name, ASCII, or binary). LKUTGL looks up a tag name or derives the address information from the address specifier.

LKUTGS and LKUTGL are identical in function; they differ in providing different granularity of address information and are distinguished by the number of arguments passed to them.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("LKUTGL", istat, xtn, tag, hwy, secadd, tt, aaaa)  
FORTRAN: CALL LKUTGL (istat, xtn, tag, addlst)

Pascal: TIWAY ("LKUTGL", istat, xtn, tag, addlst);  
LKUTGL (istat, xtn, tag, addlst);

C: tiway ("lkutgl", &istat, &xtn, &tag, addlst);  
lkutgl (&istat, &xtn, &tag, addlst);

### Notes on Call Format

Explanations of the terms used in the call format follow. For Declaration Types, see Table 3-2 and Table 3-3.

- istat – Error codes are returned in the status field.
- xtn – The transaction number must be zero.
- tag – The tag argument specifies the address list that is to be returned to the caller. This argument may take any of the available forms. The tag is converted to an address list either by searching the tag table, by converting the specification directly, or by a combination of the two. See Declaration for tag\_type in Figure 3-1.
- hwy – The hwy argument contains a highway number, which is an index into a list of consecutively numbered logical units that are mapped by the task. The PC version of the software only supports one highway; therefore, hwy = 1.
- secadd – The secadd argument is the TIWAY Secondary address, set by NIM dip switches, for the specified Secondary.
- tt – The tt argument is the data element type in the controller.
- aaaa – The aaaa argument is the data element location in the controller.

## 10.24 LKUTGS

---

The LKUTGS subroutine returns address information that describes the address specification passed to it. The address specification can be any of the available types (tag name, ASCII, or binary). LKUTGS looks up a tag name or derives the address information from the address specifier.

LKUTGS and LKUTGL are identical in function; they differ in providing different granularity of address information and are distinguished by the number of arguments passed to them.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("LKUTGS", istat, xtn, tag, addlst)  
CALL LKUTGS (istat, xtn, tag, addlst)

Pascal: TIWAY ("LKUTGS", istat, xtn, tag, addlst);  
LKUTGS (istat, xtn, tag, addlst);

C: tiway ("lkutgs", &istat, &xtn, &tag, addlst);  
lkutgs (&istat, &xtn, &tag, addlst);

### Notes on Call Format

Explanations of the terms used in the call format follow. For Declaration Types, see Table 3-2 and Table 3-3.

- istat – Error codes are returned in the status field.
- xtn – The transaction number must be zero.
- tag – The tag argument specifies the address list that is to be returned to the caller. This argument may take any of the available forms. The tag is converted to an address list either by searching the tag table, by converting the specification directly, or by a combination of the two. See Declaration for tag\_type in Figure 3-1.
- addlst – The addlst argument contains a binary specification. This can be used to convert a tag name or other non-direct specification to a binary format. The addlst argument also contains the lead-in null character to identify it as a binary specification.

## 10.25 NATIVE

---

The NATIVE subroutine allows you to bypass the normal Primitive processing and execute a native Task Code directly. Primitives are high-level commands that are not specific to any PLC model. Instead, NIMs use the Task Code commands that are native to the attached PLC to perform the function requested by the primitive.

The NATIVE subroutine accepts a Task Code command buffer, appends it to the Host Adapter Command Code and TIWAY Primitive length and Primitive 01, and passes it to the Secondary. The response, without Host Adapter and TIWAY Primitive overhead, is returned in the response buffer. This is similar to the XPAR subroutine, except that instead of the NITP protocol being added to the command and stripped from the response, the NITP protocol, Host Adapter Command Code 01, and TIWAY Primitive 01 are added to the command and stripped from the response. See the documentation for individual PLCs for information concerning the Task Codes they support.

---

**NOTE:** Only experienced programmers should use the NATIVE subroutine.

---

### Call Format

BASIC and FORTRAN: CALL TIWAY ("NATIVE", istat, xtn, tag, cmdlen, cmd275, rsplen, rsp275, sstat)  
CALL NATIVE (istat, xtn, tag, cmdlen, cmd275, rsplen, rsp275, sstat)

Pascal: TIWAY ("NATIVE", istat, xtn, tag, cmdlen, cmd275, rsplen, rsp275, sstat);  
NATIVE (istat, xtn, tag, cmdlen, cmd275, rsplen, rsp275, sstat);

C: tiway ("native", &istat, &xtn, &tag, &cmdlen, cmd275, &rsplen, &rsp275, &sstat);  
native (&istat, &xtn, &tag, &cmdlen, cmd275, &rsplen, &rsp275, &sstat);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- tag – The short form of the address (highway and Secondary) can be used.
- cmdlen – The cmdlen argument specifies the length of the command buffer.
- cmd275 – The cmd275 argument contains a Task Code command buffer.
- rsplen – The response buffer length argument specifies a maximum length for the response to the NATIVE request and the actual length on return in bytes.
- rsp275 – The rsp275 argument contains a valid response to the Task Code, with Task Code arguments.

## 10.26 POLL

---

The POLL subroutine polls the specified Secondary for the response to the broadcast it received most recently. POLL solicits a response only from the single specified Secondary. POLL is the implementation of command code 03.

---

**NOTE:** The POLL routine is intended to immediately follow the BRDCAST routine. Calling another TIWAY routine after BRDCAST and before POLL causes an invalid response (i.e., the Secondary attempts to answer the new routine as if it were responding to the POLL request).

When using BRDCAST and POLL through a UHA, refer to the *SIMATIC Unilink Host Adapter User Manual*, PPX:TIWAY-8121-x, for the implementation of these commands for your specific UHA mode.

---

### Call Format

BASIC and FORTRAN: CALL TIWAY ("POLL", istat, xtn, tag, rsplen, rsp275)  
CALL POLL (istat, xtn, tag, rsplen, rsp275)

Pascal: TIWAY ("POLL", istat, xtn, tag, rsplen, rsp275);  
POLL (istat, xtn, tag, rsplen, rsp275);

C: tiway ("poll", &istat, &xtn, &tag, &rsplen, &rsp275);  
poll (&istat, &xtn, &tag, &rsplen, &rsp275);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- tag – The short form of the address (highway and Secondary) can be used.
- rsplen – The rsplen argument must specify the maximum length of the response buffer (response buffer size) when the POLL subroutine is called. It contains the actual length of the response on return. The response buffer for POLL contains the entire response to the Host Adapter Command Code, including the command code identifier.
- rsp275 – The rsp275 argument contains the response from the Secondary. The contents of this response depends on the command sent by BRDCST.

### Related Calls

The POLL subroutine must be used after the BRDCST subroutine.

## 10.27 PUTMSG

---

The PUTMSG subroutine takes one argument, the composite status as returned from a previous call to a TIWAY Subroutine. It prints a message containing the facility code and message code and a text (if one exists for that status) on the operator's console.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("PUTMSG", istat)  
CALL PUTMSG (istat)

Pascal: TIWAY ("PUTMSG", istat);  
PUTMSG (istat);

C: tiway ("putmsg", &istat);  
putmsg (&istat);

### Notes on Call Format

Explanations of the terms used in the call format follow.

istat – The status argument is the composite status returned previously by a TIWAY Subroutine. Declaration: integer.

## 10.28 RDLOOP

---

The RDLOOP subroutine reads loop parameters from a PM550.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("RDLOOP", istat, xtn, tag, rtype, rsplen, rsp275)  
CALL RDLOOP (istat, xtn, tag, rtype, rsplen, rsp275)  
Pascal: TIWAY ("RDLOOP", istat, xtn, tag, rtype, rsplen, rsp275);  
RDLOOP (istat, xtn, tag, rtype, rsplen, rsp275);  
C: tiway ("rdloop", &istat, &xtn, &tag, &rtype, &rsplen, &rsp275);  
rdloop (&istat, &xtn, &tag, &rtype, &rsplen, &rsp275);

### Notes on Call Format

Explanations of the terms used in the call format follow.

istat – The status argument is the composite status returned previously by a TIWAY Subroutine. Declaration: integer.

xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.

tag – The short form of the address (highway and Secondary) can be used.

rtype – The rtype argument specifies the type of data to be returned. Valid values are:

<u>rtype</u>	<u>function</u>
01	display only
02	tuning only
03	display and tuning

Notice that the combination of all loops, display, and tuning are not allowed.

rsplen – The rsplen argument must specify the maximum length of the response buffer (response buffer size) when the POLL subroutine is called. It contains the actual length of the response on return. The response buffer for POLL contains the entire response to the Host Adapter Command Code, including the command code identifier.

rsp275 – The rsp275 argument contains the response from the Secondary. The contents of this response depends on the command sent by BRDCST.

### Related Calls

TI2HST may be useful in converting response buffers into real variables in the host format.

## 10.29 RDSTS

---

The RDSTS subroutine reads the status of the specified Secondary and attached device. RDSTS is the implementation of Primitive 02.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("RDSTS", istat, xtn, tag, rilen, ri32, sstat)  
CALL RDSTS (istat, xtn, tag, rilen, ri32, sstat)

Pascal: TIWAY ("RDSTS", istat, xtn, tag, rilen, ri32, sstat);  
RDSTS (istat, xtn, tag, rilen, ri32, sstat);

C: tiway ("rdsts", &istat, &xtn, &tag, &rilen, ri32, &sstat);  
rdsts (&istat, &xtn, &tag, &rilen, ri32, &sstat);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- tag – The short form of the address (highway and Secondary) can be used.
- rilen – The read buffer length as defined by the specified Secondary; usually 3.
- ri32\* – The Secondary and attached device status is returned in ri32.
- sstat\* – The attached device Operational Status is returned in sstat as usual.

### Related Calls

The RDSTS subroutine returns status information that can be affected by the CHNGST subroutine.

\* Not available for all Secondaries.



---

Possible Values:

<b><u>Symbol</u></b>	<b><u>Value</u></b>	<b><u>Device Definition</u></b>
sstat		Attached Device Operational Status (Mode):
	00	Operational and performing instruction data type and loop execution (RUN)
	01	Operational and performing instruction and data type and loop execution with a non-fatal error detected (RUN with non-fatal error)
	02	Operational and not performing instruction data type execution with loop execution (PROGRAM)
	03	Operational and not performing instruction data type or loop execution (START or REMOTE)
	04	Operational and not performing instruction data type execution with loop execution and a non-fatal error detected (PROGRAM with non-fatal error)
	05	Operational and not performing instruction and data type or loop execution and a non-fatal error is detected (START or REMOTE with non-fatal error)
	80	Not operational due to a fatal error condition (FATAL ERROR)
ri32[1]		Attached Device Auxiliary Power Source Status:
	00	Auxiliary power source good
	01	Auxiliary power source not available
	80	Auxiliary power source bad (see Secondary user manuals for more information.)
ri32[2]		Secondary Operational Status:
	00	Operational
	01	Channel A is not functional
	02	Channel B is not functional
ri32[3]		Secondary local/remote status
	00	Secondary in remote mode
	01	Secondary in local mode

---

**NOTE:** Not available for all Secondaries.

---

## 10.30 RNRDR1

---

The RNRDR1 subroutine defines a data acquisition block in a PM550. The data acquisition block can then be activated to gather the specified data using an RNRDR2, RNRDR3, or RNRDR4 subroutine.

---

**NOTE:** Unlike the NIM DEFBLK subroutine, any new definition starts with the first block, and all blocks defined after the first are numbered sequentially from that number.

---

### Call Format

BASIC and FORTRAN: CALL TIWAY ("RNRDR1", istat, xtn, tag, nblk, taglst, nnlst)  
FORTRAN: CALL RNRDR1 (istat, xtn, tag, nblk, taglst, nnlst)

Pascal: TIWAY ("RNRDR1", istat, xtn, tag, nblk, taglst, nnlst);  
RNRDR1 (istat, xtn, tag, nblk, taglst, nnlst);

C: tiway ("rnrdr1", &istat, &xtn, &tag, &nblk, &taglst, &nnlst);  
rnrdr1 (&istat, &xtn, &tag, &nblk, &taglst, &nnlst);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- tag – The short form of the address (highway and Secondary) can be used.
- nblk – The number of blocks (nblk) argument specifies how many data acquisition blocks are to be defined. The argument is also used in RDRAND and WRRAND to specify the number of data groups to be transferred.
- taglst – The taglst argument is an array of addresses of the tags that identify the data element type and starting data element location for each of the blocks defined. The array contains an address for each of the blocks defined (nblk elements). Each array element must contain a full (long) tag specification.
- nnlst – The nnlst argument is an array of bytes. Each array element specifies the number of data elements to be included in a particular block. The array is nblk elements long.

### Related Calls

A call to RNRDR2 following RNRDR1 gathers the blocks of data that were set up in the RNRDR1 subroutine.

A call to RNRDR3 following RNRDR1 gathers the blocks of data and can be used to define new blocks.

A call to RNRDR4 following RNRDR1 gathers the blocks of data and can be used to write to a set of sequential memory locations.

## 10.31 RNDRD2

---

The RNDRD2 subroutine gathers a block or group of blocks previously defined using either a RNDRD1 or RNDRD3 subroutine. The buffer returned contains only the data specified by the blocks, and those data are in the 500/505 format.

---

**NOTE:** Unlike the NIM DEFBLK subroutine, any new definition starts with the first block, and all blocks defined after the first are numbered sequentially from that number.

---

### Call Format

BASIC and FORTRAN: CALL TIWAY ("RNDRD2", istat, xtn, tag, cmask, rsplen, rsp275)  
CALL RNDRD2 (istat, xtn, tag, cmask, rsplen, rsp275)

Pascal: TIWAY ("RNDRD2", istat, xtn, tag, cmask, rsplen, rsp275);  
RNDRD2 (istat, xtn, tag, cmask, rsplen, rsp275);

C: tiway ("rndrd2", &istat, &xtn, &tag, &cmask, &rsplen, &rsp275);  
rndrd2 (&istat, &xtn, &tag, &cmask, &rsplen, &rsp275);

### Notes on Call Format

Explanations of the terms used in the call format follow.

istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).

xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.

tag – The short form of the address (highway and Secondary) can be used.

cmask – A mask, 16 bit spaces long, specifying which of 16 blocks may be accessed. The MSB specifies block 16 while the LSB specifies bit 1. For example:

```
cmask=810A(hex)
      1000000100000101 (binary)
would select blocks 1, 3, 9, and 16
```

rsplen – The response buffer length argument defines the maximum length of the response buffer; when it is returned, it contains the actual length in bytes of the response. See discussion of this argument for TIGET for an explanation on how it is computed.

rsp275 –The rsp275 argument contains the returned data in the 500/505 format.

### Related Calls

The blocks gathered with the RNDRD2 subroutine can be defined using the RNDRD1 or RNDRD3 subroutine.

The BLDMSK subroutine can be used to build a mask.

## 10.32 RNRD3

---

The RNRD3 subroutine gathers data from a block of data acquisition blocks specified by a mask and defines new data acquisition blocks.

---

**NOTE:** Unlike the NIM DEFBLK subroutine, any new definition starts with the first block, and all blocks defined after the first are numbered sequentially from that number.

---

### Call Format

**BASIC and FORTRAN:** CALL TIWAY ("RNRD3", istat, xtn, tag, cmask, rsplen, rsp275, nblk, taglst, nnlst)  
CALL RNRD3 (istat, xtn, tag, cmask, rsplen, rsp275, nblk, taglst, nnlst)

**Pascal:** TIWAY ("RNRD3", istat, xtn, tag, cmask, rsplen, rsp275, nblk, taglst, nnlst);  
RNRD3 (istat, xtn, tag, cmask, rsplen, rsp275, nblk, taglst, nnlst);

**C:** tiway ("rnrdd3", &istat, &xtn, &tag, &cmask, &rsplen, &rsp275, &nblk &taglst, &nnlst);  
rnrdd3 (&istat, &xtn, &tag, &cmask, &rsplen, &rsp275, &nblk, &taglst, &nnlst);

### Notes on Call Format

Explanations of the terms used in the call format follow.

**istat** – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).

**xtn** – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.

**tag** – The short form of the address (highway and Secondary) can be used.

**cmask** – A mask, 16 bit spaces long, specifying which of 16 blocks may be accessed. The MSB specifies block 16 while the LSB specifies bit 1. For example:

```
cmask=810A(hex)
      1000000100000101 (binary)
would select blocks 1, 3, 9, and 16
```

**rsplen** – The response buffer length argument defines the maximum length of the response buffer; when it is returned, it contains the actual length in bytes of the response. See discussion of this argument for TIGET for an explanation on how it is computed.

**rsp275** –The rsp275 argument contains the returned data in the 500/505 format.

- 
- nblk** – The number of blocks (**nblk**) argument specifies how many data acquisition blocks are to be defined. The argument is also used in **RDRAND** and **WRRAND** to specify the number of data groups to be transferred.
  - taglst** – The **taglst** argument is an array of addresses of the tags that identify the data element type and starting data element location for each of the blocks defined. The array contains an address for each of the blocks defined (**nblk** elements). Each array element must contain a full (long) tag specification.
  - nnlst** – The **nnlst** argument is an array of bytes. Each array element specifies the number of data elements to be included in a particular block. The array is **nblk** elements long.

## 10.33 RNRDR4

---

The RNRDR4 subroutine performs two functions in the same call and same host adapter transaction. It writes to a set of sequential memory locations (see the CIMWR subroutine for more detail) and also gathers data from one or more data acquisition blocks specified by a mask.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("RNRDR4", istat, xtn, tag, nnnn, wb275, cmask, rsplen, rsp275)  
FORTRAN: CALL RNRDR4 (istat, xtn, tag, nnnn, wb275, cmask, rsplen, rsp275)

Pascal: TIWAY ("RNRDR4", istat, xtn, tag, nnnn, wb275, cmask, rsplen, rsp275);  
RNRDR4 (istat, xtn, tag, nnnn, wb275, cmask, rsplen, rsp275);

C: tiway ("rnrdr4", &istat, &xtn, &tag, &nnnn, &wb275, &cmask, &rsplen, &rsp275);  
rnrdr4 (&istat, &xtn, &tag, &nnnn, &wb275, &cmask, &rsplen, &rsp275);

### Notes on Call Format

Explanations of the terms used in the call format follow.

istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).

xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.

tag – The short form of the address (highway and Secondary) can be used.

nnnn – The read/write nnnn argument is the number of data elements to convert. Declaration: integer.

wb275 – The write buffer contains the data to be written. The data should be in the 500/505 format.

cmask – A mask, 16 bit spaces long, specifying which of 16 blocks may be accessed. The MSB specifies block 16 while the LSB specifies bit 1. For example:

```
cmask=810A(hex)
      1000000100000101 (binary)
would select blocks 1, 3, 9, and 16
```

rsplen – The response buffer length argument defines the maximum length of the response buffer; when it is returned, it contains the actual length in bytes of the response. See discussion of this argument for TIGET for an explanation on how it is computed.

rsp275 – The rsp275 argument contains the returned data in the 500/505 format.

### Related Calls

The blocks gathered with the RNRDR2 subroutine can be defined using the RNRDR1 or RNRDR3 subroutine.

## 10.34 SDIAG

---

The SDIAG subroutine returns a list of statistics (diagnostics) collected by the Host Adapter about a specified Secondary. The statistics contain data about that Secondary's network usage and errors. SDIAG is the implementation of command code 07.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("SDIAG", istat, xtn, tag, rilen, ri32)  
FORTRAN: CALL SDIAG (istat, xtn, tag, rilen, ri32)

Pascal: TIWAY ("SDIAG", istat, xtn, tag, rilen, ri32);  
SDIAG (istat, xtn, tag, rilen, ri32);

C: tiway ("sdiag", &istat, &xtn, &tag, &rilen, ri32);  
sdiag (&istat, &xtn, &tag, &rilen, ri32);

### Notes on Call Format

Explanations of the terms used in the call format follow.

istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).

xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.

tag – The short form of the address (highway and Secondary) can be used.

rilen – The rilen argument contains the length of ri32.

ri32 – The read buffer is specified in integers as an array of words. The array is returned in host format. Values are:

[1] = Secondary address

[2] = number of times polled\*

[3] = number of I-frames transmitted\*

[4] = number of time-outs\*

[5] = number of I-frames received\*

[6] = number of I-frames re-transmitted\*

[7] = number of received errors from Secondary\*

[8] = number of transmitted errors to Secondary\*

[9] = number of times Secondary initialized\*

\*since last Host Adapter reset

## 10.35 SECLOG

---

The SECLOG subroutine returns a list of all Secondaries currently connected to the specified highway. SECLOG is the implementation of command code 06.

---

**NOTE:** The list returned from SECLOG represents only the existence of an entry in the Host Adapter Secondary log table for that Secondary, not that the Secondary is currently available. Secondaries not disconnected (DEACT routine) remains in this list even if the Secondary is physically disconnected or powered down.

---

### Call Format

BASIC and FORTRAN: CALL TIWAY ("SECLOG", istat, xtn, hwy, rilen, ri254)  
CALL SECLOG (istat, xtn, hwy, rilen, ri254)

Pascal: TIWAY ("SECLOG", istat, xtn, hwy, rilen, ri254);  
SECLOG (istat, xtn, hwy, rilen, ri254);

C: tiway ("seclog", &istat, &xtn, &hwy, &rilen, ri254);  
seclog (&istat, &xtn, &hwy, &rilen, ri254);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- hwy – Specifies the logical highway number which is used to communicate to the device setup for communications. For the PC version of the TIWAY package, hwy is always set to 1.
- rilen – The rilen argument identifies the number of Secondaries currently connected to the highway. If none are connected, rilen = 1 because ri254[1] = 0.
- ri254 – The addresses of the Secondaries currently connected to the highway are contained in this buffer. ri254 is an array of integers. Each element in the array (up to the number specified in rilen) contains the number of a connected Secondary, in integer format.

### Related Calls

The ACTVAT and DEACT subroutines can be used to connect or disconnect Secondaries from the highway.



The TI2HST subroutine converts a list of data of a single type from 500/505 format to host format. (The only subroutine that converts a list of returned data is the TIGET subroutine; the CIMRD, GATHER, NATIVE, RNDRD2, RNDRD3, RNDRD4, TIREAD, WRGAT, and XPAR subroutines return data in the 500/505 format.) TI2HST can convert from one buffer into another or the same buffer can be specified for source and destination.

The GATHER, WRGAT, and RNDRDx subroutines return buffers that may have to be converted in several segments.

**Call Format**

BASIC and FORTRAN: CALL TIWAY ("TI2HST", istat, xtn, tt, nnnn, tibuf, hstbuf)  
FORTRAN: CALL TI2HST (istat, xtn, tt, nnnn, tibuf, hstbuf)

Pascal: TIWAY ("TI2HST", istat, xtn, tt, nnnn, tibuf, hstbuf);  
TI2HST (istat, xtn, tt, nnnn, tibuf, hstbuf);

C: tiway ("ti2hst", &istat, &xtn, &tt, &nnnn, tibuf, &hstbuf);  
ti2hst (&istat, &xtn, &tt, &nnnn, tibuf, &hstbuf);

**Notes on Call Format**

Explanations of the terms used in the call format follow.

- istat – Error codes is returned in the status field. Declaration: integer.
- xtn – The transaction number must be zero. Declaration: integer.
- tt – The read only tt argument is the data element type (only low byte significant). Declaration: integer.
- nnnn – The read/write nnnn argument is the number of data elements to convert. Declaration: integer.
- tibuf – The read only 500/505 buffer argument is the source buffer (500/505 format). Declaration: byte array (FORTRAN); packed array [1..275] of byte (Pascal).
- hstbuf – The write only host buffer argument is the destination buffer (host format). Declaration: byte array (FORTRAN); packed array [1..275] of byte (Pascal).

Notice that no buffer lengths are passed. They are implied from the tt and nnnn fields.

**Related Calls**

The TI2HST subroutine is the reverse of the HST2TI subroutine.

## 10.37 TIGET

---

The TIGET subroutine is used to read consecutive memory locations from a specified NIM-based attached device and to convert the data from the 500/505 format to a format compatible with the host. TIGET is an implementation of Primitive 20.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("TIGET", istat, xtn, tag, nnnn, rsplen, ri135, sstat)  
FORTRAN: CALL TIGET (istat, xtn, tag, nnnn, rsplen, ri135, sstat)

Pascal: TIWAY ("TIGET", istat, xtn, tag, nnnn, rsplen, ri135, sstat);  
TIGET (istat, xtn, tag, nnnn, rsplen, ri135, sstat);

C: tiway ("tiget", &istat, &xtn, &tag, &nnnn, &rsplen, &ri135, &sstat);  
tiget (&istat, &xtn, &tag, &nnnn, &rsplen, &ri135, &sstat);

### Notes on Call Format

Explanations of the terms used in the call format follow.

istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).

xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.

tag – The tag argument specifies the address list that is to be returned to the caller. This argument may take any of the available forms. The tag is converted to an address list either by searching the tag table, by converting the specification directly, or by a combination of the two. See Declaration for tag\_type in Figure 3-1.

nnnn – The nnnn argument defines the number of data element locations to read.

rsplen – The response buffer length argument defines the maximum length of the response buffer (in bytes); when it is returned, it contains the actual length of the response. The response buffer length is the product of the number of bytes of converted data per element times the number of data element locations.

#### Example:

tag specifies V-memory (each element is two bytes long)  
nnnn = 16 (sixteen elements requested)  
rsplen = 32 (thirty-two bytes are returned)

ri135 – The read buffer contains the data returned by the Secondary. Allocate the buffer in the same variable type as either the converted data or be made equivalent to data of the variable type.

---

sstat – The Secondary-status argument contains the code for the current state of a specific NIM-based attached device. Current state means the state the device is in when the response to the subroutine call is returned. Current state is returned for all attached devices except Series 500 NIMs, Release 1.1 and earlier. (For these Series 500 NIMs, the state that is returned is the state that existed before the subroutine call was issued.) Table 3-7 shows the seven possible values.

**Related Calls**

The TIGET subroutine is similar to the TIREAD subroutine; however, TIGET converts to the host format and TIREAD does not.

## 10.38 TIPUT

---

The TIPUT subroutine is used to convert data from the host format to the 500/505 format and to write to consecutive memory locations in a specified NIM-based attached device. TIPUT is an implementation of Primitive 30.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("TIPUT", istat, xtn, tag, nnnn, wi135, sstat)  
FORTRAN: CALL TIWAY ("TIPUT", istat, xtn, tag, nnnn, wi135)  
Pascal: TIWAY ("TIPUT", istat, xtn, tag, nnnn, wi135, sstat);  
TIPUT (istat, xtn, tag, nnnn, wi135, sstat);  
C: tiway ("tiput", &istat, &xtn, &tag, &nnnn, &wi135, &sstat);  
tiput (&istat, &xtn, &tag, &nnnn, &wi135, &sstat);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- tag – The tag argument specifies the address list that is to be returned to the caller. This argument may take any of the available forms. The tag is converted to an address list either by searching the tag table, by converting the specification directly, or by a combination of the two. See Declaration for tag\_type in Figure 3-1.
- nnnn – The nnnn argument defines the number of data element locations to write.
- wi135 – The write buffer contains the data to be written in the host data format. The buffer either should be allocated in the same variable type as the converted data or be made equivalent to data of that type. This allows easy manipulation of the data.
- sstat – The Secondary-status argument contains the code for the current state of a specific NIM-based attached device. Current state means the state the device is in when the response to the subroutine call is returned. Current state is returned for all attached devices except Series 500 NIMs, Release 1.1 and earlier. (For these Series 500 NIMs, the state that is returned is the state that existed before the subroutine call was issued.) Table 3-7 shows the seven possible values.

### Related Calls

If data conversion is not required, the TIWRIT subroutine can be used.

## 10.39 TIREAD

---

The TIREAD subroutine reads consecutive memory locations from a specified NIM-based attached device. Data are returned in the 500/505 format as an array of bytes. TIREAD is an implementation of Primitive 20.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("TIREAD", istat, xtn, tag, nnnn, rsplen, rsp275, sstat)  
CALL TIREAD (istat, xtn, tag, nnnn, rsplen, rsp275, sstat)

Pascal: TIWAY ("TIREAD", istat, xtn, tag, nnnn, rsplen, rsp275, sstat);  
TIREAD (istat, xtn, tag, nnnn, rsplen, rsp275, sstat);

C: tiway ("tired", &istat, &xtn, &tag, &nnnn, &rsplen, &rsp275, &sstat);  
tired (&istat, &xtn, &tag, &nnnn, &rsplen, &rsp275, &sstat);

### Notes on Call Format

Explanations of the terms used in the call format follow.

istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).

xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.

nnnn – The nnnn argument defines the number of data element locations to be read.

rsplen – The response buffer length argument defines the maximum length of the response buffer; when it is returned, it contains the actual length in bytes of the response. See discussion of this argument for TIGET for an explanation on how it is computed.

rsp275 – The rsp275 argument contains the returned data in the 500/505 format.

### Related Calls

The TIREAD subroutine is similar to the TIGET subroutine except that TIREAD returns the data in 500/505 format.

The TIREAD subroutine is the complement of the TIWRIT subroutine.

## 10.40 TIWRIT

---

The TIWRIT subroutine writes to consecutive memory locations in a specified NIM-based attached device. The data are passed in the 500/505 format as an array of bytes. TIWRIT is an implementation of Primitive 30.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("TIWRIT", istat, xtn, tag, nnnn, wb275, sstat)  
FORTRAN: CALL TIWRIT (istat, xtn, tag, nnnn, wb275, sstat)

Pascal: TIWAY ("TIWRIT", istat, xtn, tag, nnnn, wb275, sstat);  
TIWRIT (istat, xtn, tag, nnnn, wb275, sstat);

C: tiway ("tiwrit", &istat, &xtn, &tag, &nnnn, &wb275, &sstat);  
tiwrit (&istat, &xtn, &tag, &nnnn, &wb275, &sstat);

### Notes on Call Format

Explanations of the terms used in the call format follow.

- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.
- tag – The tag argument specifies the address list that is to be returned to the caller. This argument may take any of the available forms. The tag is converted to an address list either by searching the tag table, by converting the specification directly, or by a combination of the two. See Declaration for tag\_type in Figure 3-1.
- nnnn – The nnnn argument defines the number of data element locations to write to.
- wb275 – The write buffer contains the data to be written. The data should be in the 500/505 format.
- sstat – The Secondary-status argument contains the code for the current state of a specific NIM-based attached device. Current state means the state the device is in when the response to the subroutine call is returned. Current state is returned for all attached devices except Series 500 NIMs, Release 1.1 and earlier. (For these Series 500 NIMs, the state that is returned is the state that existed before the subroutine call was issued.) Table 3-7 shows the seven possible values.

### Related Calls

If data conversion is required, the TIPUT subroutine can be used.

The TIWRIT subroutine is the complement of the TIREAD subroutine. The WRTGAT subroutine combines TIWRIT and GATHER into a single call.

## 10.41 TIXTN and TIXTNW

---

Each subroutine that does I/O is divided into two parts: the first half, which uses the argument list to build the command buffer and issue the I/O request, and the second half, which takes the results of the I/O (i.e., the response buffer) and returns pertinent parts of it to the application using the argument list. The TIXTN and TIXTNW subroutines are used to call the second half of a subroutine that does I/O and are not valid for any others.

If a subroutine is called with a transaction number of 0, it is synchronous: that is, after the first half, the routine waits until the I/O is complete and then executes the second half. In this case (the fully synchronous case), all processing is performed with a single call, either to the TIWAY entry point or to the individual subroutine. This is the normal case and is supported in all operating systems. The transaction number is a read-only variable and can be specified with the constant zero in the argument list.

If a subroutine is called with a transaction number of -1, only the first half of the subroutine is executed before control returns to the programmer. The transaction number is overwritten with a new number identifying that transaction, and a copy of the first half argument list is kept by the TIWAY library. To receive the results from the I/O (the second half), the TIXTN or TIXTNW subroutine must be called. Either of these subroutines is called with an istat and the transaction number that was returned from the first half. The results from the I/O in the second half are returned through the copy of the first half argument list; that is, no arguments to return data are required by the TIXTN or TIXTNW entry points.

The only difference between the TIXTN and TIXTNW subroutines occurs when the I/O has not completed when they are called. The TIXTN subroutine returns immediately with a status of FIRST HALF NOT FINISHED YET (facility 1, message 10 [hex]) and you should call it back later. The TIXTNW subroutine waits for the I/O to complete before returning (i.e., never returns with the FIRST HALF NOT FINISHED YET error).

### Call Format

BASIC and FORTRAN:	CALL TIWAY ("TIXTN", istat, xtn) CALL TIXTN (istat, xtn)	CALL TIWAY ("TIXTNW", istat, xtn) CALL TIXTNW (istat, xtn)
Pascal:	TIWAY ("TIXTN", istat, xtn); TIXTN (istat, xtn);	TIWAY ("TIXTNW", istat, xtn); TIXTNW (istat, xtn);
C:	tiway ("tixtn", &istat, &xtn); tixtn (&istat, &xtn);	tiway ("tixtnw", &istat, &xtn); tixtnw (&istat, &xtn);

### Notes on Call Format

Explanations of the terms used in the call format follow.

istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).

xtn – Supply the transaction number returned from the first half.

## 10.42 UPLOAD

---

The UPLOAD subroutine uploads the contents of a Secondary to a text file on the host system. This procedure is intended as a backup function for Secondary programs.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("UPLOAD", istat, tag, ftype, flen, fnam, pnlen, pname)  
CALL UPLOAD (istat, tag, ftype, flen, fnam, pnlen, pname)

Pascal: TIWAY ("UPLOAD", istat, tag, ftype, flen, fnam, pnlen, pname);  
UPLOAD (istat, tag, ftype, flen, fnam, pnlen, pname);

C: tiway ("upload", &istat, &tag, &ftype, &flen, &fnam, &pnlen, &pname);  
upload (&istat, &tag, &ftype, &flen, &fnam, &pnlen, &pname);

---

**NOTE:** For BASIC programmers: Because BASIC implements dynamic strings, force an extra character, preferably a blank, at the end of the string variable containing the tag.

Example:

```
TAG$="      " (six blanks)
Input TAG$
or
TAG$="#0102 "
```

This step is required because the TIWAY library tests the sixth character in a tag to determine length.

---

### Notes on Call Format

Explanations of the terms used in the call format follow.

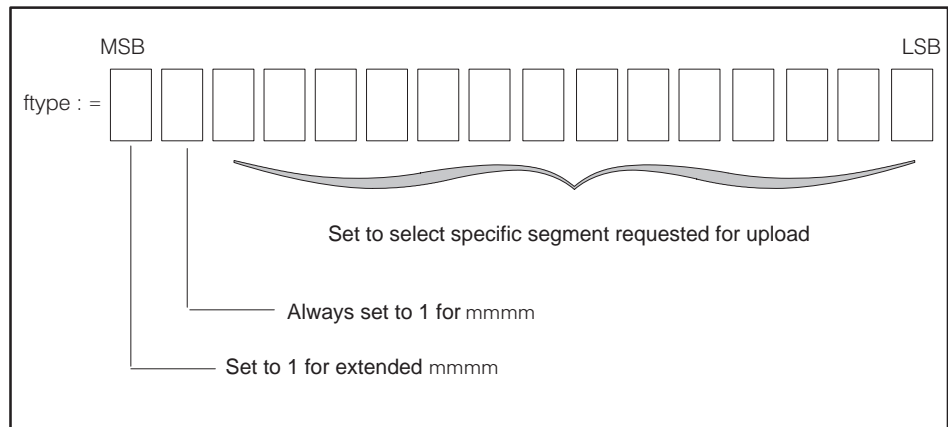
- istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).
- tag – The short form of the tag (highway and address) can be used. Since UPLOAD is written in FORTRAN, the tag argument must be passed by descriptor. See the programming manual for the language you are using for information on passing arguments by descriptor.



---

**f<sub>type</sub>** – Specifies what type of download is requested according to its value.  
Declaration: integer.

- 10 = generic upload of all memory
- 11 = generic upload of program memory
- 12 = generic upload of data memory
- 60 = upload all memory (NIM-attached device)
- 61 = partial upload L-memory
- 62 = partial upload V-memory
- 63 = partial upload Constant memory
- 64 = partial upload S-memory – loop tables
- 65 = partial upload S-memory – analog alarms
- 66 = partial upload S-memory – SF programs
- 67 = partial upload S-memory – SF subroutines
- >4000 = Unilink specific segment upload, which is translated to mmmm field in primitive where



**f<sub>len</sub>** – Length of file specification (filename). Declaration: integer.

**f<sub>nam</sub>** – File specification (filename) for uploaded data. Declaration: string or character array.

**p<sub>len</sub>** – Length of Unilink program name for specific segment transfer.  
Declaration: integer.

**p<sub>name</sub>** – Unilink program name for specific segment transfer. Declaration: string or character string.

## 10.43 WRBUF

---

The WRBUF subroutine is used to write to consecutive memory locations in NIM-based attached devices that support buffered program memory. It stores data temporarily in the NIM and downloads it from the NIM to the PLC upon request. WRBUF is the implementation of 5TI Primitive 33. The 5TI uses this subroutine for program downloads. Refer to the *SIMATIC 5TI NIM User Manual*.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("WRBUF", istat, xtn, tag, cc, nnnn, wb275, pstat, sstat)  
CALL WRBUF (istat, xtn, tag, cc, nnnn, wb275, pstat, sstat)

Pascal: TIWAY ("WRBUF", istat, xtn, tag, cc, nn, wb275, pstat, sstat);  
WRBUF (istat, xtn, tag, cc, nnnn, wb275, pstat, sstat);

C: tiway ("wrbuf", &istat, &xtn, &tag, &cc, &nnnn, &wb275, &pstat, &sstat);  
wrbuf (&istat, &xtn, &tag, &cc, &nnnn, &wb275, &pstat, &sstat);

### Notes on Call Format

Explanations of the terms used in the call format follow.

istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).

xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.

tag – The long form of the tag must be used.

cc – The Primitive option code specifies which of four memory storage actions should take place. Possible Primitive option code values are:

00 = Set storage RAM to null and start temporary storage of data.

01 = Continue temporary storage at address specified in tag.

02 = Replace attached device memory with storage memory.

03 = Abort temporary storage.

Primitive option code arguments may or may not be accompanied by data. The nnnn argument is used to determine whether the buffer is appended. If nnnn is 0, no data accompanies the cc argument. The specified amount of data accompanies the cc argument if nnnn is anything other than zero.

---

nnnn – The nnnn argument contains the number of data element locations being transferred by the call.

wb275 – The write buffer argument refers to the data element locations being transferred in the current call.

pstat – Primitive Status.  
0 = data transfer in progress.  
1 = data transfer has been aborted.

sstat – The Secondary-status argument contains the code for the current state of a specific NIM-based attached device. Current state means the state the device is in when the response to the subroutine call is returned. Current state is returned for all attached devices except Series 500 NIMs, Release 1.1 and earlier. (For these Series 500 NIMs, the state that is returned is the state that existed before the subroutine call was issued.) Table 3-7 shows the seven possible values.

 **WARNING**

Before transferring a new program by calling WRBUF, clear L-memory in its entirety (use FILL with pattern = 0000, see Chapter 4).

If this is not done, parts of residue programs execute, which could cause unpredictable operation in the controller that could result in death or serious injury to personnel and/or equipment damage.

Use FILL with pattern = 0000 to clear L-memory. See Chapter 4.

## 10.44 WRTGAT

---

The WRTGAT subroutine combines the functions of the TIWRIT (Chapter 4) and GATHER (Chapter 4) subroutines. It writes consecutive memory locations in a specific NIM-based attached device and then gathers data. The data are sent and returned in the 500/505 format. WRTGAT is the implementation of Primitive 52.

### Call Format

BASIC and FORTRAN: CALL TIWAY ("WRTGAT", istat, xtn, tag, nnnn, wb275, mask, rsplen, rsp275, sstat)  
CALL WRTGAT (istat, xtn, tag, nnnn, wb275, mask, rsplen, rsp275, sstat)

Pascal: TIWAY ("WRTGAT", istat, xtn, tag, nnnn, wb275, mask, rsplen, rsp275, sstat);  
WRTGAT (istat, xtn, tag, nnnn, wb275, mask, rsplen, rsp275, sstat);

C: tiway ("wrtgat", &istat, &xtn, &tag, &nnnn, wb275, &mask, &rsplen, &rsp275, &sstat);  
wrtgat (&istat, &xtn, &tag, &nnnn, wb275, &mask, &rsplen, &rsp275, &sstat);

### Notes on Call Format

Explanations of the terms used in the call format follow.

istat – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).

xtn – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.

tag – The long form of the tag *must* be used.

nnnn – The nnnn argument defines the number of data element locations to write to.

wb275 – The write buffer argument refers to the data element locations being transferred to the location specified in tag.

mask – See Chapter 3 for an explanation of masks.

rsplen – The length of the response buffer in bytes.

rsp275 – The rsp275 argument contains the data returned by GATHER in the 500/505 format.

---

sstat – The Secondary-status argument contains the code for the current state of a specific NIM-based attached device. Current state means the state the device is in when the response to the subroutine call is returned. Current state is returned for all attached devices except Series 500 NIMs, Release 1.1 and earlier. (For these Series 500 NIMs, the state that is returned is the state that existed before the subroutine call was issued.) Table 3-7 shows the seven possible values.

**Related Calls**

The WRTGAT subroutine is a combination of TIWRIT and GATHER.

The BLDMSK subroutine can be used to build the mask.

The XPAR subroutine provides a simple subroutine interface for formatting a command buffer, sending it to a Host Adapter (and by proper format, through a Host Adapter to any connected Secondary), and receiving a copy of the entire response. This transaction gives you the ability to use facilities that are not supported by other TIWAY Subroutines (TIWAY Primitives that are not supported, CIM Functional Commands that are not supported, and all Task Codes). A specific example is the Host Adapter Reset Command Code FF.

---

**NOTE:** You should be familiar with the Host Adapter and any addressed Secondaries before using the XPAR subroutine.

---

XPAR provides two services. First, the command and response buffers are formatted to contain the data to be interpreted by the Host Adapter without including the link level protocol encapsulation (NITP or BDLC protocol packet). Second, optional error detection is provided for Host Adapter, TIWAY Primitives, and CIM Functional Commands. (The CIM Functional Command error-checking is also suitable for PM550 CCU Task Codes.) An error-checking argument is used to determine the amount and type of error-checking performed.

The XPAR simply generates the message delimiters, the character count and the checksum for the message body you supply.

---

**NOTE:** After issuing a reset to the Unilink Host Adapter, you must follow these steps:

1. Issue a dummy command (such as command code 21) to the Unilink Host Adapter using the BDLC host protocol, because the first response following a UHA reset returns.
  2. Reconfigure the Unilink Host Adapter.
  3. Reconnect all Secondary devices.
  4. Reallocate all Source IDs.
  5. Redefine all macros.
-

---

## Call Format

**BASIC and FORTRAN:** CALL TIWAY ("XPAR", istat, xtn, hwy, cmdlen, cmd275, rsplen, rsp275, errtyp)  
CALL XPAR (istat, xtn, hwy, cmdlen, cmd275, rsplen, rsp275, errtyp)

**Pascal:** TIWAY ("xpar", istat, xtn, hwy, cmdlen, cmd275, rsplen, rsp275, errtyp);  
XPAR (istat, xtn, hwy, cmdlen, cmd275, rsplen, rsp275, errtyp);

**C:** TIWAY ("xpar", &istat, &xtn, &hwy, &cmdlen, cmd275, &rsplen, &rsp275, &errtyp);  
xpar (&istat, &xtn, &hwy, &cmdlen, cmd275, &rsplen, &rsp275, &errtyp);

## Notes on Call Format

Explanations of the terms used in the call format follow.

**istat** – Status which indicates successful completion of function, or error message indicating what problem occurred. See Section 3.11 (Status Determination).

**xtn** – Specifies synchronous (xtn=0) or asynchronous (xtn=1) completion.

**hwy** – The hwy argument specifies the highway to use for processing the command.

**cmdlen** – The cmdlen argument specifies the length of the command buffer.

**cmd275** – The cmd275 argument contains a command consisting of the following fields: Host Adapter Command Code, Secondary address, and network data, if applicable. The command is formatted for the Host Adapter to read. For more information, refer to the *Unilink Host Adapter User Manual*, PPX:TIWAY-8121-x.

**rsplen** – The rsplen argument specifies the maximum length of the response buffer (and the actual length on return [in bytes]).

**rsp275** – The response buffer contains the actual response from the Host Adapter on return. If any directive status errors (facility 7) or some of the I/O status errors (facility 6) were detected, rsp275 does not contain a response. One cause may be that the command buffer was not sent to the Host Adapter.

**errtyp** – The errtyp argument is specified by the caller to indicate the amount and type of error reporting to be performed on the istat variable. The valid options follow:

**00** – Argument errors, and I/O errors are reported.

**01** – Host Adapter errors are reported.

**02** – Host Adapter and TIWAY Primitive errors are reported.

**04** – Host Adapter and CIM Functional Command errors are reported.

# Appendix A

## Host Adapter Command Codes

---

Table A-1 Host Adapter Command Codes

Command Code	Description
00	Error response
01	Send network data
02	Broadcast network data
03	Poll Secondary
04	Connect Secondaries
05	Disconnect Secondaries
06	Read Secondary log
07	Read Secondary diagnostics
08	Read adapter diagnostics
10	Allocate source ID
11	Configure HIU
12	Report HIU configuration
13	Allocate macro storage buffers
14	Define macros
15	Gather macro responses
16	Enable/disable macro execution
17	Initialize macro response buffer
18	Report memory usage
19	Allocate HIU bandwidth
1A	Report HIU bandwidth allocation
1B	Report HIU Status
1C	Link macro
20	Configure UHA
21	Report UHA configuration
30	Configure network manager
31	Report network manager configuration
32	Report Secondary link status
33	Switch channel
34	Allocate network manager buffers

For further details, refer to the *SIMATIC TIWAY I UNILINK Host Adapter User Manual* (PPX:TIWAY-8121-x).



---

Table A-1 Host Adapter Command Codes (continued)

<b>Command Code</b>	<b>Description</b>
35	Report network manager buffers available
36	Report network bandwidth allocation
37	Report network manager Secondary statistics
38	Report network manager network statistics
FC	Modify external output status
FD	Report external input status
FE	Soft reset of host adapter
FF	Reset host adapter

# Appendix B

## TIWAY Primitives

---

Table B-1 TIWAY Primitives (Universal Command Language)

<b>Primitive</b>	<b>Description</b>
00	Exception reporting
01	Native Task Code
02	Status
03	Configuration
04	Primitive format configuration
05	Native Task Codes
10	Change state
20	Read block (single contiguous block)
21	Read random block
30	Write block (single contiguous block)
31	Write random block
32	Fill block
33	Buffered write
50	Define block (up to 32 separate blocks)
51	Gather blocks (as defined)
52	Write and gather defined blocks
55	Define records (up to 32 separate records)
56	Gather defined records
57	Write and gather defined records
60	Unilink Secondary Adapter

See the individual TIWAY NIM user manuals for the subset of TIWAY primitives supported by each Secondary.

# Appendix C

## CIM Functional Command Codes

---

Table C-1 CIM Functional Command Codes

Meaning	Command	Response
<p style="text-align: center;"><b>Read User Memory</b></p> <p>M: 2 = V-memory 4 = C-memory 8 = A-memory</p> <p>AAA = Hex starting address NN = Hex number of words (01–80) DDDD...= Hex memory contents</p>	60MAAANN	60DDD...
<p style="text-align: center;"><b>Write User Memory</b></p> <p>M: 2 = V-memory 4 = C-memory 8 = A-memory</p> <p>AAA = Hex starting address NN = Hex number of words (01–80) DDDD...= Data to be written</p>	61MAAANNDDDD...	61DDDD...(EIA) 61 (HDLC)
<p style="text-align: center;"><b>Read Image Register</b></p> <p>C: 1 = X 2 = CR 4 = Y</p> <p>AAA = Hex starting address NNN = Hex number of bits to read (001–100 for X or Y; 001–200 for CR) B... = Hex value of 4 bits</p>	62CAAANN	62B...
<p style="text-align: center;"><b>Write Image Register</b></p> <p>C: 2 = CR 4 = Y</p> <p>AAA = Hex starting address NNN = Hex number of bits to write (001–100 for X or Y; 001–200 for CR) B... = Hex value of four bits</p>	63CAAANNB...	63B...(EIA) 63 (HDLC)
<p style="text-align: center;"><b>Clear User Memory</b></p> <p>M: 1 = L-memory 2 = V-memory 4 = C-memory</p> <p style="text-align: right;">} Cumulative</p>	640M	64

Table C-1 CIM Functional Command Codes (continued)

Meaning	Command	Response
<p style="text-align: center;"><b>Download memory</b></p> <p>SS: 00 = initial download block            01 = continue download            02 = terminate download            08 = request download</p> <p>MM: 01 = L-memory }            02 = V-memory } Cumulative            04 = C-memory }</p> <p>II: 01 = CCU not in start up            02 = C and L in ROM            04 = extended C            08 = extended L</p> <p>DDDD... = hex data</p>	<p>65SSMM00DDDD...            or 65SS where            SS = 2</p>	<p>65SSMMII            or 65SS            where            SS = 2</p>
<p style="text-align: center;"><b>Upload memory</b></p> <p>SS: 00 = initial request            01 = continue            02 = terminate            04 = repeat previous block</p> <p>MM: 01 = L-memory }            02 = V-memory } Cumulative            04 = C-memory }</p> <p>II: 02 = C and L in ROM            04 = extended C            08 = extended L</p> <p>DDDD... = hex data</p>	<p>66SSMM00</p>	<p>66SSMMIIDD... </p>
<p style="text-align: center;"><b>Read errors</b></p> <p>XX = hex number of error types            NN = hex error number (00-2F)            CC = hex error count (00-FF)</p>	<p>67</p>	<p>67XXNNCCNNCC...</p>
<p style="text-align: center;"><b>Clear error</b></p> <p>NN = hex error number (00-2E or FF.)            FF clears all errors.</p>	<p>68NN</p>	<p>68</p>
<p style="text-align: center;"><b>Get CCU status</b></p> <p>X: 2 = start up            3 = hold            5 = run            6 = remote</p> <p>Y: 0 = not run            1 = program</p>	<p>69</p>	<p>69XY</p>

Table C-1 CIM Functional Command Codes (continued)

Meaning	Command	Response
<p><b>Read loop data</b> (PM550-500, -501, and -502 only)</p> <p>L: 0 = read all loops 1-8 = read loops specified by number</p> <p>T: 1 = display 2 = tuning 3 = display and tuning</p> <p>D1 = loop number D2,D3 = error status D4,D11 = process variable D12-D19 = setpoint D20-D23 = alarm status and loop status D24-D31 = output D32-D39 = process variable high range D40-D47 = process variable low range D48-D55 = bias D56-D63 = gain D64-D71 = rate D72-D79 = reset</p>	6ALT	6AD1...D79
<p><b>Read random data</b></p> <p>NN = hex number words to read (00-80)</p> <p>M: 2 = V-memory 4 = C-memory 8 = A-memory</p> <p>AAA = hex address DDDD.. = data read</p>	6BNNMAAAMAAA...	6BDDDD...
<p><b>Read CIM self test status</b></p> <p>X: 0 = no failures 1 = RAM failure 2 = ROM failure 4 = TMS9903 failure 8 = CIM/PM550 communication failure</p>	6C	6C0X

Table C-1 CIM Functional Command Codes (continued)

Meaning	Command	Response
<p><b>Read loop data (PM550-503)</b></p> <p>L: 0 = read loops 1-8 1-8 = read loop specified by number</p> <p>LL: 00 = read loops 1-8 01-10 = read loop specified by number FF = read loops 9 - 16</p> <p>T: 1 = display 2 = tuning 3 = display and tuning</p> <p>DO: = 1st digit loop number (old format)</p> <p>D4-D11 = process variable</p> <p>D12 - D19 = setpoint</p> <p>D20 - D23 = alarm status and loop status</p> <p>D24 - D31 = output</p> <p>D32 - D39 = process variable</p> <p>D40 - D47 = process variable, low range</p> <p>D2, D3 = error status</p> <p>D48 -D55 = bias</p> <p>D56-D63 = gain</p> <p>D64-D71 = rate</p> <p>D72-D79 = reset</p>	<p>6ALT (old)</p> <p>6AD0D1...D79</p>	<p>6AD1...D79 (old)</p> <p>6AD0D1...D79 (new)</p>
<p><b>Read random block</b></p> <p>Download block table</p> <p>Retrieve data</p> <p>Download and retrieve data</p> <p>Write VCA and retrieve data</p> <p>Write IR and retrieve data</p> <p>CC = hex block count (1 - 10)</p> <p>M: 2 = V-memory 4 = C-memory 8 = A-memory</p> <p>AAA = hex starting address</p> <p>NN = hex number words to read (1 - 80)</p> <p>DDDD = data read or written</p> <p>EEEE = 16-bit block selection mask</p> <p>C: 2 = CR 4 = Y</p> <p>QQQ = hex number of bits to write (001 - 100 for Y, 001-200 for CR)</p> <p>B = hex value of 4 IR bits</p>	<p>6D00CCMAAANN</p> <p>6D01EEEE</p> <p>6D02EEEECCM AAANN...</p> <p>6D03EEEEEMAAANN DDDD</p> <p>6D04EEEECAAA QQQB...</p>	<p>6D00</p> <p>6D01EEEEEDDDD...</p> <p>6D02EEEECCCC...</p> <p>6D03EEEEEDDDD...</p> <p>6D04EEEEEDDDD...</p>

# Appendix D

## PM550 CCU Task Codes

---

This appendix provides a limited definition of the PM550 CCU Task Codes.  
Please refer to the *SIMATIC PM550 NIM User Manual* for more details.

Table D-1 CCU Task Codes for PM550

Meaning	Task Code	Response
<p style="text-align: center;"><b>Return CCU Status</b></p> <p>A: 2 = Start Up 3 = Hold 5 = Run</p> <p>B: 0 = Not Program Mode 1 = Program Mode</p> <p>RR: Current CCU revision (must be 07 or greater for 8 loop CCUs)</p> <p>X: 0 = CCU not in Remote 1 = CCU in Remote</p>	01	01ABRRX
<p style="text-align: center;"><b>Read Error NN</b></p> <p>NN: = Error number from system error DD = Number of occurrences of error NN</p>	02NN	02DD
<p style="text-align: center;"><b>CCU RAM/ROM Status and Memory</b></p> <p>N: 1 = L in RAM 2 = L in ROM 4 = C in RAM 8 = C in ROM B = Copy L ROM to RAM C = Copy C ROM to RAM</p> <p>N is not cumulative.</p>	04N	04 or 00ER

Table D-1 CCU Task Codes for PM550 (continued)

Meaning			Task Code	Response
<b>System Configuration</b>			05	05DDDDD
<b>Byte</b>	<b>Bit</b>	<b>Function</b>		
D(1)	8	C in ROM		
	4	C in RAM		
	2	L in ROM		
	1	L in RAM		
D(2)	8	Port 1 1200 bps		
	4	Port 1 300 bps		
	2	Port 1 smart		
	1	Port 1 dumb		
D(3)	8	Port 2 1200 bps		
	4	Port 2 300 bps		
	2	Port 2 smart		
	1	Port 2 dumb		
D(4)	8	L = 4K		
	4	C = 2K		
	2	CIM/NIM		
D(5)	1	0 = DCE 1 = DTE		
<b>Clear Error NN</b>			06NNDD	06
NN =	Error number from Read Error NN (FF = clear error table)			
DD =	Number of occurrences from Read Error NN			
<b>Program L-/C-Memory</b>			07M	07NXAAAA
M:	1 = L-memory 2 = C-memory			
N:	1 = Programming complete 2 = Programming incomplete			
X:	1 = TMS2516 EPROM 2 = TMS2532 EPROM			
AAAA:	Address of invalid verify			
<b>Execute Single Memory Scan</b>			0A	0A
<b>Enter Program Mode</b>			0B	0B
<b>Leave Program Mode</b>			0C	0C
<b>Perform CCU Self-Diagnostics</b>			0D12134	None



Table D-1 CCU Task Codes for PM550 (continued)

Meaning	Task Code	Response
<p><b>Verify Memory Contents</b></p> <p>M: 1 = Verify L 2 = Verify C</p> <p>AAAA: Address to be verified</p> <p>N: 1 = Verify complete 2 = Verify/parity message</p> <p>X: 1 = TMS2516 EPROM 2 = TMS2532 EPROM</p> <p>DDDD = RAM data</p> <p>RRRR = ROM data</p> <p>PP = Parity</p>	0EMAAAA	NXAAAADDDDRRRRPP or error code
<p><b>Read Power Flow</b></p> <p>AAAA = Address (C000 – CFFF for 2K, C000 – DFFF for 4K)</p> <p>P = State of power flow (0 or 1) at AAAA</p>	10AAAA	10P0
<p><b>Enter Remote State</b> (Rev. 8.0 only)</p>	11	11
<p><b>Read Ladder (L) Memory</b></p> <p>AAAA = Starting address (C000 – CFFF for 2K, C000 – DFFF for 4K)</p> <p>N = Number of addresses to read (1 – 6)</p> <p>DDDD... = Memory data</p>	12AAAAAN	12DDDD...
<p><b>Leave Remote State</b> (Rev. 8.0 only)</p>	13	13
<p><b>Write L-Memory Instruction or Argument*</b></p> <p>AAAA = Starting address (C000 –CFFF for 2K, C000 –DFFF for 4K)</p> <p>DDDD... = Instruction or argument</p> <p><b>*NOTE:</b> Enter all parts of timer or counter instruction with one Task Code</p>	14AAAADDDDD	14DDDD...

Table D-1 CCU Task Codes for PM550 (continued)

Meaning	Task Code	Response
<p align="center"><b>Programming Status</b></p> <p>N: 0 = Program in progress            1 = Program complete            2 = Program incomplete            3 = Erase error</p> <p>X: 1 = TMS2516 EPROM            2 = TMS2532 EPROM</p> <p>AAAA: Address of invalid verify</p>	15	15NXAAAA
<p align="center"><b>Fill L-Memory</b></p> <p>AAAA: Starting address (C000 – CFFF for 2K,            C000 – DFFF for 4K)</p> <p>NNNN: Number of addresses to fill</p> <p>FFFF: Data word to fill in memory</p>	16AAAAANNNNFFFF	16
<p align="center"><b>Delete an L-Memory</b></p> <p>AAAA: Address (C000 – CFFF for 2K,            C000 – DFFF for 4K)</p> <p>DDDD = Words to be inserted</p>	17AAAA	17DDDD
<p align="center"><b>Insert Words Into L-Memory</b></p> <p>AAAA = Starting address (C000 – CFFF for 2K,            C000 – DFFF for 4K)</p> <p>DDDD = Words to be inserted</p>	18AAAAADDDD...	18DDDD...
<p align="center"><b>Find an L-Memory Word</b></p> <p>AAAA = Task Code starting address            (C000 – CFFF for 2K,            C000 – DFFF for 4K)            or response address of XXXX</p> <p>XXXX = Word to be found</p> <p>MMMM = Bits to be ignored</p>	19AAAAAXXXMMMM	19AAAAADDDD
<p align="center"><b>Find IR Address of L-Memory</b></p> <p>AAAA = Task Code starting address            (C000 – CFFF for 2K, C000 – DFFF            for 4K) or response address of XXXX</p> <p>XXXX = Word whose address is to be found</p> <p>DDDD = Word at AAAA</p>	1AAAAAXXX	1AAAAADDDD

Table D-1 CCU Task Codes for PM550 (continued)

Meaning	Task Code	Response
<p><b>Find Illegal Ladder Instruction</b> (Rev. 8.1 only)</p> <p>AAAA = L-Memory location to begin search (Task Code); L-memory location of illegal instruction (response)</p> <p>CC: 00 = No illegal instructions            01 = Illegal ladder instruction            02 = Invalid parameter            03 = Invalid special function control block            04 = Partial instruction at end of memory</p> <p>LLLLLL: Decimal L-memory location of illegal instruction. If no illegal instructions are found, 1B100 is returned.</p>	1BAAAA	1BCCAAAALLLLLL
<p><b>Read Image Register</b></p> <p>AAAA: 0C00 – 0CFF = X0 – X255            0D00 – 0EFF = CR0 – CR511            0F00 – 0FFF = Y0 – 255</p> <p>N = Optional number of contiguous values to read (1 – F)</p> <p>B = Value (0 or 1) of discrete I/O read</p>	1DAAAAN	1DB...B
<p><b>Write Image Register</b></p> <p>AAAA: 0C00 – 0CFF = X0 – X255            0D00 – 0EFF = CR0 – CR511            0F00 – 0FFF = Y0 – 255</p> <p>B = Value (0 or 1) of discrete I/O read 13 characters maximum</p>	1EAAAAB	1EB...B
<p><b>Read Yellow Deviation Value</b></p> <p>L = Loop number (1 – 8)            F Indicates presence of 16-loop software            LL = Loop number (1 – 16) for 16-loop software</p> <p>DDDDDDDD = Yellow deviation value in excess 64 floating point notation</p>	20L or 20FLL	20DDDDDDDD
<p><b>Write Yellow Deviation Value</b></p> <p>L = Loop number (1 – 8)            F = Presence of 16-loop software            LL = Loop number (1 – 16) for 16-loop software</p> <p>DDDDDDDD = Yellow deviation value in excess 64 floating point notation</p>	21LDDDDDDDD or 21FLDDDDDDDD	21DDDDDDDD

Table D-1 CCU Task Codes for PM550 (continued)

Meaning	Task Code	Response
<p><b>Read User Memory (V, C, or A)</b></p> <p>AAAA: E000 – E7FF = V  E800 – EFFF = C (1K)  F000 – F7FF = C (2K)  F800 – FBFF = A</p> <p>N = Optional number of contiguous words to be read (1 – F)</p> <p>DDDD = Value at one address</p>	22AAAAAN	22DDDD..
<p><b>Read Random Memory (V, C, or A) (Rev. 8.0 only)</b></p> <p>AAAA: E000 – E7FFF = V  E800 – EFFF = C (1K)  E800 – F7FF = C (2K)</p> <p>DDDD = Data read</p>	23AAAAAAAAA...	23DDDDDDDD...
<p><b>Write User Memory (V, C, or A)</b></p> <p>AAAA: E000 – E7FF = V  E800 – EFFF = C (1K)  E800 – F7FF = C (2K)  F800 – FBFF = A</p> <p>DDDD: Data to be written</p>	24AAAADDDDD	24DDDD
<p><b>Fill User Memory with Ones</b></p> <p>AAAA: E000 – E7FF = V  E800 – EFFF = C (1K)  E800 – F7FF = C (2K)  F800 – FBFF = A</p> <p>NNNN = Number of contiguous locations to fill.</p> <p>FFFF = 11111111111111</p>	26AAAAANNNNFFFF	6AAAAANNNNFFFF
<p><b>Read Status of Timer NN</b></p> <p>NN = Number of counter (1 – F)</p> <p>PPPP = Preset value</p> <p>CCCC = Current value</p> <p>F: 0 = Not protected and CCU is executing ladder logic  1 = Protected and CCU is executing ladder logic  8 = Not protected and CCU is not executing ladder logic  9 = Protected and CCU is not executing ladder logic</p>	27NN	27PPPPCCCCF

Table D-1 CCU Task Codes for PM550 (continued)

Meaning	Task Code	Response
<p><b>Write Timer Preset Value</b></p> <p>NN = Number of timer (1 – F)                      PPPP = Preset value                      F: 0 = Not protected and CCU is executing ladder logic                      1 = Protected and CCU is executing ladder logic                      8 = Not protected and CCU is not executing ladder logic                      9 = Protected and CCU is not executing ladder logic</p>	28NNPPPP	28PPPPF
<p><b>Write Timer Current Value</b></p> <p>NN = Number of timer (1 – F)                      CCCC = Current value                      F: 0 = Not protected and CCU is executing ladder logic                      1 = Protected and CCU is executing ladder logic                      8 = Not protected and CCU is not executing ladder logic                      9 = Protected and CCU is not executing ladder logic</p>	29NNCCCC	29CCCCF
<p><b>Read Status of Counter NN</b></p> <p>NN = Number of timer (1 – F)                      PPPP = Preset value                      CCCC = Current value                      F: 0 = Not protected and CCU is executing ladder logic                      1 = Protected and CCU is executing ladder logic                      8 = Not protected and CCU is not executing ladder logic                      9 = Protected and CCU is not executing ladder logic</p>	2ANN	2APPPCCCCF

Table D-1 CCU Task Codes for PM550 (continued)

Meaning	Task Code	Response
<p><b>Write Counter Preset Value</b></p> <p>NN = Number of counter (1 – F)</p> <p>PPPP = Preset value</p> <p>F: 0 = Not protected and CCU is executing ladder logic</p> <p>1 = Protected and CCU is executing ladder logic</p> <p>8 = Not protected and CCU is not executing ladder logic</p> <p>9 = Protected and CCU is not executing ladder logic</p>	2BNNPPPP	2BPPPPF
<p><b>Write Counter Current Value</b></p> <p>NN = Number of counter (1 – F)</p> <p>CCCC = Current value</p> <p>F: 0 = Not protected and CCU is executing ladder logic</p> <p>1 = Protected and CCU is executing ladder logic</p> <p>8 = Not protected and CCU is not executing ladder logic</p> <p>9 = Protected and CCU is not executing ladder logic</p>	2CNNCCCC	2CCCCCF

Table D-1 CCU Task Codes for PM550 (continued)

Meaning	Task Code	Response																		
<p style="text-align: center;"><b>Read Loop Variable</b></p> <p>L: 0 = Select next loop            1-8 = Loop number</p> <p>F Indicates presence of 16-loop software</p> <p>LL = Loop number (1-16) for 16-loop software</p> <p>M: Lower display mode            0 = Blank (do not use)            1 = Setpoint (SP)            2 = Deviation            3 = Output            4 = Bias            5 = Gain            6 = Rate            7 = Reset</p> <p>D0 = First digit of loop number (0-1) for 16-loop software</p> <p>D1 = Loop number (1-8) for 8-loop software;            second digit of loop number (1-6) for 16-loop software</p> <p>D3-D10 = Process variable in floating point</p> <p>D11-D18 = Lower display value in floating point</p> <p>D19-D22 = V-table loop status word</p> <p>D23 = CCU status flags</p> <table border="0" style="margin-left: 40px;"> <thead> <tr> <th style="text-align: left;"><u>Bit</u></th> <th style="text-align: left;"><u>Value</u></th> </tr> </thead> <tbody> <tr><td>0</td><td>X</td></tr> <tr><td>1</td><td>X</td></tr> <tr><td>2</td><td>X</td></tr> <tr><td>3</td><td>X</td></tr> <tr><td>4</td><td>A</td></tr> <tr><td>5</td><td>B</td></tr> <tr><td>6</td><td>C</td></tr> <tr><td>7</td><td>X</td></tr> </tbody> </table> <p>If: A = 1, loops are being calculated            B = 1, a loop is critical            X is not used</p>	<u>Bit</u>	<u>Value</u>	0	X	1	X	2	X	3	X	4	A	5	B	6	C	7	X	2DLM or 2DFLLM	2DD1D2...D23 or 2DD0D1...D23
<u>Bit</u>	<u>Value</u>																			
0	X																			
1	X																			
2	X																			
3	X																			
4	A																			
5	B																			
6	C																			
7	X																			

Table D-1 CCU Task Codes for PM550 (continued)

Meaning	Task Code	Response																		
<p style="text-align: center;"><b>Read Loop Constant</b></p> <p>L: 0 = Select next loop            1-8 = Loop number</p> <p>F Indicates presence of 16-loop software</p> <p>LL = Loop number (1-16) for 16-loop software</p> <p>M: Lower display mode            0 = Blank (do not use)            1 = Setpoint (SP)            2 = Deviation            3 = Output            4 = Bias            5 = Gain            6 = Rate            7 = Reset</p> <p>D0 = First digit of loop number (0-1) for 16-loop software</p> <p>D1 = Loop number (1-8) for 8-loop software; second digit of loop number (1-6) for 16-loop software</p> <p>D3-D10 = Process variable in floating point</p> <p>D11-D18 = Lower display value in floating point</p> <p>D19-D22 = V-table loop status word</p> <p>D23 = CCU status flags</p> <table border="0" style="margin-left: 40px;"> <thead> <tr> <th style="text-align: left;"><u>Bit</u></th> <th style="text-align: left;"><u>Value</u></th> </tr> </thead> <tbody> <tr><td>0</td><td>X</td></tr> <tr><td>1</td><td>X</td></tr> <tr><td>2</td><td>X</td></tr> <tr><td>3</td><td>X</td></tr> <tr><td>4</td><td>A</td></tr> <tr><td>5</td><td>B</td></tr> <tr><td>6</td><td>C</td></tr> <tr><td>7</td><td>X</td></tr> </tbody> </table> <p>If: A = 1, loops are being calculated            B = 1, a loop is critical            X is not used</p>	<u>Bit</u>	<u>Value</u>	0	X	1	X	2	X	3	X	4	A	5	B	6	C	7	X	<p>2EL or 2EFLL</p>	<p>2ED1D2...D23 or 2DD0D1...D23</p>
<u>Bit</u>	<u>Value</u>																			
0	X																			
1	X																			
2	X																			
3	X																			
4	A																			
5	B																			
6	C																			
7	X																			



Table D-1 CCU Task Codes for PM550 (continued)

Meaning	Task Code	Response
<p style="text-align: center;"><b>Write a Loop</b></p> <p>L: 1–8 = Loop Number</p> <p>F Indicates presence of 16-loop</p> <p>LL = Loop number (1–16) for 16-loop software</p> <p>M: Lower display mode</p> <p>0 = Blank (do not use)</p> <p>1 = Setpoint (SP)</p> <p>2 = Deviation</p> <p>3 = Output</p> <p>4 = Bias</p> <p>5 = Gain</p> <p>6 = Rate</p> <p>7 = Reset</p> <p>8 = Manual Mode Request</p> <p>9 = Auto Mode Request</p> <p>A = Cascade Mode Request (Closed Cascade Mode)</p>	<p>2FLMDDDDDDDD or 2FFLMDDDDDDDD</p>	<p>2FDDDDDDDD or 2FFDDDDDDDD</p>

Table D-1 CCU Task Codes for PM550 (continued)

Meaning	Task Code	Response
<p><b>Read CCU Status</b> (Rev. 8.0 and later)</p> <p>AAA = System state            000 = Power fail            001 = Idle start            010 = Start state            011 = Hold state            100 = Wait state            101 = Run state            110 = Remote state</p> <p>B = Program Mode status            0 = Not in Program            1 = In Program</p> <p>C = Switch 1 set for:            0 = COMM 1 dumb            1 = COMM 1 smart</p> <p>D = Switch 2 set for:            0 = COMM 1 @ 300 bps            1 = COMM 1 @ 1200 bps</p> <p>E = Switch 3 set for:            0 = COMM 2 dumb            1 = COMM 2 smart</p> <p>F = Switch 4 set for:            0 = COMM 2 @ 300 bps            1 = COMM 2 @ 1200 bps</p> <p>G = Switch 5 set for:            0 = 2K L-memory            1 = 4K L-memory</p> <p>H = Switch 6 set for:            0 = 1K C-memory            1 = 2K C-memory</p> <p>I = Not defined</p> <p>J = Switch 8 set for:            0 = CIM not installed            1 = CIM installed</p> <p>K = Not defined</p> <p>L = Switch 10 set for:            0 = DTE            1 = DCE</p> <p>M = C-memory in:            0 = ROM            1 = RAM</p> <p>N = L-memory in:            0 = ROM            1 = RAM</p>	<p>30</p>	<p>30 (six status words)            status word 1 =            AAABCDEF GHIJKLMN</p>



Table D-1 CCU Task Codes for PM550 (continued)

Meaning	Task Code	Response
<p><b>Write Process Variable (PV) (low value)</b>                      L = Loop number (1–8)                      F = Presence of 16-loop software                      LL = Loop number (1–16) for 16-loop software                      DDDDDDDD = PV low value in excess 64 floating point notation</p>	<p>31LDDDDDDDD                      or                      31FLLDDDDDDDD</p>	31DDDDDDDD
<p><b>Read Process Variable (PV) (high value)</b>                      L = Loop number (1–8)                      F = Presence of 16-loop software                      LL = Loop number (1–16) for 16-loop software                      DDDDDDDD = PV high value in excess 64 floating point notation</p>	<p>32L                      or                      32FLL</p>	32DDDDDDDD
<p><b>Write Process Variable (PV) (high value)</b>                      L = Loop number (1–8)                      F = Presence of 16-loop software                      LL = Loop number (1–16) for 16-loop software                      DDDDDDDD = PV high value in excess 64 floating point notation</p>	<p>33LDDDDDDDD                      or                      33FLLDDDDDDDD</p>	33DDDDDDDD
<p><b>Read Orange Deviation Value</b>                      L = Loop number (1–8)                      F = Presence of 16-loop software                      LL = Loop number (1–16) for 16-loop software                      DDDDDDDD = Orange deviation in excess 64 floating point notation</p>	<p>34L                      or                      34FLL</p>	34DDDDDDDD
<p><b>Write Sample Rate</b>                      L = Loop number (1–8)                      F = Presence of 16-loop software                      LL = Loop number (1–16) for 16-loop software                      DDDD = Sample rate</p>	<p>35LDDDD                      or                      35FLLDDDD</p>	35DDDD
<p><b>Write Orange Deviation Value</b>                      L = Loop number (1–8)                      F = Presence of 16-loop software                      LL = Loop number (1–16) for 16-loop software                      DDDDDDDD = Orange deviation in excess 64 floating point notation</p>	<p>36LDDDDDDDD                      or                      36FLLDDDDDDDD</p>	36DDDDDDDD

Table D-1 CCU Task Codes for PM550 (continued)

Meaning	Task Code	Response
<p><b>Write Loop Gain</b></p> <p>L = Loop number (1–8)            F = Presence of 16-loop software            LL = Loop number (1–16) for 16-loop software            DDDDDDDD = Orange deviation in excess 64 floating point notation</p>	<p>37LDDDDDDDD            or            37FLLDDDDDDDD</p>	<p>37DDDDDDDD</p>
<p><b>Read Reset Value</b></p> <p>L = Loop number (1–8)            F = Presence of 16-loop software            LL = Loop number (1–16) for 16-loop software            DDDDDDDD = Reset value in excess 64 floating point notation</p>	<p>38L            or            38FLL</p>	<p>38DDDDDDDD</p>
<p><b>Write Reset Value</b></p> <p>L = Loop number (1–8)            F = Presence of 16-loop software            LL = Loop number (1–16) for 16-loop software            DDDDDDDD = Reset value in excess 64 floating point notation</p>	<p>39LDDDDDDDD            or            39FLLDDDDDDDD</p>	<p>39DDDDDDDD</p>
<p><b>Read Loop Rate</b></p> <p>L = Loop number (1–8)            F = 16-loop            LL = Loop number (1–16) for 16-loop software            DDDDDDDD = Loop rate in excess 64 floating point notation</p>	<p>3AL            or            3AFLL</p>	<p>3ADDDDDDD            or            3AFDDDDDD</p>
<p><b>Write Loop Rate</b></p> <p>L = Loop number (1–8)            F = 16-loop            LL = Loop number (1–16) for 16-loop software            DDDDDDDD = Loop rate in excess 64 floating point notation</p>	<p>3BLDDDDDDDD            or            3BFLL</p>	<p>3BDDDDDDDD            or            3AFDDDDDD</p>
<p><b>Read High Alarm Value</b></p> <p>L = Loop number (1–8)            F = 16-loop            LL = Loop number (1–16) for 16-loop software            DDDDDDDD = High alarm value in excess 64 floating point notation</p>	<p>3CL            or            3CFLL</p>	<p>3CDDDDDDDD            or            3CFDDDDDD</p>

Table D-1 CCU Task Codes for PM550 (continued)

Meaning	Task Code	Response
<p><b>Write High Alarm Value</b></p> <p>L = Loop number (1-8)            F = 16-loop            LL = Loop number (1-16) for 16-loop software            DDDDDDDD = High alarm value in excess 64 floating point notation</p>	<p>3DLDDDDDDDD            or            3DFLLDDDDDDDD</p>	<p>3DDDDDDDD            or            3DFDDDDDDDD</p>
<p><b>Read Low Alarm Value</b></p> <p>L = Loop number (1-8)            F = 16-loop            LL = Loop number (1-16) for 16-loop software            DDDDDDDD = Low alarm value in excess 64 floating point notation</p>	<p>3EL            or            3EFL</p>	<p>3EDDDDDDD            or            3EFDDDDDDDD</p>
<p><b>Write Low Alarm Value</b></p> <p>L = Loop number (1-8)            F = 16-loop            LL = Loop number (1-16) for 16-loop software            DDDDDDDD = Low alarm value in excess 64 floating point notation</p>	<p>3FLDDDDDDDD            or            3FFLLDDDDDDDD</p>	<p>3FDDDDDDDD            or            3FFDDDDDDDD</p>

# Appendix E

## Error Listings

---

E.1	Facility 1: TIWAY Subroutine Library Exceptions .....	E-2
E.2	Facility 2: Network Exceptions .....	E-5
E.3	Facility 3: Host Adapter Internal Exceptions .....	E-6
E.4	Facility 4: Host Adapter Exceptions .....	E-8
E.5	Facility 5: TIWAY Primitive Exceptions .....	E-9
E.6	Facility 6: Operating System Status Exceptions .....	E-11
E.7	Facility 7: I/O Status Exceptions .....	E-12
E.8	Facility 8: CIM Exceptions .....	E-13
E.9	Facility 9: Native Task Codes .....	E-14

## E.1 Facility 1: TIWAY Subroutine Library Exceptions

---

These errors (shown in Table E-1) are detected by the TIWAY Subroutines and generally represent invalid or inconsistent arguments. All subroutines are discussed in this user guide.

Table E-1 Facility 1: TIWAY  
(Internal to TIWAY Subroutine Library Errors)

<b>Message Number (Hex)</b>	<b>Description</b>
00	Unable to map tag table
01	No such function, bad dispatch argument
02	Not enough arguments for specified function
03	Too many arguments for specified function
04	Wrong INITed state for specified function
05	Wrong length response for query
06	Highway number out of range
07	Bad ASCII tag specification
08	Tag not found
09	User-supplied buffer too small for requested function
0A	Unexpected response length from NIM
0B	Invalid CIM memory or image register data type
0C	Illegal number of blocks defined
0D	Illegal data type (TT, MM, or CC)
0E	Illegal transaction number for 1st/2nd half call
0F	Illegal state for transaction number in 1st/2nd half call
10	First half not finished yet
11	Illegal floating point number, host format
12	Illegal floating point number, 500/505 controller format
13	Invalid number of Secondaries
14	Too much data specified for operation
15	Errtype parameter out of range
16	Secondary address out of range
17	Address out of range for non-extended addressing
18	CIM address out of range
19	Wblen does not match the length derived from TT and NNNN fields



Table E-1 Facility 1: TIWAY  
(Internal to TIWAY Subroutine Library Errors) (continued)

Message Number (Hex)	Description
1A	NIM primitive sent to CIM device
1B	CIM functional command sent to NIM device
20	Error reading loop programs
21	Error reading analog alarms
22	Error reading Special Function programs or subprograms
23	Error reading I/O configuration
24	Error writing loop programs
25	Error writing analog alarms
26	Error writing Special Function programs or subprograms
27	Error writing I/O configuration
28	Error reading scan time
29	Error writing scan time or forced I/O
30	Error writing S-memory
31	Error in cold start after memory clear
32	Error in downloading memory configuration table
33	NIM/CIM Secondary mismatch
34	Secondary model mismatch
35	No support for model type
36	CCU not in startup or remote mode—cannot continue
37	CCU not in startup or remote mode—cannot continue download
38	Secondary must be in non-execution state
39	Invalid file format found
3A	Invalid download file format
3B	Error reading file
3C	Invalid file specification
3D	Inconsistent memory sizes
3E	Invalid memory type specified for 5TI
3F	Invalid primitive status from buffered write

## Facility 1: TIWAY Subroutine Library Exceptions (continued)

---

Table E-1 Facility 1: TIWAY  
(Internal to TIWAY Subroutine Library Errors) (continued)

<b>Message Number (Hex)</b>	<b>Description</b>
40	Transfer between requested Secondaries not allowed
41	Device does not support generic upload
42	Device does not support generic download
43	Error in initial generic upload request
44	Error in initial generic download request
45	Invalid upload request
46	Invalid download request
47	Memory type not configured for this device
48	Requested record type not found in upload file
49	Error in creating CIM permanent file
4A	Device does not support partial upload
4B	Device does not support partial download
4C	No segment specified for specific transfer
4D	WARNING! Not all specified segments transferred
4E	Error reading specific segment transfer file
4F	Device does not support specific segment upload
50	Device does not support specific segment download
51	Device rejected specific segment download request
52	Device rejected generic download request
53	Data not ready for generic upload request
60	Invalid number of config options
61	Invalid reset option
62	Too many macro buffers requested
63	Invalid xtn received
64	No macro buffers available
65	Unable to map highway port list
66	Highway port number not defined
70	Valid response return — transaction continuing
71	Unable to enable frozen macro

## E.2 Facility 2: Network Exceptions

---

These errors, detected by the Host Adapter, pertain to the TIWAY communications. See Table E-2.

Table E-2 Facility 2: NETEXCEPT  
(Host Adapter—Network Exception Errors)

<b>Message Number (Hex)</b>	<b>Description</b>
00	Undefined network problem
01	Secondary timed out
02	Miscellaneous Secondary protocol errors
03	Wrong Secondary responded
04	Secondary frame too short
05	Secondary frame too long
06	Bad (null) address
07	No data returned from Secondary
08	Receiver buffer overflow
09	Block checksum errors on frame received
0A	Frames aborted by Secondary
0B	Secondary timeouts
0C	Transmitter underruns
0D	Receiver overruns
0E	Frame did not end on byte boundary
10	Lost DCD on RS-232 receive
11	Lost CTS on RS-232 receive
12	Received an invalid frame
13	Illegal communications interrupt
14	Timed-out waiting on DCD (half-duplex RS-232)
15	Timed-out waiting on CTS (RS-232)
31	Message length error
41	A HIU to NM message command was out of range
42	A HIU to NM message field was out of range
43	A broadcast mode error occurred in a HIU to NM message
44	Incorrect HDLC field in Secondary response

## E.3 Facility 3: Host Adapter Internal Exceptions

---

These errors, detected by the Host Adapter, pertain to inconsistencies or other internal errors. See Table E-3.

Table E-3 Facility 3: INTERNEXCP  
(Host Adapter—Adapter Internal Exception Errors)

<b>Message Number (Hex)</b>	<b>Description</b>
00	Undefined adapter problem
01	Memory management error
02	No media card installed on host port
03	System stack management error
04	System queue management error
05	Illegal interrupt received by processor
06	Invalid op-code encountered by processor
07	Buffer management error
08	No buffer available at present
09	No network media card installed
0A	All memory is allocated
0B	ONLINE switch in OFFLINE position or HIU not initialized
0C	Not enough memory to allocate Secondary status block
0D	Not enough memory to allocate Source ID
0E	Requested bandwidth statistics not available
0F	Invalid auto redundant request
12	NM and HIU not configured
13	NM not configured
14	HIU not configured
16	Option sent twice
17	HIU is already configured
18	Mixed media definition for channel
19	Device addressed is not a HIU
1A	Not enough memory for specified macros

Table E-3 Facility 3: INTERNEXCP  
(Host Adapter—Adapter Internal Exception Errors) (continued)

Message Number (Hex)	Description
26	Macro is already enabled
20	Invalid source ID
21	Maximum number of source IDs already allocated
22	Source ID not allocated
23	Invalid macro type
24	Macro command buffer too small
25	Macro buffer not allocated
27	Macro and command source IDs are not equal
28	Command not valid for non-repetitive macro
29	Exception flag set on macro
2A	Response was larger than allocated buffer space
2B	Action macro is enabled
2C	Action macro primitive error
2D	Trigger and action macro data types do not match
2E	Macro link already exists
2F	No macro link exists
30	Trigger macro not disabled
31	Trigger macro primitive error
32	NM buffer definition not allowed
33	NM buffers have already been allocated
34	Macro command buffer has not been defined

## E.4 Facility 4: Host Adapter Exceptions

---

These errors, detected by the Host Adapter, pertain to host communications errors or are detected while a Host Adapter Command Code is being processed. See Table E-4.

Table E-4 Facility 4: HOSTEXCEPT  
(Host Adapter—Host Exception Errors)

<b>Message Number (Hex)</b>	<b>Description</b>
00	Problem undefined on host
01	Lost DCD on host port transaction
02	Lost CTS on host port transaction
03	Timed out waiting on host transaction
04	Unrecognized command code
05	Invalid field received with command code
06	Host frame of excessive length
07	Host frame too short
08	Secondary not connected to network
09	Missed start of message delimiter
0A	Missed end of message delimiter
0B	Message length count error
0C	Message checksum bad
0D	Invalid hex-ASCII character received

## E.5 Facility 5: TIWAY Primitive Exceptions

---

These errors (shown in Table E-5) are detected by NIM-based attached devices while processing a TIWAY Primitive.

Table E-5 Facility 5: PRIMITIVE  
(TIWAY Primitive Errors)

<b>Message Number (Hex)</b>	<b>Description</b>
00	Primitive is not implemented
01	Data type (tt) is not defined on device
02	Data element location (nnnn) is out of range
03	Primitive has excess data unit bytes
04	Primitive has insufficient data unit bytes
05	Number of bytes received does not match length
06	Device in wrong mode for Primitive execution
07	User program has communications disabled
08	Written data type location (nnnn) did not verify
09	Data type location (nnnn) is write-protected
0A	Device fails to respond
0B	Primitive aborted due to a fatal error in device
0C	Invalid data type value due to Primitive execution
0D	An error was encountered while executing Primitive
0E	Primitive not valid for specified data type
0F	Data pattern requested was not found
10	Number of locations exceeds the maximum allowed
11	Data acquisition block number exceeds the maximum allowed
12	The block number requested has not been defined
13	The number of data bytes in requested block too large
14	Report by condition data type location too big
15	Primitive not allowed in local mode
16	Data type not allowed in specified device
17	Error in attached device communications
18	Data type not implemented in NIM, but is in device
19	Data element location out of range

## Facility 5: TIWAY Primitive Exceptions (continued)

---

Table E-5 Facility 5: PRIMITIVE  
(TIWAY Primitive Errors) (continued)

<b>Message Number (Hex)</b>	<b>Description</b>
1A	Attached device communications is not established
1B	Store and forward buffer full, message discarded
1C	Data element field improperly formatted
1D	Number of locations to access is zero
1E	Conflict with another device in write operation
1F	Unexpected Primitive while temporary RAM reserved
20	CC Command Code incorrect for current state of 33 Primitive
21	Non-contiguous data location while temporary RAM reserved
22	Command modifier not supported by device
23	Number of data blocks defined exceeds maximum supported
24	Illegal IEEE floating point value
25	Use extended structure format
26	Host identifier not valid
27	Communication buffer overflow
28	Broadcast transactions are not supported
29	Store and forward not supported with this Primitive
2A	Address value exceeds structure format (not extended)
2B	Write refused; device in upload or download mode
2C	Command field CC not supported
2D	Device is already in progress with upload or download
2E	Segment(s) requested not supported or defined
2F	Data count value error
30	Conflict with another device in write operation
31	Unexpected primitive while temporary RAM reserved
32	CC command code incorrect for current state of 33 primitive
33	Non-contiguous data location while temporary RAM reserved
DD	Attached device exception not identified
FF	Attached device exception not identified



## E.6 Facility 6: Operating System Status Exceptions

---

These are no facility 6 operating system errors in the PC version of the TIWAY Host Package.

## E.7 Facility 7: I/O Status Exceptions

---

These input/output (I/O) errors, reported by the device driver, are detected in setup or execution of I/O to the Host Adapter. See Table E-6.

Table E-6 Facility 7: I/O Status Error Listings

<b>Message Number (Hex)</b>	<b>Description</b>
01	I/O Status – Invalid function
02	I/O Status –Invalid port setup string
03	I/O Status – I/O not finished yet
04	I/O Status – User buffer too small
05	I/O Status – Fatal error, retry count exceeded

## E.8 Facility 8: CIM Exceptions

---

These are errors reported by CIM-based attached devices while processing a CIM Functional Command. See Table E-7.

Table E-7 Facility 8: CIM EXCEPT  
(CIM Functional Command Errors)

<b>Message Number (Hex)</b>	<b>Description</b>
50	Invalid memory specified
51	Invalid loop number specified
52	Loop not defined
53	Error number greater than 46 in clearing error number
54	Invalid command code or illegal CCU Task Code
55	Bad block table
56	Ignore this message
57	N not specified or bad data for read/write memory/IR
58	Ignore this message
59	Read after write check error
5A	Download error—invalid use, illegal character, not 8 character
5B	Upload error—invalid use, illegal character, not 8 character
5C	Attempt to send less than 2 or over 32 characters to CCU
5D	Not used
5E	Wrong amount of data with specified command function
5F	CCU communications error
60	Data link fatal error
64	Not enough data in download, expected 8+512 characters
65	L or C in ROM -- attempt to download to ROM
66	CCU in run state, invalid for download
67	Invalid download sequence
68	MM is incorrect for downloading
69	Non-ASCII character in input buffer
6A	Download active, other commands invalid
6C	Upload active, other commands invalid
6D	Invalid upload sequence
6E	MM is incorrect for upload
6F	CCU put non-ASCII characters in output buffer

## E.9 Facility 9: Native Task Codes

---

Because these error messages are specific to the type of attached device being used, Facility 9 errors are returned with a message number, but no text. You can determine the error by referring to the documentation on the attached device you are using.

Facility 9 includes more than just native task code errors. These are machine-specific, and for information regarding their meaning, please refer to the appropriate user manual for the machine in question (i.e., SIMATIC 530, PM550, etc.).

*Appendix F*

# Programmer's Notes for PC

---

F.1	General Information .....	F-2
F.2	Pascal .....	F-5
F.3	BASIC .....	F-6
F.4	C .....	F-7
F.5	Fortran .....	F-8
F.6	Linking Programs with the TIWAY Subroutine Library .....	F-9
F.7	Using Upload and Download from an Applications Program .....	F-10

## F.1 General Information

---

Before calling any TIWAY subroutines which use the `istat` variable, it is recommended that the `istat` variable be set to `-1`. This is an out-of-range value for the error status. After calling each TIWAY subroutine, the `istat` variable indicates the success or failure of the function. If `istat` is still equal to `-1`, the subroutine was abnormally aborted and did not complete.

The following information has been provided to aid you in learning how to invoke the TIWAY Subroutine Library from an application's program. The TIWAY Subroutine Library has been written in assembly language to provide speed, efficiency, and a simple interface to the five supported languages. In most cases, calling a TIWAY subroutine is no more difficult than calling a subroutine written in the same language. If you do encounter difficulties in interfacing your programs with the TIWAY Subroutine Library after reading the following notes, refer to the programmer's reference and the user guide for your application's language.

The Programmer's Notes for PC have been divided into eight sections: General Information, Pascal, BASIC, C, Fortran, Linking Programs with the TIWAY Subroutine Library, and Using Upload and Download from an Applications Program.

Example programs for some of the languages have been provided in  
FULLSAMP.C (a full C example that calls all library functions)  
Sample.C (and Sample2.C) (a small C example)  
Sample.PAS (Pascal examples)  
Sample.BAS (BASIC examples)

Example command procedures for compiling and linking these programs in C are provided in:  
MAKEC.BAT and  
SampleC.MAK

Other language compiling and linking information is documented in this manual, and additional information is located in `readme`.

---

The example programs and command files are all located in the subdirectory where you installed the TIWAY Host Software, and are accessible after installation of the TIWAY Host Software Package for personal computers. Each example program contains a call to subroutine(s) in the TIWAY Subroutine Library. FULLSAMP.C contains a call to each subroutine in the TIWAY subroutine library.

It is important to have a thorough understanding both of the language being used and of how it passes arguments. All arguments passed to TIWAY subroutines must be passed by reference. The one exception is the `tag` argument that can also be passed by descriptor. Most languages do pass integers and arrays that make up the majority of the TIWAY subroutine arguments by reference. Since character strings are seldom used, except for the `tag` argument, the default argument-passing mechanisms of most languages are usually sufficient. The one exception to this rule is C, which passes integers as values by default. For more information on calling TIWAY subroutines in C, refer to the Programmer's Notes below.

The argument names are unimportant. You can always create your own variable names when declaring arguments for the TIWAY Subroutine Library. The length of the data buffers is also unimportant as long as the buffers are large enough to hold the data received from or sent to the TIWAY subroutines. For example, you can use either the `ri4` argument or the `ri135` argument, depending on how much data is expected. To preserve storage, you may want to have two large buffers allocated for all TIWAY subroutine calls rather than having several arrays of different lengths.

The TIWAY Subroutine Library is written in assembly language, so there is no problem with types. The TIWAY Subroutine Library does not know the difference between integer and character arrays and so passes the data in the same manner, regardless of how the buffer is declared in the program.

The `S_BARY275` was introduced for strongly-typed languages so that you could use the same array for storing real, integer, and byte data. The `S_BARY275` type can be used to replace any type that requires 275 bytes. You can also declare your own similar types of varying lengths. Remember, when any type is used in an external subroutine declaration, you must use a variable declared of this type in all calls to that subroutine.

## General Information (continued)

---

Use `rsplen` and `rilen` arguments for response buffer lengths. These arguments must be initialized to the size of the response buffer prior to using them in a TIWAY subroutine call. The TIWAY subroutines use these arguments to determine whether the response buffer is large enough to hold the data returned from the TIWAY Network. This keeps other data in the program from unintentionally being corrupted. If the buffer is not large enough, or if the response length argument has not been initialized and contains a value of zero, the TIWAY subroutine returns an `istat` value of 0109 hex. The error message for facility 1, message 9, is **User supplied buffer too small for requested function.**

You must be familiar with how data is stored. Remember that PCs do not store data in the same way as do the secondaries or the Host Adapter. For example, in PCs, some data is stored in four bytes with the least significant byte first. Integers are stored by the secondaries and the Host Adapter in two bytes with the most significant byte first. Therefore, an array of bytes in Fortran containing an integer value of one is stored in PC format as shown:

```
X(1) = 01
X(2) = 00
X(3) = 00
X(4) = 00
```

The same data in Series 500/505 format is shown below.

```
X(1) = 00
X(2) = 01
```

The `TI2HST` and `HST2TI` support routines have been provided to handle these conversions for your convenience.



The SAMP\_PAS.PAS file and the SAMP\_PAS.TYP file contain both the forward and variable declarations and the type definitions required to call all subroutines in the TIWAY Subroutine Library. You can include these files in your programs to automatically make the necessary declarations for any TIWAY subroutine calls. You may find it more expedient, however, to make these declarations yourself if you wish to use only a few of the routines in the TIWAY Subroutine Library.

Pascal's most outstanding features are that it is a strongly-typed language and is extremely sensitive to any mode conflicts. For example, something of type S\_BARY275 cannot be passed to a Pascal procedure that expects something of type BARY275, although the two types have the same memory allocation. Pascal requires external routine declarations and the types assigned in those declarations are the only types that can be used in the procedure call.

Pascal records with the case option should be declared as volatile and have no case qualifier. The case qualifier requires some memory allocation in the starting address of any variable declared of this type. The TIWAY Subroutine Library is not expecting this and results from such a call are either wrong or may cause an error to be generated.

The SAMPLE.BAS file and the DEF.BAS file contain both the forward and variable declarations and the type definitions required to call all subroutines in the TIWAY Subroutine Library. These files can be included in your programs to make the necessary declarations for any TIWAY subroutine calls automatically. You may find it more expedient, however, to make these declarations yourself if you wish to use only a few of the routines in the TIWAY Subroutine Library.

In BASIC, array element 0 is generally not used by the programmer, but for TIWAY subroutines this element is used as the first element of the array.

All TIWAY subroutines must be called in BASIC with the BY REF qualifier. External subroutine declarations for TIWAY subroutines must also use this qualifier.

The SAMPLE.C, SAMPLE2.C, and SAMPC.TYP files contain both the variable declarations and the type definitions required to call all subroutines in the TIWAY Subroutine Library. These files can be included in your programs to automatically make the necessary declarations for any TIWAY subroutine calls. If you wish to use only a few of the routines in the TIWAY Subroutine Library, you may find it more expedient to make these declarations yourself.

C automatically passes arrays by reference, but integer values and typedefs of unions or structures are not automatically passed by reference. A structure, union, or integer must have a preceding ampersand when calling a routine from the TIWAY Subroutine Library. For example, a call to the TIWAY subroutine POLL is made as follows:

```
POLL ( &ISTAT, &XTN, &TAG, &RSPLEN, &RSP275 )
```

The variable `tag` is declared of type `TAG_TYPE`. `TAG_TYPE` is a typedef union. The variables `istat`, `xtn` and `rsplen` are integers. The variable `rsp275` is of type `s_bary275` which is also a typedef union.

TIWAY subroutine calls from Fortran are very simple and straightforward. If you have problems calling TIWAY subroutines from Fortran, try compiling the program with the /NOOPTIMIZE qualifier. Compiling Fortran programs with the optimizer has caused problems due to errors in the Fortran compiler.

## F.6 Linking Programs with the TIWAY Subroutine Library

---

For more information on linking your application with the TIWAY Host Library, refer to Section 1.6 for additional link options. You may also need to refer to the user guide for the language you are using.

## F.7 Using Upload and Download from an Applications Program

---

If you wish to use the upload and download facilities found in TIUSER with a program, you may link these facilities with an application program as follows:

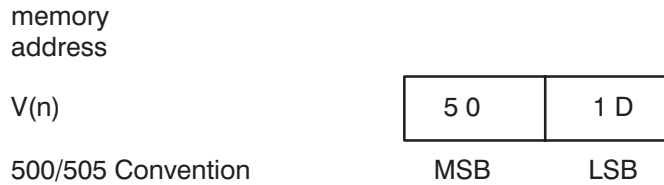
```
$ LINK Your_program,UPLOAD,TIFUNC,TISUBM,TISUBF,TIBDAT  
$ LINK Your_program,DNLOAD,TIFUNC,TISUBM,TISUBF,TIBDAT
```

# 500/505 and Host Computer Data

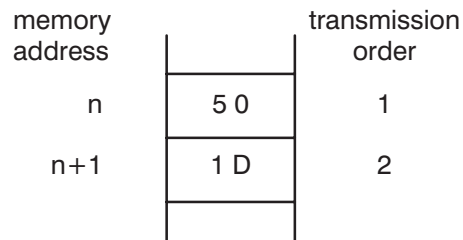
---

There is a significant difference in the way data is stored within 500/505 Secondaries and the way data can be stored within IBM-compatible computers. It is important that you understand these differences and how the TIWAY Host Software Package treats them.

This format is best described by example. The hexadecimal value 501D is stored within PM550 V-memory as follows:



The transmission order across the TIWAY network is MSB followed by LSB. In our example, 50 is followed by 1D. That value is stored in a computer in the order received:



---

However, assuming this value was defined within an applications program as type WORD\_INTEGER (two-byte integer), PC data formats define the first byte as the LSB and the second as MSB. This is opposite of the true value. 501D hex becomes 1D50-byte swapping.

memory address		PC convention	transmission order
n	5 0	LSB	1
n+1	1 D	MSB	2

The TIWAY Host Software Package has two mechanisms available to resolve this conflict.

When reading data from a 500/505 Secondary, if conversion from 500/505 format to PC format is required, then use the TIGET routine. TIGET reads the value 501D hex from the PM550 or other 500/505 controller and performs the necessary byte swap. In the example above, TIGET stores 50 hex, received first, in the MSB of the WORD\_INTEGER and stores 1D hex, received second, in the LSB. The memory appears as:

memory address		PC convention	transmission order
n	1 D	LSB	2
n+1	5 0	MSB	1



---

The original value is preserved.  $V(n) = 501D$  hex is stored as 501D in the PC computer. If you do not wish conversion to take place, then use the TIREAD routine. The result of the TIREAD operation is as originally shown:

memory address		PC convention	transmission order
n	5 0	LSB	1
n+1	1 D	MSB	2

The  $V(n)$  value is stored in the PC computer as 1D50 hex.

The corollary to TIGET (writing to a Secondary from the PC computer) is the TIPUT routine, while the corollary to the TIREAD routine is TIWRIT. In other words, TIPUT transmits our example 501D in MSB/LSB order:

memory address		PC convention	transmission order
n	1 D	LSB	2
n+1	5 0	MSB	1

The value is stored in  $V(n)$  as 501D hex.

---

The TIWRIT routine, on the other hand, transmits in the “as is” order:

memory address		PC convention	transmission order
n	1 D	LSB	1
n+1	5 0	MSB	2

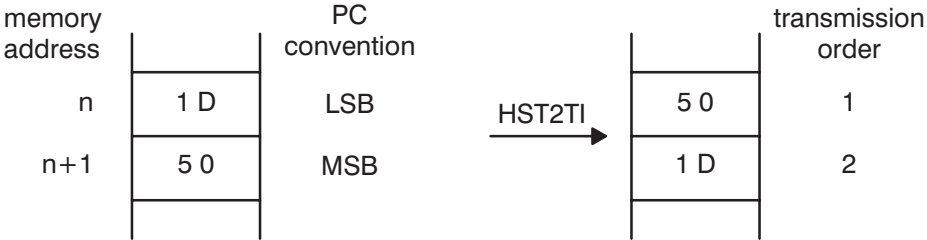
In this last example, 501D hex is stored in the PM550 as 1D50 hex.

Use the HST2TI and TI2HST routines with data manipulation routines that do not automatically perform the data conversion. The TI2HST routine reads our example 501D hex value from the PM550 in MSB/LSB order and writes into successive memory in the PC computer in LSB/MSB order.



---

The HST2TI routine is the corollary to TI2HST for writing data to a 500/505 Secondary.



# Unilink Dipswitch Reference Card

### Dipswitch Settings For NITP Protocol

**DIPSWITCH 1**

**DIPSWITCH 2**

**KEY**

UP (1)    DOWN (0)

**Run Time LED Display**

Adapter Good    Online    Receive    Transmit

**LED Error Codes**

Error Code	1	2	3	4	5	6
No Error (Run Mode)	●	○	○	○	○	○
No Error (Diagnostic Mode)	○	○	○	○	○	○
Diagnostics ROM Checksum Error	○	○	○	○	○	●
RAM Test Error	○	○	○	○	●	○
TIWAY Port 1 External Loopback Error	○	○	○	○	●	●
TIWAY Port 2 External Loopback Error	○	○	○	○	○	○
Host Port (Port 3) Loopback Error	○	○	○	○	○	○
Port 4 Loopback Error	○	○	○	○	○	○
TIWAY Serial Communications Controller Failure	○	○	○	○	○	○
PIM Checksum Error	○	○	○	○	○	○
CPU Failure	○	○	○	○	○	○
-5 Volt Power Supply Failure	○	○	○	○	○	○
Watchdog Timer Failure	○	○	○	○	○	○
External I/O Loopback Error	○	○	○	○	○	○
Network Address Error	○	○	○	○	○	○
ROM Checksum Error	○	○	○	○	○	○

Legend    ○ - Flashing    ● - On (Lit)    ○ - Off (Extinguished)

**Baud Rate Table**

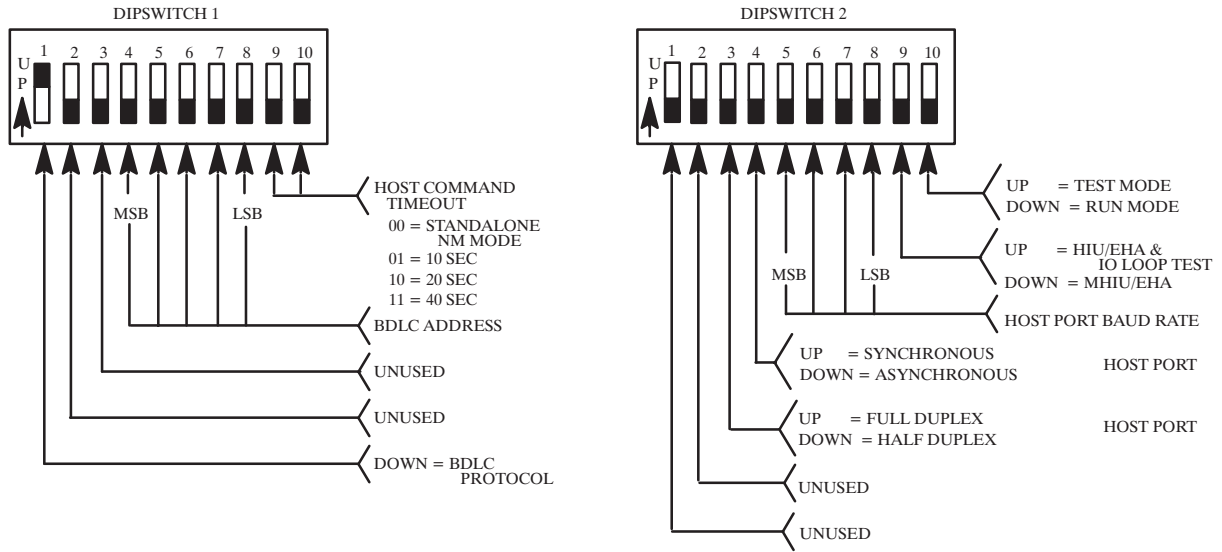
SWITCHES				BAUD RATE
MSB		LSB		
0	1	0	0	110
0	1	0	1	150
0	1	1	0	300
0	1	1	1	600
1	0	0	0	1200
1	0	0	1	2400
1	0	1	0	4800
1	0	1	1	9600
1	1	0	0	19200
1	1	0	1	38400
1	1	1	0	57600*
1	1	1	1	115200*

0 = DOWN    1 = UP  
\* THESE VALUES ARE NOT AVAILABLE ON THE HOST PORT

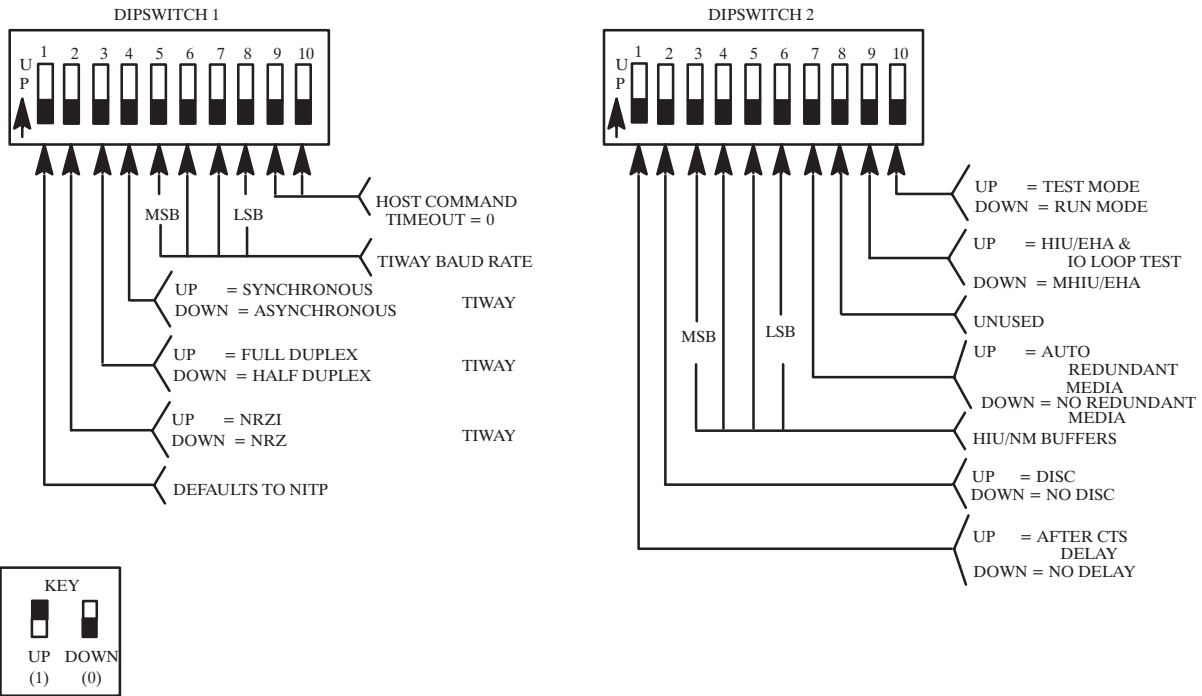
**Host Command Time-out Table**

DIPSWITCH 1		
Position		Description
9	10	
Down	Down	0 sec - Accepts No Host Commands (SANM Mode)
Down	Up	10 sec Timeout
Up	Down	20 sec Timeout
Up	Up	40 sec Timeout

### For BDLC Protocol Dipswitch Settings



### Dipswitch Settings For Configuring A Stand-alone Network Manager



## A

Activate, ACTVAT, 4-8, 7-5, 7-21, 10-4

ACTVAT, 4-8, 10-4

Adapter diagnostics, ADIAG, 4-8, 7-22, 10-5–10-6

### Addressing

arguments, 3-18

ASCII specification, 3-18

binary specification, 3-19

data elements, 3-17

highways, 3-17

secondaries, 3-17

tag name specification, 3-18

ADIAG, 10-5–10-6

### Arguments

addressing, 3-6

buffers

command, 3-21

read, 3-22

response, 3-21

write, 3-22

data manipulation, 3-23

diagnostics, 3-25

list of subroutine, 3-6

masks, 3-26

network access, 3-27

status, 3-28

Asynchronous, 3-27

## B

BLDMSK, 10-7

BRDCST, 10-8–10-9

Broadcast, BRDCST, 4-8, 7-23, 10-8–10-9

### Buffers

command, 3-21

read, 3-22

response, 3-21

write, 3-22

Build mask, BLDMSK, 10-7

## C

C, declarations, 3-13

C declarations, list of, 3-13

CCUSTS, 4-15–4-16, 10-10

Change state, CHNGST, 4-12, 7-6, 7-26, 10-11

CHNGST, 4-12, 10-11

CIM Functional Commands, 7-32–7-37, C-1–C-4

CCUSTS, 4-15–4-16, 10-10

CIMDNL, 4-15–4-16, 10-12–10-13

CIMRD, 4-15–4-16, 10-14

CIMUPL, 4-15–4-16, 10-15–10-16

CIMWR, 4-15–4-16, 10-16

RDLOOP, 4-15–4-16, 10-37

RNDRD1, 4-15–4-16, 10-40–10-41

RNDRD2, 4-17, 10-41–10-42

RNDRD3, 4-17, 10-42–10-43

RNDRD4, 4-17, 10-44–10-45

TIPROG Commands

CC, 7-32

CR, 7-32

CW, 7-33

R1, 7-34

R2, 7-34

R3, 7-35

R4, 7-36

RL, 7-33

CIMDNL, 4-15, 10-12–10-13

CIMRD, 4-15–4-16, 10-14

CIMUPL, 4-15–4-16, 10-15–10-16

CIMWR, 4-15–4-16, 10-16

Command codes, host adapter, 2-3

Compiling, 1-12

Computer

host, 2-2, 2-3

supported languages, 2-5

CONFIG, 4-12, 10-17–10-18

Configuration, CONFIG, 7-27, 10-17–10-18

Configuring Unilink, 9-2

Customer support, 1-3

---

## D

Data acquisition, 2-2  
Data element types, 3-14–3-16  
  list of, 3-14–3-16  
Data format conversion, 2-5  
DEACT, 4-9, 10-18–10-19  
Deactivate, DEACT, 4-9, 7-7, 7-23, 10-18–10-19  
Declarations, C, 3-13  
DEFBLK, 4-12, 10-20–10-21  
Define block, DEFBLK, 4-12, 7-27, 10-20–10-21  
DNLOAD, 10-22–10-23  
Download, DNLOAD, 7-8, 10-22–10-23

## E

Error messages, E-2–E-16  
Exception messages, E-2–E-16

## F

Facility codes, E-2–E-16  
Files, description, 1-4  
FILL, 4-12, 7-28, 10-24  
FIN, 4-5, 7-20, 10-25

## G

GATHER, 4-12, 7-28, 10-26–10-27  
  
Gather  
  blocks using mask, 10-26–10-27  
  blocks with mask, 10-58–10-59  
Get length, GETLEN, 4-13, 7-28, 10-27  
GETLEN, 4-13, 10-27  
GETMSG, 10-28

Index-2

## H

Hardware requirements, 1-2  
Help, technical support, 1-3  
Host adapter, 2-3  
  commands, A-1–A-3  
  network activity statistics, 10-5  
Host-to-TI, HST2TI, 7-37, 10-29  
HST2TI, 7-37, 10-29

## I

INIT, 4-5, 7-20, 10-30  
Initialization subroutine, INIT, 4-5, 10-30  
Installing TIWAY package, 1-8  
istat, 3-28

## L

Languages supported, 2-5  
Linking, 1-13  
LKUFMT, 10-31–10-32  
LKUTGL, 10-32  
LKUTGS, 10-33  
Local area network, 2-2  
Local line, 2-2  
Look up format, LKUFMT, 7-37, 10-31–10-32  
Look up tag  
  LKUTGL, 10-32  
  LKUTGL, long format, 7-38  
  LKUTGS, 10-33  
  LKUTGS, short format, 7-38

## N

NATIVE, 4-13, 7-29, 10-34–10-35  
NETAUTOC.DAT, 9-2  
Network  
  characteristics, 2-2  
  communication levels, 2-3

---

## P

PM550 CCU Task Codes, D-1–D-17  
POLL, 4-9, 7-24, 10-35  
Primitives, TIWAY, B-1  
Program mode. *See* CHNGST  
PUTMSG, 10-36

## R

RDLOOP, 4-15–4-16, 10-37  
RDSTS, 10-38–10-39  
Read loop, RDLOOP, 4-15 – 4-17, 10-37  
Read status, RDSTS, 4-13, 7-29  
Requirements  
  hardware, 1-2  
  software, 1-3  
RNDRD1, 4-15–4-16, 10-40–10-41  
RNDRD2, 4-15, 4-17, 10-41–10-42  
RNDRD3, 4-15, 4-17, 10-42–10-43  
RNDRD4, 4-15, 4-17, 10-44–10-45  
Run mode. *See* CHNGST

## S

SDIAG, 10-45  
SECLOG, 10-46  
Secondary  
  data format, 2-5  
  definition, 2-2  
Secondary diagnostics, SDIAG, 4-9, 7-24, 10-45  
Secondary log, SECLOG, 4-10, 7-25, 10-46  
Software requirements, 1-3  
sstat, 3-30  
Statistics, host adapter, network activity, 10-5  
Status determination, 3-28  
Subroutine  
  ACTVAT, 4-8, 10-4  
  ADIAG, 4-8, 10-5–10-6

arguments, 3-6  
BLDMSK, 10-7  
BRDCST, 4-8, 10-8–10-9  
CCUSTS, 4-15–4-16, 10-10  
CHNGST, 4-12, 10-11  
CIMDNL, 4-15–4-16, 10-12–10-13  
CIMRD, 4-15–4-16, 10-14  
CIMUPL, 4-15–4-16, 10-15–10-16  
CIMWR, 4-15–4-16, 10-16  
CONFIG, 4-12, 10-17–10-18  
DEACT, 4-9, 10-18–10-19  
DEFBLK, 4-12, 10-20–10-21  
DNLOAD, 10-22–10-23  
FILL, 4-12, 10-24  
FIN, 4-5, 10-25  
GATHER, 4-12, 10-26–10-27  
GETLEN, 4-13, 10-27  
GETMSG, 10-28  
HST2TI, 10-29  
INIT, 4-5, 10-30  
LKUFMT, 10-31  
LKUTGL, 10-32  
LKUTGS, 10-33  
NATIVE, 4-13, 10-34–10-35  
POLL, 4-9, 10-35  
PUTMSG, 10-36  
RDLOOP, 4-15–4-16, 10-37  
RDSTS, 4-13, 10-38–10-39  
RNDRD1, 4-15–4-16, 10-40–10-41  
RNDRD2, 4-17, 10-41–10-42  
RNDRD3, 4-17, 10-42–10-43  
RNDRD4, 4-17, 10-44–10-45  
SDIAG, 4-9, 10-45  
SECLOG, 4-10, 10-46  
TI2HST, 10-47  
TIGET, 4-13, 10-48–10-49  
TIPUT, 4-13, 10-50  
TIREAD, 4-14, 10-51  
TIWRIT, 4-14, 10-52  
TIXTN, 10-53–10-54  
TIXTNW, 10-53–10-54  
UPLOAD, 10-54–10-55  
WRBUF, 4-14, 10-56–10-57  
WRTGAT, 4-14, 10-58–10-59  
XPAR, 4-10, 10-60–10-61

Subroutines and arguments, reference list,  
  3-3–3-13

Switches, Unilink dipswitches, H-1

Synchronous, 3-27



---

## T

Tag names, defining, 8-2

Tag table,  
  accessing, 1-26  
  building, 8-3  
  creating, 1-25

Technical support, 1-3

TI-to-host, TI2HST, 7-39, 10-47

TI2HST, 10-47

TIGET, 4-13, 7-30, 10-48–10-49

TIPROG, 7-16–7-39

TIPROG Commands

  CIM Commands

    CC, 7-32

    CR, 7-32

    CW, 7-33

    R1, 7-34

    R2, 7-34

    R3, 7-35

    R4, 7-36

    RL, 7-33

  Support Commands

    HT, 7-37

    LF, 7-37

    LL, 7-38

    LS, 7-38

    TH, 7-39

  TIWAY Primitive Commands

    CH, 7-26

    CN, 7-27

    DE, 7-27

    FL, 7-28

    GA, 7-28

    GL, 7-28

    NA, 7-29

    RS, 7-29

    TG, 7-30

    TP, 7-30

    TR, 7-30

    TW, 7-31

    WB, 7-31

    WG, 7-32

TIPUT, 4-13, 7-30, 10-50

TIREAD, 4-14, 7-30, 10-51

TIUSER, 7-3–7-15

TIUSER Commands

  ACTIVATE, 7-3–7-4, 7-5

  CHANGESTATE, 7-3–7-4, 7-6

  DEACTIVATE, 7-3–7-4, 7-7

  DOWNLOAD, 7-3–7-4, 7-8–7-10

  FILESTAT, 7-3–7-4, 7-11

  LIST, 7-3–7-4, 7-11

  QUIT, 7-3–7-4, 7-15

  STATISTICS, 7-3–7-4, 7-11

  UPLOAD, 7-3–7-4, 7-12–7-14

  VERIFY, 7-3–7-4, 7-15

TIWAY primitives, B-1–B-3

TIWRIT, 4-14, 7-31, 10-52

TIXTN, 10-53–10-54

TIXTNW, 10-53–10-54

## U

Unilink

  adapters, 2-2–2-3

  dipswitches, H-1

  host adapter, 2-3

UPLOAD, 7-12, 10-54–10-55

## W

WRBUF, 4-14, 10-56–10-57

Write

  blocks with mask, 10-58–10-59

  buffered, 10-56–10-57

Write and gather, WRTGAT, 7-32

Write buffer, WRBUF, 7-31, 10-56–10-57

WRONG INITed STATE, 7-20

WRTGAT, 4-14, 10-58–10-59

## X

XPAR, 4-10, 7-25, 10-60–10-61

# Customer Response

---

We would like to know what you think about our user manuals so that we can serve you better.  
How would you rate the quality of our manuals?

	Excellent	Good	Fair	Poor
Accuracy	_____	_____	_____	_____
Organization	_____	_____	_____	_____
Clarity	_____	_____	_____	_____
Completeness	_____	_____	_____	_____
Graphics	_____	_____	_____	_____
Examples	_____	_____	_____	_____
Overall design	_____	_____	_____	_____
Size	_____	_____	_____	_____
Index	_____	_____	_____	_____

Would you be interested in giving us more detailed comments about our manuals?

- Yes!** Please send me a questionnaire.
- No.** Thanks anyway.

Your Name: \_\_\_\_\_

Title: \_\_\_\_\_

Telephone Number: (\_\_\_\_) \_\_\_\_\_

Company Name: \_\_\_\_\_

Company Address: \_\_\_\_\_

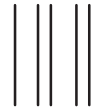
\_\_\_\_\_

\_\_\_\_\_

**Manual Name:** SIMATIC TIWAY Host Software for PC User Manual  
**Manual Assembly Number:** 2804790-0001  
**Order Number:** PPX:TIWAY-8108-3

**Edition:** Third  
**Date:** 05/96

FOLD



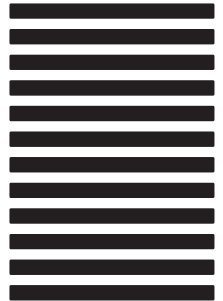
NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO.3 JOHNSON CITY, TN

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN: Technical Communications M/S 519  
SIEMENS ENERGY & AUTOMATION INC.  
3000 BILL GARLAND RD  
P O BOX 1255  
JOHNSON CITY TN 37605-1255



FOLD

SIMATIC is a registered trademark of Siemens AG.

5TI, UNILINK, TIWAY, PM550, Series 500, and Series 505 are trademarks of Siemens Energy & Automation, Inc.

MS-DOS, MS-FORTRAN, and Microsoft C are trademarks of Microsoft Corporation.

IBM, IBM PC/AT, and IBM PC/XT are registered trademarks of International Business Machines.

TI, Texas Instruments, and TIPC are trademarks of Texas Instruments, Incorporated.