BRI

Basic Reader Interface

Programmer Reference Manual

Intermec Technologies Corporation

Worldwide Headquarters
6001 36th Ave.W.
Everett, WA 98203
U.S.A.

www.intermec.com

Document Change Record

This page records changes to this document. The document was originally released as version -001.

Version Number	Date	Description of Change
012	7/2013	Removed unsupported propietary commands.
011	11/2012	Updated to support BRI Version 3.17 of the BRI Spec: • Added the PRESET command • Added the EPCC1G2 Select Logic data condition • Added the QUERYSEL attribute • Added the QUERYTARGET attribute
010	7/2012	Updated to support BRI Version 3.16 of the BRI Spec. • Added flex_query_selector to all appropriate commands • Added Generic Tag Access command
009	1/2011	Updated to support BRI Version 3.14 of the BRI Spec. • Added “attribute antennas and “cap antenna” syntax to support bistatic antenna operations for the IF2 Network Reader • Added RNSI, SNR, and CHANNEL keywords • Added BLOCKPERMALOCK command • Added NXP, Fujitsu, and Impinj tag extensions
008	10/2009	Manual was revised to support the new ETSI standard for RF products.
007	7/2009	Updated to support BRI Version 3.14 of the BRI Spec. • Added two new battery events • Updated KILL password description
006	4/2009	Updated to support BRI Version 3.12 of the BRI Spec. • Updated keywords reserved for command parameters • Updated keywords reserved for reader attributes • Added DIAGNOSTICS command • Added SESSIONS attribute • Added BRI command aliases
005	9/2008	Updated to support BRI Version 3.11 of the BRI Spec. • Added NXPREADPROTECT, NXPEAS, NXPALARM support • Updated CAPABILITIES for NXP support • Added the EPCC1G1 tagtype • Increased TRIGGER support

Version Number	Date	Description of Change
004	05/2008	Updated to support BRI Version 3.00, 3.01, and 3.10 of the BRI Spec. Note that any commands, attributes, or functions introduced in newer BRI versions will not work on older BRI versions. Revisions made in BRI Version 3.00 include: • Power-up sequence and BOOT macro removed. • RESET, FACDFLT, and VERSION commands updated. • LBTSCANENABLE and LBTCHANNEL updated. • READ, WRITE, LOCK, KILLTAG, and PROTECT commands update with inline TAGTYPE support. • Added PLATDFLT and HWCC command. • Added EVT:RESET to the list of event messages. • TRIGGER ACTION enhancements documented. • CTI command removed. Revisions made in BRI Version 3.01 include: • Added BTPWROFF attribute. Revisions made in BRI Version 3.10 include: • Added RSSI data field. • Added CAPABILITIES and ERASE command. • Added WRITEGPIO single-pin definition. • Removed UTIL SINGULATE, IDENTIFY, and TAGTYPE. • Added data field requirement to WRITE.
003	09/2006	Updated to support the BRI Version S: • Added support for access passwords. • Introduced support for ISO 18000-6C UIIs. • Added LOCK command and updated it for use only with ISO 18000-6C UII tags. • Updated WRITE command to include LOCK functions. • Added KILLTAG, VERSION, and RESET commands. • Added an action macro to the TRIGGER command. • Introduced BIT data type. • Added PROTECT command. • Updated PING command to include TIMESTAMP. • Updated READ command so that it is possible to get TAGTYPE as part of the reported data returned. • Updated the FIELDSTRENGTH attribute to include dB values. • Introduced EPCglobal C1ass 1Gen 1 as a valid TAGTYPE and generally updated the TAGTYPE command. • Introduced HWID, HWPROD and HWREGION commands.
002	04/2006	Updated to support the BRI Version R.

Contents

Before You Begin xi
Safety Information xi
Global Services and Support xi
Warranty Information xi
Web Support xi
Send Feedback xi
Telephone Support xii
Who Should Read This Manual xii
Related Documents xii

1 Introducing the Basic Reader Interface 1

Overview of the Basic Reader Interface 2
What's New? 2
General Features of the BRI Architecture 2

Two Typical BRI Usage Scenarios 2

BRI TCP Applications 3
Choosing a TCP Port for an Application Using TCP 3

Resetting the Reader to the Factory Default Configuration 3

Identifying the Version of BRI on Your Reader 4

Conventions Used in This Manual 5

2 Designing Robust BRI Applications 7

Before You Begin Programming 8
Managing RFID Configuration 8
Using the RFID Resource Kit 9

Recommended Software Structure 9

Programming the BRI Message Layer 11
BRI Message Types 11
BRI Command Processor 12
BRI Asynchronous Event Handler 12
Response Handler 12

Programming the BRI Transport Layer 12
Transport Operation 12
Transport Initialization 13

Multi-Threaded Implementation 13

3 Understanding BRI Programming Elements 15

BRI Logical Interface 16

Using Message Checksums 16

Reserved Keywords 17
- Keywords Reserved for BRI Commands 17
- Keywords Reserved for Command Parameters 18
- Keywords Reserved for Reader Attributes 20
- Keywords Reserved for Reader Error and Success Responses 21

Constants 22

Data Field Definitions 22
- ANTENNA 22
- AFI 23
- BIT(memory_bank:startbit, bits) 23
- COUNT 23
- EPCID 23
- HEX(*memory_bank:address, length*) 24
- INT(*memory_bank:address, length*) 25
- ISOUII 26
- PC 27
- RSSI 27
- STRING(*memory_bank:address, length*) 27
- TAGTYPE 28
- TIME 28

Data Conditions 28
- Using Native Tag Selector Logic in Data Conditions 29
- Using AND/OR Logic in Data Conditions 30
  - Using the AND and OR Keywords 31
  - Grouping Expressions Without Using Parentheses 32
- Using EPCC1G2 Select Logic in Data Conditions 33
- Multi-Protocol Condition Usage 34

4 BRI Commands 35

BRI Commands 36
- ATTRIBUTE 36
  - Changing the Reader Attributes 36
  - Reading the Reader Attributes 38
- BLOCKPERMALOCK 38
- BLOCKPERMALOCK READ 40
- BRIVER 40
- CAPABILITIES 41
- DIAGNOSTICS 44
- ERASE 45
- FACDFLT 45
- FLEXQUERY 47

HELP 47
HWCC 48
HWID 48
HWPROD 48
HWREGION 48
HWVER 49
KILLTAG 50
LOCK 50
PING 51
PLATDFLT 51
PRESET 52
PRINT 52
PROTECT 52
READ 54
READGPI 60
REPEAT 60
RESET 61
SET 62
SWVER 63
TRIGGER 63
TRIGGERQUEUE 66
TRIGGERREADY 67
TRIGGERWAIT 68
VERSION 68
WRITE 69
WRITEGPO 73

BRI Extensions for NXP Tags 74
NXPALARM 74
NXPCONFIG 75
NXPEAS 75
NXPREADPROTECT 76

BRI Extensions for Fujitsu Tags 76
FJAREAREADLOCK 76
FJAREAWRITELOCK 76
FJAREAWRITELOCKWOPWD 77
FJBURSTERASE 77
FJBURSTWRITE 77
FJCHGAREAGROUPPWD 78
FJCHGBLOCKGROUPPWD 78
FJCHGBLOCKLOCK 79
FJCHGWORDLOCK 79
FJREADBLOCKLOCK 79

BRI Extensions for Impinj Monza 4 Tags 80
READ IMPINJQT 80
WRITE IMPINJQT 80

BRI Extensions for EM Microelectronics Tags 81
READ EMM_GETUID 81
READ EMM_GETSENSOR 81
WRITE EMM_SENDSPI 82

WRITE EMM_RESETALARMS 82

Generic Tag Access Command 82
ACCESSCMD 83

Understanding [READ FIELD] and [WRITE FIELD] Parameters 85
[READ FIELD] Examples 85
[WRITE FIELD] Examples 85

Understanding the <ATTRIBUTE NAME> Parameter 86
ANTENNAS or ANTS 86
ANTTIMEOUT 87
ANTTRIES 87
BAUD 87
BROADCASTSYNC 88
BTPWROFF 88
CHANNEL 88
CHKSUM 88
DENSEREADERMODE 89
ECHO 89
EPCC1G2PARAMETERS 89
FIELDSEP 89
FIELDSTRENGTH 90
IDREPORT 91
IDTIMEOUT 91
IDTRIES 92
INITIALQ 92
INITTRIES 92
LBTCHANNEL 92
LBTSCANENABLE 93
LOCKTRIES 93
NOTAGRPT 93
QUERYSEL 94
QUERYTARGET 94
RDTRIES 94
RPTTIMEOUT 95
SCHEDULEOPT 95
SELTRIES 96
SESSION 96
TAGTYPE 96
TIMEOUTMODE 97
TTY 97
UNSELTRIES 98
UTCTIME 98
XONXOFF 98
WRTRIES 99

Understanding the Timeouts and Tries 99
Setting IDTIMEOUT and ANTTIMEOUT 99
When IDTIMEOUT <= ANTTIMEOUT 99
When IDTIMEOUT > ANTTIMEOUT 100
Setting IDTRIES and ANTTRIES 100
When IDTRIES < ANTTRIES 100

When IDTRIES >= ANTTRIES 101
Setting INITTRIES 102

Understanding the [LITERAL] Parameter 103

Reading and Writing STRING Fields 103

Understanding ACCESS and KILL Passwords 104
Memory Bank 3 User Memory 105
Memory Bank 2 TID Memory 106
Memory Bank 1 EPCID Memory 107
Memory Bank 0 PASSWORD Memory 108

Understanding Error and Success Responses 109

Understanding EVENT Messages 110

Understanding the Format of BRI Command Responses 112

Creating and Using BRI Macros 113
Creating a Command Macro 114
Executing a Command Macro 114
Creating a Parameter Macro 114
Executing a Command With a Parameter Macro 115
Listing All Macros Stored in Memory 115
Displaying the Contents of a Macro 115
Deleting a Macro 115

5 Reader-Specific Platform Specifications 117

Reader-Specific Platform Specifications 118

ITRF*xxx*01 Readers 118
BRI Disabled by Default 118
Features Not Implemented 118
Buffer Sizes 118
Memory Management 119
Error Responses 119
Antennas 119
GPIO 119
Reader Attributes 119
BRI Commands 120
EVENT Responses 120
Phillips V1.19 TAG Type 120

Readers That Contain the IM5 Module 120
Buffer Sizes 120
Antennas 121
GPIO 121
Reader Attributes 121
FACDFLT Command 121

Readers That Contain the IM4 Module 121

WRITEGPI Command. 121
READGPI Command. 122
TRIGGER Command 122
Antennas 122
ANTENNA Data Type Definition 122
ANTENNAS or ANTS Attribute 123
Sensing an Over-Temperature Condition 123

Index 125

Before You Begin

This section provides you with safety information, technical support information, and sources for additional product information.

Safety Information

Your safety is extremely important. Read and follow all warnings and cautions in this document before handling and operating Intermec equipment. You can be seriously injured, and equipment and data can be damaged if you do not follow the safety warnings and cautions.

This section explains how to identify and understand cautions and notes that are in this document.

A caution alerts you to an operating procedure, practice, condition, or statement that must be strictly observed to prevent equipment damage or destruction, or corruption or loss of data.

Note: Notes either provide extra information about a topic or contain special instructions for handling a particular condition or set of circumstances.

Global Services and Support

Warranty Information

To understand the warranty for your Intermec product, visit the Intermec website at **www.intermec.com** and click **Support** > **Returns and Repairs** > **Warranty**.

Disclaimer of warranties: The sample code included in this document is presented for reference only. The code does not necessarily represent complete, tested programs. The code is provided "as is with all faults." All warranties are expressly disclaimed, including the implied warranties of merchantability and fitness for a particular purpose.

Web Support

Visit the Intermec website at **www.intermec.com** to download our current manuals (in PDF).

Visit the Intermec technical knowledge base (Knowledge Central) at **intermec.custhelp.com** to review technical information or to request technical support for your Intermec product.

Send Feedback

Your feedback is crucial to the continual improvement of our documentation. To provide feedback about this manual, please contact the Intermec Technical Communications department directly at **TechnicalCommunications@intermec.com**.

Telephone Support

In the U.S.A. and Canada, call **1-800-755-5505**.

Outside the U.S.A. and Canada, contact your local Intermec representative. To search for your local representative, from the Intermec website, click **About Us** > **Contact Us**.

Service Location Support

For the most current listing of service locations, click **Support** > **Returns and Repairs** > **Repair Locations**.

Who Should Read This Manual

This programmer reference manual is for the person who is responsible for using Basic Reader Interface (BRI) commands to manage RFID readers. This manual describes BRI commands and programming concepts.

Before you work with the BRI, you should be familiar with your RFID reader, RFID system, and RFID concepts such as tag types.

Related Documents

The Intermec website at **www.intermec.com** contains our documents (as PDF files) that you can download for free.

To download documents

1. Visit the Intermec website at **www.intermec.com**.
2. Click the **Products** tab.
3. Using the **Products** menu, navigate to your product page. For example, to find the CN3 computer product page, click **Computers** > **Handheld Computers** > **CN3**.
4. Click the **Manuals** tab.

If your product does not have its own product page, click **Support** > **Manuals**. Use the **Product Category** field, the **Product Family** field, and the **Product** field to help you locate the documentation for your product.

Here is a list of third-party documents that you may find useful. The first two are available on the EPCglobal Inc. website at **www.epcglobalinc.org.** The third is a Phillips document.

- *EPC™ Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz – 960 MHz Version 1.2.0*
- *EPC™ Tag Data Standards Version 1.3 Implementation of EPC Tag Data on U-Code EPC 1.19 Version 1.0 June 2004*
- *Implementation of EPC Tag Data on U-Code EPC 1.19 Version 1.0 June 2004*

1

Introducing the Basic Reader Interface

This chapter introduces the Basic Reader Interface (BRI). This chapter contains these sections:

- **Overview of the Basic Reader Interface**
- **Two Typical BRI Usage Scenarios**
- **BRI TCP Applications**
- **Resetting the Reader to the Factory Default Configuration**
- **Identifying the Version of BRI on Your Reader**
- **Conventions Used in This Manual**

Overview of the Basic Reader Interface

This programmer reference manual defines the architecture of the Basic Reader Interface (BRI) intended for use with Intermec RFID readers. The BRI is an RFID scripting language developed to improve productivity for an RFID programmer and reduce the time required to develop RFID applications. RFID programmers can use the BRI protocol to write programs that communicate with Intermec RFID devices. With the BRI, programmers can configure RFID control parameters, define event triggers, and perform tag operations.

What's New?

This programmer reference manual supports the Basic Reader Interface Version 3.17, and was updated to remove unsupported proprietary commands.

General Features of the BRI Architecture

The BRI architecture meets these design criteria:

- Provides an ASCII-only command set. The commands are human-readable.
- Provides a simple command/reply operations, and asynchronous event messages.
- Allows presentation on a variety of physical interfaces such as serial, Bluetooth, and Ethernet.
- Allows configuration of the general purpose input and output (GPIO) interface of a reader device.
- Contains a command set that is flexible and can expand to accommodate specific reader variations and allow for future expansion and reader features.

Two Typical BRI Usage Scenarios

The BRI is typically used as a computer-to-computer programming interface. You can access the BRI using either a serial RS-232 connection or a TCP connection. In general, a reader that supports Ethernet or a Wi-Fi® radio makes BRI accessible via TCP; otherwise, the reader provides a serial interface to the BRI. For example:

- The IF4 fixed reader has only a serial interface.
- The IF61 and IF30 fixed reader have a TCP interface.
- The IP30 has a USB or Bluetooth connection, but the BRI is accessed through a TCP connection to a handheld computer.

You can also use the BRI interactively as a tool for trial, experimentation, and diagnostics. There are many available tools that let you access the BRI interactively:

- The IF61 web browser interface lets you enter BRI commands in the BRI Window. For help, see the ***IF61 Fixed Reader User's Manual***.
- HyperTerminal or another terminal emulation program for serial connections
- Telnet Client for TCP connections

BRI TCP Applications

If your BRI application uses a TCP connection to communicate with the reader, you need to understand TCP ports which are described in the following sections.

Also, be aware of any notes in this manual marked by these phrases:

- "For BRI Applications Using TCP"
- "If your BRI application communicates with the reader over a TCP connection"

Choosing a TCP Port for an Application Using TCP

Your reader listens for BRI connections on TCP port 2189, by default. You can choose another TCP port. For example, when configuring the IF61 fixed reader, you can enter another port in the **BRI TCP Port** field in the web browser interface. For detailed information about choosing a TCP port, see your reader user manual.

Resetting the Reader to the Factory Default Configuration

You can reset the reader to its factory default BRI configuration settings such as attributes and triggers with the FACDFLT command. This command has no parameters.

```
FACDFLT<CRLF>
```

If checksums are enabled, use this command instead:

```
FACDFLTF4<CRLF>
```

To learn more about checksums, see **"Using Message Checksums" on page 16**.

You can use this command to display the current values for the reader attributes on your reader:

```
ATTRIBUTE<CRLF>
```

The following table lists the default value for each reader attribute.

Default Factory Configuration

Reader Attribute	Default Value
ANTS	1
ANTTIMEOUT	50
ANTTRIES	3
BAUD	115200
BTPWROFF	300

Default Factory Configuration (continued)

Reader Attribute	Default Value
CHKSUM	OFF
DENSEREADERMODE	OFF
FIELDSEP	ASCII space character (0x20)
FIELDSTRENGTH	30dB, 30dB, 30dB, 30dB
IDREPORT	OFF
IDTIMEOUT	100
IDTRIES	1
INITIALQ	4
INITTRIES	1
NOTAGRPT	OFF
RDTRIES	3
RPTTIMEOUT	0
SELTRIES	1
SESSION	2
TAGTYPE	EPCC1G2
TIMEOUTMODE	OFF
TTY	OFF
UNSELTRIES	1
WRTRIES	3

Note: Not all BRI devices support all of these attributes (such as BTPWROFF and BAUD).

Identifying the Version of BRI on Your Reader

Identify the version of BRI software on your reader before you contact Intermec Technical Support. Use the BRIVER command, described in **"BRIVER" on page 40**, to display the BRI version.

This manual supports the Basic Reader Interface Version 3.17 and covers features that are not available in older versions of BRI software. If you have an older version of BRI software, see your Intermec representative for help upgrading BRI.

Conventions Used in This Manual

The example BRI commands and responses in this manual make the following assumptions:

- The attribute TTY is disabled, which is the default. The host interface terminates all commands with a <CRLF> (carriage-return, line-feed). For details, see **"TTY" on page 97**.
- The attribute CHKSUM is disabled, which is the default. The BRI does not return checksums for each line of output. For details, see **"Using Message Checksums" on page 16**.
- The attribute IDREPORT is disabled, which is the default. The tag identifiers are not displayed in the response to a READ or WRITE command. For details, see **"IDREPORT" on page 91**.
- The attribute NOTAGRPT is enabled, which is not the default. A message is sent to notify you when no tags were found to operate on. For details, see **"NOTAGRPT" on page 93**.
- The field separator is assumed to be set to the space character.

The command syntax descriptions in this manual follow these formatting conventions:

- Parameters in brackets [ ] are optional.
- Parameters in angle brackets < > are required.

2

Designing Robust BRI Applications

This chapter introduces guidelines for designing applications with a robust BRI interface that can successfully control any BRI-capable Intermec reader and handle a variety of conditions. This chapter contains these sections:

- **Before You Begin Programming**
- **Recommended Software Structure**
- **Programming the BRI Message Layer**
- **Programming the BRI Transport Layer**
- **Multi-Threaded Implementation**

Before You Begin Programming

A BRI host application is an application that uses the BRI protocol to control an RFID reader to conduct RFID operations. An application with a robust BRI interface should successfully control any BRI-capable Intermec reader and handle a variety of conditions. This chapter explains how to achieve a robust BRI application.

Before you begin programming, you need to make two decisions:

- Will the BRI host application manage all RFID configuration parameters, or will you use a configuration management application such as SmartSystems™ Foundation? For details, see the next section, "**Managing RFID Configuration**."
- Will you use the RFID Resource Kit, available as a download from the Intermec Developer Library (IDL)? For details, see **"Using the RFID Resource Kit" on page 9**.

Managing RFID Configuration

You need to decide if the BRI host application manages all RFID configuration parameters, or if a configuration management application such as SmartSystems Foundation manages all RFID configuration parameters.

Here are the advantages of using SmartSystems Foundation (or a similar application) for RFID configuration:

- **Extensibility**. The RFID data flow is separated from the RFID configuration management. The host application focuses on managing RFID data flows. You use the SmartSystems interfaces to manage the configuration changes which may be required for upgrades or system tuning.
- **Scalability**. The host application can be designed to operate independently of device-specific values, such as the device address and name. Adding additional RFID readers to the system will not impact the host application software.
- **Maintainability**. Separating the data flow from the configuration management makes it easier to isolate problems. It is also easier to replace existing hardware for repairs.

Here are the advantages of integrating RFID configuration into your BRI application:

- **Rapid development**.
- **Minimize third-party dependence**. You eliminate your dependency on Intermec product support.

If you decide to separate RFID configuration from the BRI host application, then the host application should only set BRI attributes that are essential to the application. For example, the BRI host application may set BRI attributes that manage dynamic control of the reader (such as the ANTENNA attribute, which manages switching between antennas). For details about attributes, see **"Understanding the <ATTRIBUTE NAME> Parameter" on page 86**.

Using the RFID Resource Kit

You can use the RFID APIs to provide an abstraction layer between Intermec RFID devices. This ensures compatibility with all Intermec BRI readers and future enhancements to BRI. The RFID APIs are included in the RFID Resource Kit, which you can download from the Intermec Developer Library (IDL).

To download the RFID Resource Kit

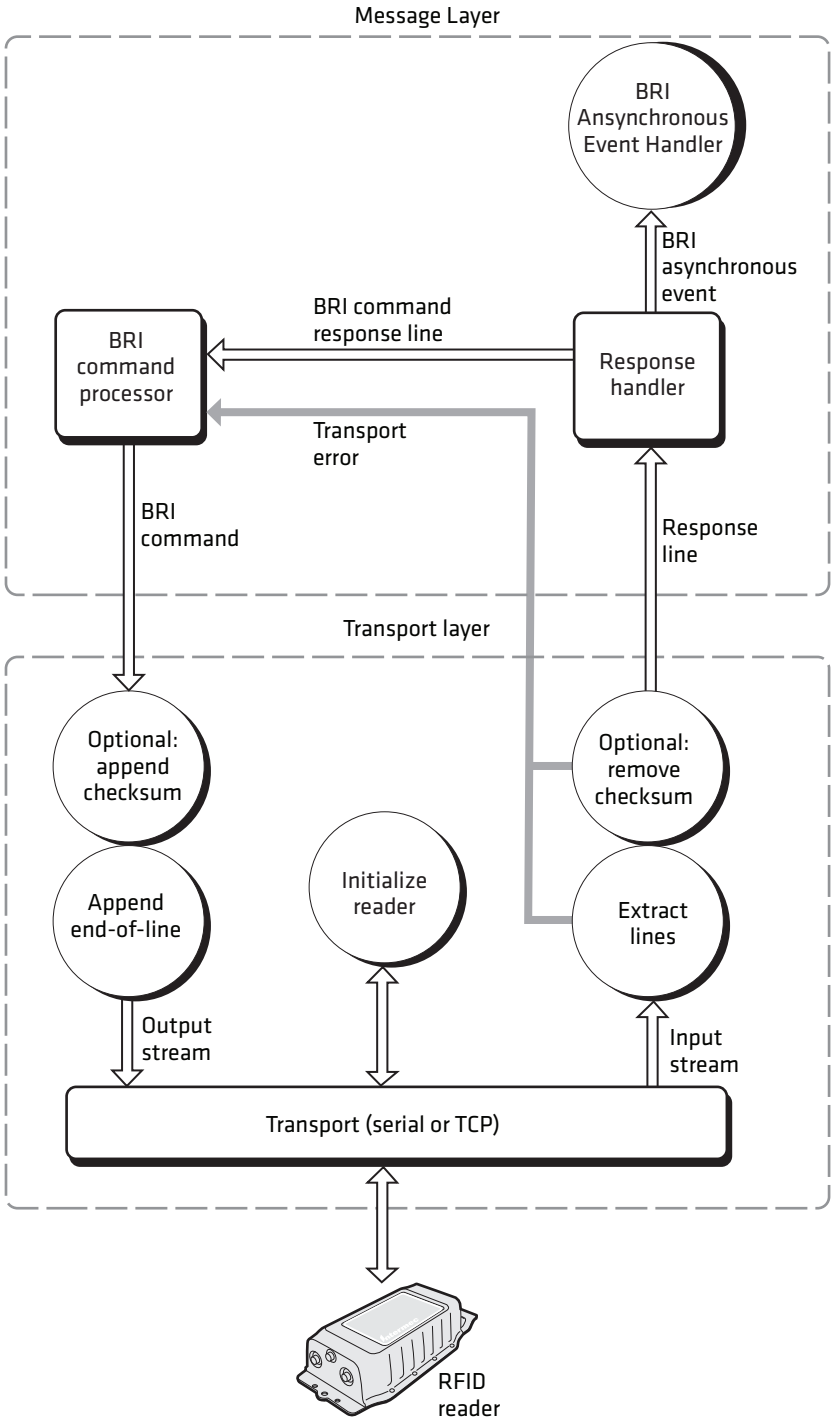
- From the Intermec website at **www.intermec.com**, select **Products** > **Software and Tools** > **Developer Library** > **Developer Resource Kits**.

If you cannot use the RFID Resource Kit, you should implement the software structure described in this chapter.

Recommended Software Structure

This section describes a design for software that communicates with a BRI reader. Follow this design if you need to develop your own implementation without the RFID Resource Kit.

This software structure has distinct Message and Transport layers, which are described in **"Programming the BRI Message Layer" on page 11** and **"Programming the BRI Transport Layer" on page 12**.



Recommended Software Structure: *If you are not using the RFID Resource Kit, your BRI application should follow this structure.*

Programming the BRI Message Layer

When programming the BRI Message Layer, you need to understand the message types and how they are handled.

BRI Message Types

The BRI protocol includes three message types:

- **BRI Command**. This message type is a single BRI command sent to a reader.
- **BRI Command Response**. This message type is the reader response to a BRI command. There are two types of BRI Command Response messages:
  - **Command Error Response**: An error response consists of two lines, one error line followed by one OK> response terminator. The error line may be formatted in three ways:

    ```
    ERR
    ERR <keyword>
    CKERR
    ```

    Note: The BRI specification also defines field-level errors, such as WRERR. These errors do not indicate that the entire command has failed. They indicate that only part of the command failed.

    For a list of all errors, see **"Understanding Error and Success Responses" on page 109**.
  - **Normal Command Response**: Any command response that is not an error response is a Normal Command Response. It consists of zero or more BRI response lines followed by one OK> response terminator line. In general, the normal command response syntax is as follows:

    ```
    ResponseLine<CRLF>
    ResponseLine<CRLF>
    ...
    OK><CRLF>
    ```

    As described in the previous note, the Normal Command Response also includes any field-level errors.
- **Asynchronous Event Messages.** This message type is a single line message that is not a response to a command. Each asynchronous event begins with the EVT: prefix. An asynchronous event may occur between lines of another message. In general the asynchronous event syntax is as follows:

  ```
  EVT:<Type> <EventDetails><CRLF>
  ```

  Asynchronous event messages are single-line messages. They are not followed by their own OK> line. For more details about EVENT messages, see **"Understanding Error and Success Responses" on page 109**.

BRI Command Processor

A BRI Command Processor composes BRI Command strings for the application and sends them to the transport layer. It receives and accumulates BRI Command Response lines until it receives the OK> line marking a complete BRI Command Response. Responses are received in the order in which their corresponding commands were sent.

You can parse a normal command response line into data fields by separating the line at each field separator character that is not inside a quoted string.

BRI Asynchronous Event Handler

A BRI Asynchronous Event Handler must handle all types of BRI Asynchronous Events used by the application. It should quietly discard event types it does not recognize.

Response Handler

A Response Handler handles all incoming lines by sending:

- all lines that start with EVT: to the BRI Asynchronous Event Handler.
- all other lines to the BRI Command Processor as command response lines.

Programming the BRI Transport Layer

When programming the BRI Transport Layer, you need to understand transport operation and initialization.

Transport Operation

The Transport Layer appends an end-of-line to each outgoing message. It continually monitors the input stream, accumulating complete input lines and sending them to the Message Layer.

For serial transport implementations that use message checksums the Transport Layer also adds checksums to outgoing commands and verifies and removes checksums from incoming message lines. The command processor should be notified of transport failure when an incoming checksum is wrong. For details about checksums, see **"Using Message Checksums" on page 16**.

The detailed processing steps of the Transport Layer are:

1. (Optional) Append checksum. This optional step may be used only over a serial connection to a reader. Sum the binary value of the characters in the message, encode the least significant 8 bits of the sum as a two-digit hexadecimal value, and append the two digits to the end of the message.
2. Append end-of-line character. Append a <CRLF> to mark the end of the message in the output stream.
3. Extract lines from input stream. Accumulate input characters until the end-of-line is received, and then send the complete response line to the Response

Handler. Lines must be separated in some way, such as preserving the <CRLF>s or converting each line into a separate string.

4. (Optional) Remove and verify the checksum. Remove the last 2 characters from the message line and interpret them as an 8-bit hexadecimal value. Compare the value to the least significant 8 bits of the sum of the rest of the message. If the comparison fails, there is no way to tell the reader to resend a line. Instead, you should notify the Message Layer of a communication failure.

Transport Initialization

A BRI application should assume as little as possible about the state of the RFID reader when it first connects to the reader. The application can overcome some unexpected prior states of the reader by sending a few BRI commands to initialize the reader before entering normal operation.

Immediately after establishing data transport with the reader, the BRI Transport Layer may send an initialization command sequence to the reader and then consume the input stream until it receives the expected number of OK><CRLF> response terminators.

The initialization command sequence depends on the type of transport you are using. To confirm connectivity and expected operation of the RFID reader, use the BRIVER command. For help, see **"BRIVER" on page 40**.

Multi-Threaded Implementation

You can implement the Message Layer and Transport Layer using multiple threads. In most cases, you can follow this design, which includes three threads:

- **BRI Command Processor Thread.** This thread calls the BRI Command Processor to execute a BRI command. Once a BRI command is sent to the reader via the transport layer, the Command Processor Thread usually blocks waiting for the OK> BRI response terminator to be returned by the Input Message Thread, described next.
- **Input Message Thread.** It is best to maintain a separate thread that always monitors responses and events from the reader. This input thread routes messages based on whether they are command responses or asynchronous event messages. Typical processing logic for the Input Message Thread is defined by the following pseudo-code:

```
While connected to reader {
    Read line from input stream
    If input line begins with "EVT:" then
        Send line to BRI Event Handler
    Else
        Send line to BRI Command Processor
}
```

You should design the Input Message Thread such that a single pass through the read loop executes quickly. An input line is read and then passed on to another thread for further processing.

- **BRI Event Handler Thread.** Since the Input Message Thread must execute quickly. it may be useful to use a separate thread to handle BRI events. The Input Message Thread quickly stores a new message in a queue and the Event Handler Thread removes messages from the queue as soon as it can get to them.

3

Understanding BRI Programming Elements

This chapter introduces various programming elements of the BRI. This chapter contains these sections:

- **BRI Logical Interface**
- **Using Message Checksums**
- **Reserved Keywords**
- **Constants**
- **Data Field Definitions**
- **Data Conditions**

BRI Logical Interface

The BRI is an RFID reader interface that uses this command/response structure:

1. An external host sends the reader a command.
2. The reader executes the command.
3. The reader responds to the host.

Both the commands and responses use ASCII characters followed by a terminator. The default terminator is a carriage return/line feed (<CRLF>) character sequence and can be changed if required. For help, see **"TTY" on page 97**.

There is no timing associated with sending a command to the BRI. The BRI waits indefinitely for the command terminator before executing the command.

Using Message Checksums

Beyond the command/response terminators, you can also use message checksums to enhance the communication resilience between the external host and reader.

All data sent to and returned from the BRI is returned as printable ASCII characters. Except for the command and response delimiters, these are <CRLF> by default, the XON and XOFF start and stop characters, or other non-printable ASCII characters as defined by the user.

Certain applications may require an additional level of data verification. In order to provide an increased level of data integrity, you can enable a BRI attribute that allows message checksums to be added to data sent to and from the BRI. The checksum value is sent or returned as two ASCII-readable characters and represents a modulo 256 summing of the command or response, up to but not including the response delimiter characters. The checksum shall be calculated and then returned as two ASCII characters representing the hex value of the modulo 256 sum. The field separator character is inserted just before the checksum and is included in the checksum calculation.

```
READ TAGIDA5<CRLF>
0123456789ABCDEFh 0A<CRLF>
```

In this example, the checksum value for the response is 0x40A. The response checksum is returned as the two least significant characters of the modulo 256 calculated checksum. In this case, the characters 0 and A are returned.

You enable both command and response message checksums when you enable the attribute CHKSUM. If a checksum error is detected on an incoming message, the BRI returns an error message CKERR to the host. For commands, the checksum is done on all characters up to the actual checksum itself. If you decide to put an ASCII space character before the checksum value, you must include that space character in the checksum calculation.

You can use this case-sensitive command to turn off checksums:

```
ATTRIBUTE CHKSUM=OFFB7<CRLF>
```

If TTY is disabled, you should terminate this ATTRIBUTE command by pressing **Ctrl-J** (<LF>). For details, see **"TTY" on page 97**.

Reserved Keywords

This section briefly describes the keywords reserved for use by the BRI. Other sections in this manual contain more details about keyword usage.

Keywords Reserved for BRI Commands

The following table lists the keywords reserved for BRI commands.

Keywords Reserved for BRI Commands

Keyword	Description
ATTRIBUTE or ATTRIB	Reads or sets RFID reader attributes.
BRIVER	Returns the BRI specification version or feature level supported by reader or module.
CAPABILITIES or CAP	Returns the capabilities of the RFID reader.
DIAGNOSTICS or DIAG	Determines certain runtime characteristics of an RFID reader.
ERASE	Allows you to perform a block erase of an EPC Gen 2 tag.
FACDFLT	Sets factory default values for attributes.
FLEXQUERY	Allows the command to the use FLEXQUERY command instead of the standard EPCC1G2 QUERY command.
HELP	Lists all BRI commands supported by the reader.
HWCC	Returns the country code of the reader.
HWID	Returns a unique identifier that represents the reader hardware instance.
HWPROD	Returns the product name of the reader device.
HWREGION	Returns the region data for the reader device.
HWVER	Returns the board version level of the reader device.
KILLTAG	Supports the EPC Class 1 Gen 2 KILL operation.
LOCK	Maintains backward compatibility with previous BRI versions.
PING	Checks if a reader is available.
PLATDFLT	Returns your reader to the platform defaults.
PRESET	Resets the reader so that it behaves as if external power was disconnected and then reconnected.
PRINT	Displays the contents of a macro.
PROTECT	Allows you to turn memory access locks on or off.
R, RD, or READ	Reads information from one or more tags.
READGPI, RDGPI, RDGPIO, or READGPIO	Reads the current general purpose input port values.
REPEAT or RPT	Re-issues the last READ or WRITE command.
RESET	Causes the reader to execute a warm boot.

Keywords Reserved for BRI Commands (continued)

Keyword	Description
SET	Creates, lists, and deletes macros.
SWVER	Returns the current firmware version.
TRIGGER or TRIG	Defines a trigger based on time or a general purpose I/O signal.
TRIGGERQ, TRIGQ, TRIGQUEUE, or TRIGGERQUEUE	Checks the trigger event queue for events that are currently stored.
TRIGGERREADY, TRGRDY, TRIGREADY, or TRIGGERRDY	Transitions the reader event queue to the READY state; enables the asynchronous reporting of the oldest trigger event in the reader event queue.
TRIGGERWAIT, TRIGW, TRIGWAIT, or TRIGGERW	This command is no longer available. Use TRIGGERREADY instead.
VERSION or VER	Displays the reader firmware version.
W, WR, or WRITE	Writes information to one or more tags.
WRITEGPO, WRGPO, WRGPIO, or WRITEGPIO	Writes the general purpose output values.

Note: The TRIGGERWAIT command is no longer available. Because the term "wait" is confusing, the TRIGGERWAIT command has been replaced by the TRIGGERREADY command. The TRIGGERWAIT command remains in the BRI for backward compatibility. Intermec recommends that you use TRIGGERREADY.

Keywords Reserved for Command Parameters

The BRI command syntax defines several reserved keywords to express parameters for each of the BRI commands. The following table lists the reserved keywords, special characters, and their meaning.

Keywords Reserved for Command Parameters

Keyword or Special Character	Description
!=	Not equal to.
\n	Defines a line feed value.
\r	Defines a carriage return value.
\xxx	Defines an octal character.
" "	Double quotes are the default characters to define simple text string values.
<	Less than.
=	Equal to.
>	Greater than.
ACTION	When used in defining a TRIGGER, this keyword can be used to have the trigger execute a command based on a defined macro.
AND	Performs a logical AND operation of statements in a WHERE clause.
ANTENNA or ANT	Specifies that the command should return the antenna used during the READ or WRITE command.

Keywords Reserved for Command Parameters (continued)

Keyword or Special Character	Description
BLOCK	Specifics that the reader should use EPCC1G2 block-write commands to write the data to the tag.
CHANNEL	Shortcut name to this keyword is CHAN. This keyword is an integer data field that specifies the channel frequency on which the tag was identified.
COUNT	Indicates the number of times a tag was seen in the current inventory round.
EPCID	Shortcut for data type HEX(1:4,L), the EPC field on an EPCglobal Class 1 Gen 2 /ISO 18000-6C tag.
FILTER or FILT	Specifies the time required to delay after a defined I/O event occurs prior to resuming processing of the event. Used only by the TRIGGER command.
HEX	Defines a hexadecimal field.
INT	Defines a 1, 2, 3, or 4 byte unsigned numeric field.
ISOUII or UII	Keyword that is returned when the tags in the field of view are inventoried.
NOT	Logical inverse operation.
OR	Performs a logical OR operation of statements in a WHERE clause.
PC	Displays the protocol control bits of an EPCglobal Class 1 Gen 2 tag.
PERMANENT	Permanently locks or unlocks a specified memory bank.
RNSI	Negative numeric data field associated with the received noise strength of a tag being returned for the current tag operation. Values are measured in dBm. Larger values indicate louder noise, while smaller values indicate quieter noise.
RSSI	Determines the received signal strength of a tag being returned for the current tag operation.
SNR	Numeric data field that specifies the signal-to-noise ratio measured as a tag is responding. SNR is equivalent to RSSI minus RNSI. Values are measured in dBm.
STANDARD	Specifics that the reader should use EPCC1G2 standard-write commands to write the data to the ta
STRING or STR	Defines an ASCII field.
TAGID	Defines a hexadecimal field that represents the tag identification information contained on the tag.
TAGTYPE	String representation of the type of tag being returned for the current tag operation.
TIME	Specifies that the command should return the timestamp for the READ or WRITE command.
WHERE	Defines the conditions applied to reading and writing specific tags.

Keywords Reserved for Reader Attributes

The following table lists the reserved keywords for reader attributes.

Keywords Reserved for Reader Attributes

Keyword	Description
ANTENNAS or ANTS	Sets the sequence of antennas to be used during READ and WRITE commands.
ANTTIMEOUT or ANTTO	Sets the timeout of a READ or WRITE on each antenna.
ANTTRIES	Sets the number of times a READ or WRITE is executed on each antenna.
ATA or ATAHALF	Sets the tagtype to America Trucking Association in either full or half frame.
BAUD	Sets the reader baud rate.
BTPWROFF	Sets the time period (in seconds) the Bluetooth radio (if available) will search for a Bluetooth connection before turning off to save power.
CHKSUM	Sets the BRI to return checksums for each line of output.
DENSEREADERMODE or DRM	Enables or disables the dense reader mode operation.
ECHO	Sets the BRI to echo input commands back to the host.
EPCC1G2PARAMETERS or EPCC1G2PARMS	Selects the EPCC1G2 configuration parameters. Tags from some vendors will not work reliably until you select the proper parameter set.
FIELDSEP	Sets the BRI to format the output data by placing this value between fields.
FIELDSTRENGTH or FS	Sets the RF power level of each antenna.
IDREPORT	Specifies if the BRI automatically reports tag IDs.
IDTIMEOUT or IDTO	Sets the ID timeout of the identify algorithm execution.
IDTRIES	Sets the number of times the identify algorithm is to be executed.
INITIALQ	Sets the initial Q value for the EPCglobal Class 1 Gen 2 query command.
INITTRIES	Sets the number of initialization tries.
LBTCHANNEL	Determines the default transmit channel for ETSI 302-208 when executing the Listen Before Talk algorithm.
LBTSCANENABLE	Enables ETSI 302-208 channel frequency scanning when running the Listen Before Talk algorithm.
LOCKTRIES	Sets the number of times that an attempt is made to find tags before a response is returned to a LOCK command.
NOTAGRPT	Specifies if the BRI automatically sends a message if no tags are detected during a READ or WRITE command.
RDTRIES	Sets the number of times the read algorithm is to be executed.
RPTTIMEOUT or RPTTO	Sets the amount of time to delay event reports for READ commands in Continuous mode.

Keywords Reserved for Reader Attributes (continued)

Keyword	Description
SCHEDULEOPT or SCHEDOPT	Determines how antennas are switched during the inventory process. For details, see **"Understanding the Timeouts and Tries" on page 99**.
SELTRIES	Sets the number of times GROUP SELECT commands are issued to the tag field during a read or write.
SESSION	Sets the EPCglobal Gen 2 session mode information.
TAGTYPE	Sets the type of Intellitag tags or EPCglobal Gen 2 tags that will be read and written.
TIMEOUTMODE	Establishes the use of IDTIMEOUT and ANTTIMEOUT. For details, see **"Understanding the Timeouts and Tries" on page 99**.
TTY	Sets the BRI to respond to <CR> or <LF> only, or to the <CRLF> sequence.
UNSELTRIES	Sets the number of times unselect commands are issued to the tag field during a read or write.
XONXOFF	Enables or disables the flow control for a serial connection.
WRTRIES	Sets the number of times the write algorithm is to be executed.

Keywords Reserved for Reader Error and Success Responses

The following table lists the reserved keywords for reader error and success responses. For more details, see **"Understanding Error and Success Responses" on page 109**.

Keywords Reserved for Reader Error and Success Responses

Keyword	Description
ADERR	Response to a WRITE command for an EPCglobal Class 1 Gen 2 tag when you did not write an even-length value to an even-byte addresses.
CKERR	Response to a command with an invalid checksum.
ERASEERR	Response to an erase tag field command that was not successful.
ERASEOK	Response to an erase tag field command that was successful.
ERR	Response to a command that was not successful. Response generally indicates a command syntax error.
MERR	Response to any command that causes an "out of memory" error.
NOTAG	Response to a READ or WRITE command when no tags are found and when the NOTAGRPT attribute is enabled.
OK>	Response terminator that is included at the end of every command response.
PVERR	Response when the user has not established the proper privilege to access a tag. Possibly because the memory is locked or an incorrect password was used.
PWERR	Response when the WRITE command failed because the tag had low power.
RDERR	Response to a read tag field command that was not successful.

Keywords Reserved for Reader Error and Success Responses (continued)

Keyword	Description
WRERR	Response to a write tag field command that was not successful.
WROK	Response to a write tag field command that was successful.

Constants

This section describes the string, hex, binary, integer, and octal constants:

- String constants are specified by surrounding the string text with double quotes (“ ”). String constants can include non-printable characters by using the \ notation; for example, `\007`.
- Hex constants are specified by placing an uppercase `H` or lowercase `h` in front of the hexadecimal characters (0 to 9, A to F). Hex constants must be specified in pairs. For example, `H0F`.
- Binary constants are specified by placing an upper case `B` or lowercase `b` in front of the binary characters (0 to 1).
- Integer constants are specified using the characters 0-9. Integer values can range from 0 to 4,294,967,295.
- Octal constants are specified using the characters 0 to 7. Octal values can be specified by using the \0xxx notation or the 0xxx notation.

Note: Any numeric constant that contains a leading 0 (zero) will be interpreted as an octal constant.

Data Field Definitions

BRI commands use parameters to define the location and type of data memory stored or retrieved from the tag. These parameters are referred to as data type definitions. The BRI has reserved, predefined data type definitions associated with the tag.

ANTENNA

You can also use the shortcut name ANT.

ANTENNA is positive integer data type associated with the tag and indicates which antenna primarily located the tag. ANTENNA is a read-only value reported during the execution of READ and WRITE commands. Additionally, this data field cannot be used in a WHERE clause.

ANTENNA is a reserved keyword which, when specified in a READ or WRITE command, returns the antenna number used during the read or write. This value is returned as a decimal integer value ranging from 1 to 4. The antenna number can range from 1 to 4 depending upon the reader and the number of connected antennas.

AFI

AFI is a keyword used to point at the bits within the Gen 2 PC bits that are used to carry the Application Family Identifier. When writing the AFI the tag EPC/ISO bit is set to one. For example:

```
WRITE AFI=Hxx
```

Once the AFI has written to a tag, the tag takes on the ISO-6C definitions. The EPCID keyword will return the 96 bit value, however, once the AFI is written, the EPCID keyword can no longer be used in the WHERE clause for EPCID matching.

BIT(memory_bank:startbit, bits)

Data types declared as BIT can range in length from zero to the number of bits given by the *bit* parameter. All BIT data types are represented as the binary characters [0-1].

The BIT data type is currently restricted for use in WHERE clauses only. This means that the BIT data type can not be used to refer to data that are either read or written from /to tags.

The *memory_bank* parameter is optional and only applies to EPCglobal Class 1 Gen 2 tags. For a list of valid values for *memory_bank*, see the Valid Memory Bank Values table in the data field definition: **"HEX(memory_bank:address, length)" on page 24**.

The *startbit* parameter can range from zero to the maximum address for the defined field. For EPCglobal Class 1 Gen 2 tags and other tag types, if an address is larger than the space available on the tag, an error is returned for out of range addresses.

The *bits* parameter can range from zero to the number of bits in memory minus the startbit value. The *length* parameter is required.

Bit addressing is defined as the least significant bit if the byte with the lowest address in a bank has a bit address of zero. The most significant bit of the same byte has a bit address of seven. The least significant bit of the next byte has the bit address of eight, and so forth.

COUNT

The COUNT keyword is a positive integer data field associated with a tag and indicates the number of times a tag was seen in the current inventory round. Additionally, this data field cannot be used in a WHERE clause.

EPCID

For both EPCglobal Class 1 Gen 2 and ISO 18000-6C tags, EPCID is a special keyword that equates to `HEX(1:4,L)` which is the EPC memory bank. Here `L` is variable length that depends on the data read or written to the tag.

EPCID corresponds to the electronic product code (EPC), the unique identifier for the tag, that is automatically returned when the tags in the field of view or the reader are inventoried. The EPCID refers to the 12 byte or greater data starting in location 4 of memory bank 1 of a tag. The first 4 bytes should not be written to by a user.

EPCID can also apply to UCODE119 tags from Phillips. The 12-byte EPCID is coded as described in the Phillips document, *Implementation of EPC Tag Data on U-Code EPC 1.19 Version 1.0 June 2004*.

The following rules apply to EPCID:

- `WRITE EPCID=H`*`XXXXX`* is a shortcut for writing an EPC to a tag.
  - EPCglobal Class 1 Gen 2 defines memory in 16-bit words on even byte addresses.
  - The hex value must be one or more bytes. A typical EPCID is 12 bytes or 96 bits, but can be more or less than 12 bytes.
  - The header field is automatically calculated when using the EPCID keyword. The AFI flag and the 8 AFI bits are set to zero indicating that the tag format is now EPC.
- For READ EPCID, the value returned is at least one byte.

You can write data to the EPCglobal Class 1 Gen 2 tags EPC memory bank using:

`HEX(1:`*`B`*`, `*`L`*`)`

where:

B is the byte offset into the memory bank. *B* must be even.

L is the length. *L* must be greater than 1 and less than 67.

The EPCglobal Class 1 Gen 2 standard permits up to 66 bytes (CRC-16, PC-16, 31-EPC data words) in this bank but the tag manufacturer may supply less than 31 EPC data words. An attempt to write data beyond what exists in the tag results in a write error.

You should be careful writing bytes 0-3 on a tag. Bytes 0-1 correspond to the CRC-16. The tag recalculates the CRC-16 value each time the tag is powered on. Bytes 2-3 correspond to the protocol control (PC) word which includes the data length field, the EPC/ISO bit, and the header bits. You must be careful to encode the PC correctly. The data length field is the word count of the PC plus the EPC data words. An incorrect value written to locations 2 and 3 could render the tag unreadable until proper values are written to location 2 and 3.

HEX(memory_bank:address, length)

Data types declared as HEX can range in length from zero to the maximum tag address in length as specified by the *length* parameter. All HEX data types are represented as the hexadecimal characters (0 to 9, A to F).

A unique feature of the HEX data type when used in a WHERE clause is the ability to use wildcard character pairs of `??` to represent a "don't care" condition. For details, see **"READ" on page 54** and **"WRITE" on page 69**.

Note: The wildcard notation cannot be used when the tagtype is set to EPCglobal Class1 Gen 2 tags.

The *memory_bank* parameter is optional and only applies to EPCglobal Class 1 Gen 2 tags. Tag memory is divided into four sections (0 to 3) and the *memory_bank* parameter indicates the section. The following table lists the valid values for *memory_bank*. If you omit the *memory_bank* parameter, the EPC memory bank is assumed for EPCglobal Class 1 Gen 2 tags.

Valid Memory Bank Values

Value	Memory Bank Name for the Section of Tag Memory	Default
0	Reserved (passwords)	
1	EPC (electronic product code)	X
2	TID	
3	User Memory	

The *address* parameter can range from zero to the maximum tag address. If an address is larger than the space available on the tag, the response depends on the tag type:

- For ISO 18000-6B tags, the address wraps to the beginning of the tag memory.
- For EPCglobal Class 1 Gen 2 tags and other tag types, an error is returned for out-of-range addresses.

The *length* parameter can range from zero to the length of the data space on the tag minus the address. The *length* parameter is required.

A HEX field with *length* zero in a read command implies "read until end of memory bank." This only works for EPCC1G2 tag types. Other tag types will respond with "RDERR" when zero lengths are specified.

INT(memory_bank:address, length)

Data types declared as INT can range from one to four bytes in length as specified by the *length* parameter. All INT data types can be represented as decimal values. The range of each INT data type is shown below:

- 1 byte: 0 to 255
- 2 byte: 0 to 65,535
- 3 byte: 0 to 16,777,215
- 4 byte: 0 to 4,294,967,295

Data written to the tag is stored in big endian format. The most significant byte of the data is stored at the first address location specified in the data type.

The *memory_bank* parameter is optional and only applies to EPCglobal Class 1 Gen 2 tags. For a list of valid values for *memory_bank*, see the Valid Memory Bank Values table in the data field definition: **"HEX(memory_bank:address, length)" on page 24**. If you do not specify a memory bank, the EPC memory bank is assumed for EPCglobal Class 1 Gen 2 tags.

The *address* parameter can range from zero to the maximum address for the defined field. If an address is larger than the space available on the tag, the response depends on the tag type:

- For ISO 18000-6B tags, the address wraps to the beginning of the tag memory.
- For EPCglobal Class 1 Gen 2 tags and other tag types, an error is returned for out-of-range addresses.

The *length* parameter is required and must be 1,2, 3, or 4.

ISOUII

Note: Currently, this feature is not fully functional in this version on the BRI and this feature is only capable of returning the EPCglobal defined EPCID information. The length of the EPCID data can vary from 1 to 32 bytes depending on the value programmed in the PC of memory bank 1 on the tag.

You can also use the shortcut name UII.

For ISO 18000-6C tags, ISOUII is a special keyword that equates to `HEX(1:4,L)` which is the UII memory bank. Here `L` is variable length that dependent on the data that is read or written to the tag.

ISOUII corresponds to the ISO Unique Item Identifier that is automatically returned when the tags in the field of view are inventoried. The data bytes correspond to the UII portion of the UII memory bank.

The following rules apply to ISOUII:

- `WRITE ISOUII=HXXXXX` is a shortcut for writing a UII to a tag.
  - There must be an even number of bytes in the hex value. ISO 18000-6C defines memory in 16-bit words on even byte addresses.
  - The hex value must be at least 2 bytes long (the header field plus 1 UII word).
  - The hex value must be less than 62 bytes long (the header field plus 31 UII words).
  - The header field implies an UII length. The data length written to the tag is implied by the length of the hex value supplied.
  - The EPC/ISO bit in the tag protocol control word is set to one. Once the ISOUII is written then the ISOUII can be used in a WHERE clause to match the ISOUII information and the EPCID can no longer be used to match.
- For READ ISOUII, the length UII data is determined by the tag length field found in the PC word. UII is this length minus one times two ((L-1) * 2) bytes.

You can write data to the UII memory bank using:

`HEX(1:`*`B`*`, `*`L`*`)`

where:

B is the byte offset into the memory bank. *B* must be even.

L is the length. *L* must be greater than 1 and less than 67.

The ISO 18000-6C standard permits up to 66 bytes (CRC-16, PC-16, 31-UII data words) in this bank but the tag manufacturer may supply less than 31 UII data words. An attempt to write data beyond what exists in the tag will result in a write error.

You should be careful writing bytes 0 to 3 on a tag. Bytes 0 and 1 correspond to the CRC-16. The tag recalculates the CRC-16 value each time the tag is powered on. Bytes 2 and 3 correspond to the protocol (PC) control word which includes the data length field, the EPC/ISO bit, and the header bits. You must be careful to encode the PC correctly. The data length field is the word count of the PC plus the UII data words.

Note: Locations 0 through 3 should not be written and the starting address for the command should be 4 or greater.

PC

The PC keyword is used to display the protocol control bits of an EPCglobal Class 1 Gen 2 tag. This is a read-only field. The displayed value is a hexadecimal value of 4, 8, or 12 digits. The first set of four digits represents the tag protocol control (PC) word. The second set of four digits represents the first extended protocol control word (XPC_1) for the tag. The third set of four digits represents the second extended protocol control word (XPC_2) for the tag.

RSSI

The RSSI keyword is a negative integer data field associated with the received signal strength of the tag being returned for the current tag operation. The minimum value for RSSI is -128 dBm, and the maximum value is 0 dBm.

STRING(memory_bank:address, length)

You can also use the shortcut name STR.

Data types declared as STRING can range in length from zero to the length of the data space as specified by the *length* parameter.

All STRING data types are represented as the printable ASCII character set. If a value is not in the ASCII printable range (32 to 127), the data is displayed in a format that is dependent on the host device. If a string data type is defined, it should contain only printable ASCII characters.

The *memory_bank* parameter is optional and only applies to EPCglobal Class 1 Gen 2 tags. For a list of valid values for *memory_bank*, see the Valid Memory Bank Values table in the data field definition: **"HEX(memory_bank:address, length)" on page 24**.

The *address* parameter can range from 0 to the length of the data space. If an address is larger than the space available on the tag, the response depends on the tag type:

- For ISO 18000-6B tags, the address wraps to the beginning of the tag memory.
- For EPCglobal Class 1 Gen 2 tags and other tag types, an error is returned for out-of-range addresses.

The *length* parameter can range from zero to the length of the data space minus the address. The *length* parameter is required.

When a STRING data type is written to a tag, only the requested characters are written. The NULL character is not stored in tag memory at the end of a STRING data type. For example:

```
STRING(10,5)="HELLO"
```

The data stored in tag memory starting at address 10 is H, E, L, L, O for a total of five characters.

When STRING data types are specified in READ commands, the data returned is equal to length specified in the READ command. Unprintable characters are returned in a HEX format \xYY where YY is the data representing the hex value.

When writing string fields, you can enter binary values using the *xxx* notation. However, do not try to use this method to enter a NULL character into a string field. BRI returns ERR if you enter \000 into a string field.

TAGTYPE

TAGTYPE is a string representation of the type of tag being returned for the current tag operation. Possible values for TAGTYPE consist of all of the supported TAGTYPE attribute values. This data field cannot be used in a WHERE clause. For more information, see **"TAGTYPE" on page 96**.

TIME

TIME is a positive integer data type associated with the tag and indicates the time when the tag was primarily located. Time is a read-only data field, and cannot be used in a WHERE clause.

TIME is a reserved keyword which, when specified in a READ or WRITE command, returns a time value indicating when the READ command was completed. The time value is returned as an integer value with 1 millisecond (ms) resolution.

The time base value returned is reset to zero when a reader is first powered up. Readers do not provide real time clock capabilities, so time is relative to the reader powering up or being reset. The range of the TIME parameter is returned as the number of milliseconds since power-up of the reader and ranges from 0 to 4294967295 ms. This represents approximately 49.7 days before the time rolls over if the reader is on continuously and the BRI session is maintained.

The TIME information is returned as an integer in the following format:

```
1234
```

The value 1234 represents a time of 1.234 seconds.

Data Conditions

Data conditions are used to select tags that have data matching a defined constant value. The data conditions can be as simple or complex as necessary to uniquely identify a specific group of tags to operate on.

There are two limitations:

- The comparison must be made between data that is stored in a tag and a constant value. You cannot make a comparison between two memory locations contained on a tag. The BRI returns ERR<CRLF> if you compare two memory locations on a tag.
- The type of tags you read and write to controls the operators you can use in data conditions:
  - ISO 18000-6B tags support all operators (=, !=, >, <).
  - EPCglobal Class 1 Gen 2 tags support only the = and != operators.

You can use these formats for data conditions:

- Native tag selector logic or NOT logic lets you use all data condition operators (=, !=, >, <). For help, see the next section, "Using Native Tag Selector Logic in Data Conditions."
- AND/OR logic lets you use only the = and != data condition operators. For help, see **"Using AND/OR Logic in Data Conditions" on page 30**.
- EPCC1G2 select logic gives the application direct control of the EPCC1G2 select commands.

Using Native Tag Selector Logic in Data Conditions

```
WHERE [<Data Field> <Operator> <Constant>], [NOT <Data Dield>
<Operator> <Constant>]
```

where:

<*Data Field*> is one of the data types described in **"Data Field Definitions" on page 22**.

<*Operator*> is one of the operators described in the following table, "Operators for Native Tag Selector Logic Data Conditions."

<*Constant*> is one of the constants described in **"Constants" on page 22**.

Operators for Native Tag Selector Logic Data Conditions

Operator	Description
=	The value at the specified tag memory address is equal to the comparison value.
!=	The value at the specified tag memory address is not equal to the comparison value.
>	The value at the specified tag memory address is greater than the comparison value. This operator is not supported for EPCglobal Class 1 Gen 2 tags.
<	The value at the specified tag memory address is less than the comparison value. This operator is not supported for EPCglobal Class 1 Gen 2 tags.

Multiple data conditions must be separated by a comma or a space.

A simple example of a data condition is:

```
INT(0,1)=1
```

In this example, `INT(0,1)` specifies that an integer of one byte length starting at address 0 of the tag memory will be compared with integer value of one as specified by `=1`.

You can use the keyword NOT in the data condition when you use native tag selector logic. If a data condition contains NOT, the matching tags are not selected.

All data conditions are processed from left to right.

The following example applies only to ISO 18000-6B tags because it contains the > operator. Suppose you want to implement the following expression where *data* is the integer values at address 18:

```
100 < data < 200
```

You can specify this condition using this selector logic:

```
WHERE INT(18,1) > 100, NOT INT(18,1) > 199
```

The first condition selects all ISO 18000-6B tags whose data at address 18 is greater than 100. The second condition then unselects all tags whose data at address 18 is greater than 199. This data condition sequence implements the specified expression.

Using AND/OR Logic in Data Conditions

```
WHERE [<Data Field> <Operator> <Constant>] AND [<Data Field>
<Operator> <Constant>] OR...
```

where:

<*Data Field*> is one of the data types described in **"Data Field Definitions" on page 22**.

<*Operator*> is one of the operators described in the following table, "Operators for AND/OR Logic Data Conditions."

<*Constants*> is one of the constants described in **"Constants" on page 22**.

It should be noted that WHERE clauses are only allowed when the TAGTYPE attribute is set to a single tag type. WHERE clauses are not allowed in operations involving multiple air protocols. Wildcards may also not be used if TAGTYPE has more than a single value.

The data in the WHERE clause conditions depends on the TAGTYPE used.

Data Fields Supported by Different Tag Types

Tag Type	Data Field
G1/ISO6BG1	HEX, INT, STRING
G2/ISO6BG2	HEX, INT, STRING
ICODE119	HEX, INT, STRING
UCODE119 or V119	HEX, INT, STRING
EPCC1G2	BIT, HEX, INT, STRING

Operators for AND/OR Logic Data Conditions

Operator	Description
=	The value at the specified tag memory address is equal to the comparison value.
!=	The value at the specified tag memory address is not equal to the comparison value.
>	The value at the specified tag memory address is greater than the comparison value. This is not supported for EPCglobal Gen 2 tags.
<	The value at the specified tag memory address is less than the comparison value. This is not supported for EPCglobal Gen 2 tags.

A simple example of a data condition is:

```
INT(0,1)=1
```

In this example, `INT(0,1)` specifies that an integer of one byte length starting at address 0 of the tag memory will be compared with integer value of one as specified by `=1`.

Because the following expression attempts to compare two tag memory locations, the BRI returns ERR<CRLF> if you use this expression in a command:

```
INT(0,1)=INT(1,1)
```

Using the AND and OR Keywords

Two additional keywords are available for specifying data conditions:

- AND—If two or more data conditions are joined with AND, the tag data must match the conditions specified in both data conditions parameters.
- OR—If two or more data conditions are joined with OR, the tag data must match either of the conditions specified in the data conditions parameters.

A WHERE clause is processed from left to right. An OR condition selects a group of tags to be included in an expression. An AND keyword unselects or excludes tags from an expression.

The following examples demonstrate this concept.

```
READ INT(18) WHERE INT(19)=1 OR INT(20)=2 AND INT(21)=3
```

The WHERE expression is evaluated from left to right as follows:

- Include all tags if the data at address 19 equals 1.
- Include all tags if the data at address 20 equals 2.
- Exclude all tags if the data at address 21 does not equal 3.

Therefore, tags are selected only if there is a 3 at location 21 and either a 1 or a 2 at locations 19 and 20 respectively.

It is important to note that the AND/OR keywords apply to the condition that follows. The first term in a WHERE expression always has an implied OR and includes tags that contain the specified data.

You can write the previous example differently and still achieve the same result, as shown in the following example.

```
READ INT(18) WHERE INT(20)=2 AND INT(21)=3 OR INT(19)=1
```

The WHERE expression is evaluated from left to right as follows:

- Include all tags if the data at address 20 equals 2.
- Exclude all tags if the data at address 21 does not equal 3.
- Include all tags if the data at address 19 equals 1.

Therefore, tags are selected only if there is a 3 at location 21 and either a 1 or a 2 at locations 19 and 20 respectively.

Grouping Expressions Without Using Parentheses

You cannot use parentheses to group expressions using AND/OR logic. Therefore, it may be difficult to implement an expression using the AND/OR logic without a great deal of planning. To allow some primitive grouping in defining WHERE expressions, the AND/OR expression logic has been modified. Every time an AND expression is followed by an OR, the expression following the `OR` starts a new tag selection criteria. The following example illustrates this concept, but also adds a variation in expression evaluation. Using the previous example, the next two examples evaluate differently, strictly due to the order of the data in the expression.

In the following example, the WHERE expression evaluates exactly as described above.

```
READ INT(18) WHERE INT(19)=1 OR INT(20)=1 AND INT(21)=3
```

However, the following example changes the order of the expression, so the resulting tags selected are evaluated as if parentheses were being used:

```
READ INT(18) WHERE INT(20)=1 AND INT(21)=3 OR INT(19)=1
```

Because an OR follows an AND, the expression is evaluated as follows using parentheses:

```
READ INT(18) WHERE ( INT(20)=1 AND INT(21) ) OR INT(19)=1
```

Implied parentheses are applied any time an OR keyword follows an AND keyword.

Suppose you must find a tag that contains the following information: four-byte STRING data at location 18 with data `NAME`, two-byte INT data at location 22 with data `17`, and one-byte INT data at location 24 with data `E`. The expression that describes this tag is shown below:

```
STRING(18,4)="NAME" AND INT(22,1)=17 AND INT(24,1)='E'
```

Using EPCC1G2 Select Logic in Data Conditions

```
WHERE
SELECT(<target>,<action>,<truncate>,<membank>:<pointer>,<length>
)=B<mask>[,SELECT(...)=B<mask>]
```

where:

<*target*> is a numeric value from zero to four described in the following table, "Targets for EPCC1G2 Select Logic Data Conditions."

<*action*> is a numeric value from zero to seven that defines the action applied to the target. These values are described in the following table, "Actions for EPCC1G2 Select Logic Data Conditions."

<*truncate*> is a numeric value of 0 or 1 that specifies the value to use in the Truncate field of the Select command.

<*membank*> is a numeric value that specifies the memory bank on the tag to use with the data field.
<*pointer*> specifies the bit address.
<*length*> specifies the bit length.

<*mask*> is a bit string that a tag compares against the memory location that begins at <*pointer*> and ends <*length*> bits later. <*mask*> must have the same number of binary digits as <*length*>.

Targets for EPCC1G2 Select Logic Data Conditions

Value	Target
0	Inventoried (S0)
1	Inventoried (S1)
2	Inventoried (S2)
3	Inventoried (S3)
4	SL

Actions for EPCC1G2 Select Logic Data Conditions

Value	Matching Action	Non-Matching Action
0	Assert SL or Inventoried -> A	De-assert SL or Inventoried -> B
1	Assert SL or Inventoried -> A	Do nothing
2	Do nothing	De-assert SL or Inventoried -> B
3	Negate SL or (A -> B, B -> A)	Do nothing
4	De-assert SL or Inventoried -> B	Assert SL or Inventoried -> A
5	De-assert SL or Inventoried -> B	Do nothing
6	Do nothing	Assert SL or Inventoried -> A
7	Do nothing	Negate SL or (A -> B, B -> A)

You can use the QUERYSEL and QUERYTARGET attributes with the WHERE SELECT syntax to select a sub-population of tags. For more information, see **"QUERYSEL" on page 94** and **"QUERYTARGET" on page 94**.

This example selects all tags that match 0x1001E (the 19 MSB bits) at memory bank 1, bit location 0x28 (decimal 40) for 19 bits:

```
ATTRIB QUERYSEL=3
READ WHERE SELECT(4,0,0,1:0x28,19)=b0001000000000001111
```

This example selects tags with EPC IDs that are 12 bytes long and begin with 0x12 (8 bits):

```
ATTRIB QUERYSEL=3
READ WHERE SELECT(4,0,0,1:0x10,5)=b00110,
SELECT(4,1,0,1:0x20,16)=b00010010
```

Multi-Protocol Condition Usage

READ and WRITE commands can be applied to multiple tag types. The BRI attribute TAGTYPE specify global values that apply to READ and WRITE operation by default. These global values can be overridden locally by specifying TAGTYPE.

In the following example, the TAGTYPE stipulates that only ISO180006B tags will be read:

```
READ TAGID TAGTYPE=ISO6BG1
```

However, in the following example, the TAGTYPE stipulates that both ISO180006B and EPCC1G2 tags will be read:

```
READ TAGID TAGTYPE=ISO6BG1, EPCC1G2
```

Note: The order of the tag types specified also determine the order in which the tags are identified.

4

BRI Commands

This chapter describes the BRI commands and related topics. This chapter contains these sections:

- **BRI Commands**
- **BRI Extensions for NXP Tags**
- **BRI Extensions for Fujitsu Tags**
- **BRI Extensions for Impinj Monza 4 Tags**
- **BRI Extensions for EM Microelectronics Tags**
- **Generic Tag Access Command**
- **Understanding [READ FIELD] and [WRITE FIELD] Parameters**
- **Understanding the <ATTRIBUTE NAME> Parameter**
- **Understanding the Timeouts and Tries**
- **Understanding the [LITERAL] Parameter**
- **Reading and Writing STRING Fields**
- **Understanding ACCESS and KILL Passwords**
- **Understanding Error and Success Responses**
- **Understanding EVENT Messages**
- **Understanding the Format of BRI Command Responses**
- **Creating and Using BRI Macros**

BRI Commands

The BRI commands and responses are defined in this section. These examples follow these formatting conventions:

- Values in brackets [ ] are optional parameters.
- Values in angle brackets < > are required parameters.

Note: Not all BRI commands are supported by all Intermec RFID readers, and specific ranges for commands and attributes may differ from product to product depending on hardware options. For help, see **"Reader-Specific Platform Specifications" on page 118**," and the documentation for your Intermec RFID reader.

For BRI extensions and commands that are specific to:

- NXP tags, see **"BRI Extensions for NXP Tags" on page 74**.
- Fujitsu tags, see **"BRI Extensions for Fujitsu Tags" on page 76**.
- Impinj Monza 4 tags, see **"BRI Extensions for Impinj Monza 4 Tags" on page 80**.
- EM Microelectronics tags, see **"BRI Extensions for EM Microelectronics Tags" on page 81**.

ATTRIBUTE

Purpose: This command changes or reads the reader attributes.

Command Shortcut: `ATTRIB`

Changing the Reader Attributes

For BRI Applications Using TCP: If your BRI application communicates with the reader over a TCP connection, you cannot use the ATTRIBUTE command to modify the BAUD, CHKSUM, ECHO, or FIELDSEP attributes from their default settings. For details, see **"BRI TCP Applications" on page 3**.

Syntax: `ATTRIBUTE [<ATTRIBUTE NAME>=[<VALUE>]]*`

Parameters: <ATTRIBUTE NAME> = This parameter specifies the attribute to change in the reader. For a complete description of the reader attributes, see **"Understanding the <ATTRIBUTE NAME> Parameter" on page 86**.

<VALUE> = This parameter specifies the new value for the attribute:

- For "tries" attributes (such as RDTRIES), the value corresponds to a count.
- For "timeout" attributes (such as IDTIMEOUT), the value corresponds to a time period (units of milliseconds).

Examples: These examples demonstrate how to use the ATTRIBUTE command to change reader attributes.

Example 1:

This example returns the current settings for all attributes:

`ATTRIBUTE<CRLF>`

Example 2:

This example specifies that the reader should attempt a read operation on each [READ FIELD] up to three times:

```
ATTRIBUTE RDTRIES=3
```

For details about the [READ FIELD], see **"Understanding [READ FIELD] and [WRITE FIELD] Parameters" on page 85**.

Example 3:

This example specifies that timeouts will not be used when trying to read or write tags. Instead, IDTRIES and ANTTRIES will be used to specify a maximum number of attempts:

```
ATTRIBUTE TIMEOUTMODE=OFF
```

Example 4:

This example specifies that timeouts IDTIMEOUT and ANTTIMEOUT will be used when trying to read or write tags:

```
ATTRIBUTE TIMEOUTMODE=ON
```

Example 5:

This example specifies that the reader should use three identify cycles attempting to identify tags:

```
ATTRIBUTE IDTRIES=3
```

Example 6:

This example specifies that the reader should execute the identify operation for three seconds:

```
ATTRIBUTE IDTIMEOUT=3000
```

Example 7:

This example sets the reader attribute INITTRIES to a value of two:

```
ATTRIBUTE INITTRIES=2
```

Here is an example response when this ATTRIBUTE command is successful:

```
OK><CRLF>
```

Here is an example response when this ATTRIBUTE command fails:

```
ERR<CRLF>
```

Example 8:

This example sets the parameters WRTRIES to three, IDTRIES to two, and RDTRIES to three:

```
ATTRIBUTE WRTRIES=3, IDTRIES=2, RDTRIES=3
```

If there is an error on the command line or the parameter specified is not defined, the BRI returns ERR<CRLF>.

Reading the Reader Attributes

Syntax: `ATTRIBUTE <ATTRIBUTE NAME>`

Parameters: <ATTRIBUTE NAME> = This parameter specifies the attribute to read. For a complete description of the reader attributes, see **"Understanding the <ATTRIBUTE NAME> Parameter" on page 86**.

If you do not include an <ATTRIBUTE NAME> in the command, the BRI returns the current settings of all the reader attributes.

Examples: These examples demonstrate how to use the ATTRIBUTE command to read the reader attributes.

Example 1:

This example returns the current settings for all attributes:

```
ATTRIBUTE<CRLF>
```

Example 2:

This example requests the value for the RDTRIES attribute:

```
ATTRIBUTE RDTRIES
```

The BRI returns the value of the parameter followed by a <CRLF> sequence. The BRI returns OK><CRLF> for successfully reading the parameter. Here is an example response to this command:

```
RDTRIES=3<CRLF>
OK><CRLF>
```

Example 3:

This example requests the values for the IDTRIES and WRTRIES attributes:

```
ATTRIBUTE IDTRIES, WRTRIES
```

The BRI returns the value of the parameter followed by a <CRLF>. If an attribute name is specified that is not defined, the BRI returns ERR<CRLF>.

Here are example responses to this command:

- `IDTRIES=2<CRLF>`
  `WRTRIES=3<CRLF>`
  `OK><CRLF>`
- `ATTRIBUTE IDTRIES<CRLF>`
  `IDTRIES=4<CRLF>`
  `OK><CRLF>`
- `ATTRIBUTE IDTRY<CRLF>`
  `ERR<CRLF>`
  `OK><CRLF>`

BLOCKPERMALOCK

Purpose: This command allows the reader to permanently lock blocks of user memory. The size and number of memory blocks are dependent on the tag implementation. A variation of the command allows the reader to read the permalock status.

Syntax: `BLOCKPERMALOCK [flex_query_selector] LOCK [data_field]*[TAGTYPE=tagtype list] [WHERE <data condition>] [PASSWORD=<access_password>]`

Parameters: [flex_query_selector] = This parameter modifies the command for certain types of EPCC1G2 tags. For details, see **"FLEXQUERY" on page 47**.

<data_field> = Specifies a bank, offset, length, and mask in which the Permalock field is to operate. For example:

HEX(<Bank>:<Offset>,<Length>)=H<Data>

<Bank> = The bank should always be 3.

<Offset> = This parameter is the BlockPtr parameter of the EPCGlobal Gen2 specification. It specifies the starting address for the Mask field in units of 16 blocks. For example, <Offset> 0 indicates block 0, <Offset> 1 indicates block 16.

<Length> = This parameter is the BlockRange parameter of the specification. It specifies the range of the Mask field starting at <Offset> and ending (16*<Length>)-1 blocks later.

<Data> = Contains a bit mask indicating which blocks in the range should be locked. One bits indicate that a block should be locked. Zero bits indicate that a block is left alone. The most significant bit of <Data> corresponds to the lowest address block. <data> must be specified as a hexadecimal string and it must contain 16 bits for each unit of <length>. For example, when <length> is 1, <data> must consist of 2 hex pairs.

Examples: These examples demonstrate how to use the BLOCKPERMALOCK command.

Example 1:

```
BLOCKPERMALOCK LOCK HEX(3:0,1)=H0800
H0123456789ABCDEF01234567 LCKOK
```

Lock the 5th block of user memory.

Example 2:

```
BLOCKPERMALOCK LOCK HEX(3:2,1)=H0001
H0123456789ABCDEF01234567 LCKOK
```

Lock the 48th block of user memory.

Example 3:

```
BLOCKPERMALOCK LOCK HEX(3:0,1)=H0800 HEX(3:2,1)=H0001
H0123456789ABCDEF01234567 LCKOK
```

Lock the 5th and 48th blocks of user memory.

Example 4:

```
BLOCKPERMALOCK LOCK HEX(3:0,3)=H080000000001
H0123456789ABCDEF01234567 LCKOK
```

Lock the 5th and 48th blocks of user memory.

BLOCKPERMALOCK READ

Purpose: This command allows you to read the lock status.

Syntax: `BLOCKPERMALOCK [flex_query_selector] READ [data_field]*[TAGTYPE=<tagtype list>] [WHERE <data condition>] [PASSWORD=<"access_password">]`

data_field = Specifies a bank, offset, and length in this form: HEX(<bank>:<offset>,<length>).

<bank> = The bank should always be three.

<offset> = The offset is the "BlockRange" parameter of the specification. It specifies the range of blocks for the which the status should be returned. The range starts at <offset> and ends (16*<length>)-1 blocks later.

The result of a "BLOCKPERMALOCK READ" command is a hexadecimal value that represents a bit map of locked and unlocked blocks. A one bit indicates that the corresponding block is locked. A zero bit indicates that the block is unlocked. The most significant bit of the resulting hexadecimal value corresponds to the lowest address block.

Examples:

Example 1:

```
BLOCKPERMALOCK READ HEX(3:0,1)
HH0123456789ABCDEF01234567 H0800
```

Read the permalock status of the first 16 blocks of user memory.

Example 2:

```
BLOCKPERMALOCK READ HEX(3:3,1)
HH0123456789ABCDEF01234567 H0001
```

Read the permalock status of the third 16 blocks of user memory.

Example 3:

```
BLOCKPERMALOCK READ HEX(3:2,2)
HH0123456789ABCDEF01234567 H00000001
```

Read the permalock status of the second and third 16 blocks of user memory.

Example 4:

```
BLOCKPERMALOCK READ HEX(3:2,1) HEX(3:3,1)
HH0123456789ABCDEF01234567 H0000 H0001
```

Read the permalock status of the second and third 16 blocks of user memory.

BRIVER

Purpose: This command returns the BRI specification version or feature level supported by the reader or module.

Syntax: `BRIVER`

Examples:

```
BRIVER<CRLF>
3.16<CRLF>
OK>
```

CAPABILITIES

Purpose: This command allows an application to programmatically determine the capabilities of an RFID reader. The following syntax defines the CAPABILITIES command.

Command Shortcut: `CAP`

Syntax:
```
CAPABILITIES [GPICOUNT] [GPOCOUNT] [GPITYPE<index>]
[GPOTYPE<index>] [ANTENNAS] [TAGTYPE] [TRIGGERCOUNT]
[FIELDSTRENGTH<MIN | MAX>[index]] [DENSEREADERMODE]
[LISTENBEFORETALK [CHANNEL <MIN | MAX>][SESSIONS]
[EPCC1G2PARAMETERS] [SCAN] [NXP]
```

Parameters: Unlike other commands, the CAPABILITIES command has sub-commands which allow you to determine the specifications of your reader.

[GPICOUNT] = This sub-command returns the number of general purpose inputs available on your device. For example:

```
CAP GPICOUNT<CRLF>
4<CRLF>
OK>
```

[GPOCOUNT] = This sub-command will return the number of general purpose outputs available on your device. For example:

```
CAP GPOCOUNT<CRLF>
4<CRLF>
OK>
```

[GPITYPE] = This sub-command will return the type of input (digital or analog) for the input specified. If no specific input is requested, the type for each input will be returned. This command will return either DIGITAL or ANALOG. For example (assuming that there are 4 inputs):

```
CAP GPITYPE<CRLF>
DIGITAL<CRLF>
DIGITAL<CRLF>
DIGITAL<CRLF>
DIGITAL<CRLF>
OK>
```

[GPOTYPE] = This sub-command will return the type of output (digital or analog) for the output specified. If a specific output pin is requested, the type will be returned for each output pin. This command will return either DIGITAL or ANALOG. For example (assuming there are 4 outputs):

```
CAP GPOTYPE<CRLF>
DIGITAL<CRLF>
DIGITAL<CRLF>
DIGITAL<CRLF>
DIGITAL<CRLF>
OK>
```

[ANTENNAS] = This sub-command will return the number of antennas that are available on the device. For example:

```
CAP ANTENNAS<CRLF>
4<CRLF>
OK><CRLF>
CAP ANTENNAS 1<CRLF>
MONOSTATIC<CRLF>
OK><CRLF>
CAP ANENNAS 2<CRLF>
BISTATIC<CRLF>
OK><CRLF>
```

[TAGTYPE] = This sub-command will return the names of all supported tagtype values accepted by the TAGTYPE attribute. For example:

```
CAP TAGTYPE<CRLF>
MIXED<CRLF>
G1<CRLF>
G2<CRLF>
ISO5BG1<CRLF>
ISO6BG2<CRLF>
V119<CRLF>
UCODE119<CRLF>
ICODE119<CRLF>
EPCC1G2<CRLF>
OK<CRLF>
```

[FIELDSTRENGTH] = This sub-command will return either the minimum or maximum value supported for the specified antenna index. If no antenna is specified, the requested value will be returned for all antennas. The values returned are in dB. For example:

```
CAP FIELDSTRENGTH MAX 1<CRLF>
30DB<CRLF>
OK><CRLF>
CAP FIELDSTRENGTH MIN<CRLF>
15DB<CRLF>
15DB<CRLF>
15DB<CRLF>
15DB<CRLF>
OK><CRLF>
```

[DENSEREADERMODE] = This sub-command will return whether or not the reader supports dense reader mode. The response will either be TRUE or FALSE. For example:

```
CAP DENSEREADERMODE<CRLF>
TRUE<CRLF>
OK><CRLF>
```

[LISTENBEFORETALK] = This sub-command is used to determine the LBT capabilities of the reader. When called with no parameters, the reader will indicate whether or not LBT is supported by responding with either TRUE or FALSE.

If LBT is supported, the additional options of this sub-command will also be supported.

If LBT is not supported, all additional parameters for the LISTENBEFORETALK sub-command will return ERR.

An example of the LISTENBEFORETALK sub-command is shown below:

```
CAP LISTENBEFORETALK<CRLF>
TRUE<CRLF>
OK><CRLF>
CAP CHANNEL MAX<CRLF>
13<CRLF>
OK><CRLF>
CAP LISTENBEFORETALK SCAN<CRLF>
TRUE<CRLF>
OK><CRLF>
```

[SESSIONS] = This sub-command will return the maximum number of sessions that are supported on the current BRI connection transport (TCP or serial). Serial connections will only support one session, because it is a point-to-point transport. An example of the SESSIONS sub-command for TCP and Serial connections is shown below:

Example (TCP)

```
CAP SESSIONS<CRLF>
8<CRLF>
OK><CRLF>
```

Example (Serial)

```
CAP SESSIONS<CRLF>
1<CRLF>
OK><CRLF>
```

[EPCC1G2PARAMETERS] = This sub-command will return a list of the supported EPCC1G2 parameter sets. An alias for this command is EPCC1G2PARMS. For example:

```
CAP EPCC1G2PARMS<CRLF>
IDS=4,6,9,10<CRLF>
```

Each ID represents a value that can be assigned to the EPCC1G2PARAMETERS attribute to select a particular set of EPCC1G2 parameters.

You can include the ID parameter in the capabilities command and it will display the EPCC1G2 parameters for the selected parameter set. For example:

```
CAP EPCC1G2PARMS 10<CRLF>
BLF=320KHz
MILLER=4<CRLF>
TAI=17<CRLF>
OK><CRLF>
```

[NXP] = This sub-command returns the NXP command extensions supported by the BRI interface. If a reader does not support the NXP extensions, no commands will be returned.

DIAGNOSTICS

Purpose This command is available for an application that wants to determine certain runtime characteristics of an RFID reader.

Command Shortcut `DIAG`

Syntax `DIAGNOSTICS [LASTREADCOUNT] [SESSIONS]`

Parameters [LASTREADCOUNT] = This sub-command will return the number of tag singulations that occurred during the last tag operation. If a tag operation is in progress (such as a continuous read), the number of tags singulated at the time DIAG LASTREADCOUNT is executed will be returned. For example:

```
READ<CRLF>
H300833B2DDD9014035050604<CRLF>
H300833B2DDD9014035050605<CRLF>
H300833B2DDD9014035050607<CRLF>
H300833B2DDD9014035050606<CRLF>
OK><CRLF>
DIAG LASTREADCOUNT
4
OK><CRLF>
```

[SESSIONS] = This sub-command will return the number of sessions that are currently in use on a particular transport. For information about determining the number of supported sessions, see the SESSIONS parameter in **"CAPABILITIES" on page 41**. Serial readers only support one session.

An example of the SESSIONS sub-command for TCP and Serial connections is shown below:

Example (TCP)

```
DIAG SESSIONS<CRLF>
3<CRLF>
OK><CRLF>
```

Example (Serial)

```
DIAG SESSIONS<CRLF>
1<CRLF>
OK><CRLF>
```

ERASE

Purpose: This command allows you to perform a block erase of EPC Gen 2 tags. The ERASE command must operate on even data lengths.

Syntax:
```
ERASE [flex_query_selector][DATA FIELD][TAGTYPE=<tagtype_list>]
[WHERE<data_condition>] [PASSWORD=<"access_password">]
```

Parameters: [flex_query_selector] = This parameter modifies the command for certain types of EPCC1G2 tags. For details, see **"FLEXQUERY" on page 47**.

[DATA FIELD] = This parameter can be any data type defined in **"Data Field Definitions" on page 22**. For more details, see **"Understanding [READ FIELD] and [WRITE FIELD] Parameters" on page 85**.

[TAGTYPE] = This parameter represents the type of tag that is written.

[WHERE] = This parameter can be any expression defined in **"Data Conditions" on page 28**.

[PASSWORD] = This specific keyword is used to specify a password to access the data fields that are locked.

Errors: The following errors are reported by the ERASE command.

Error 1:

```
ERASEERR<CRLF>
OK><CRLF>
```

This error indicates that hat the tag has failed to erase the data in the tag.

Error 2:

```
PVERR<CRLF>
OK><CRLF>
```

This error indicates that the data at the specified address has been previously locked.

Error 3:

```
PWERR<CRLF>
OK><CRLF>
```

This error only applies to EPC Global Class 1 Gen 2 tags and can be returned to any command that writes to a tag.

Examples:
```
ERASE STRING(18,5)
```

This ERASE command erases all data at address 18 on any tags currently in the reader field. An example of the response for a single tag is written is shown below:

```
ERASEOK<CRLF>
OK><CRLF>
```

FACDFLT

Purpose: The factory defaults are the set of attribute, macro, and trigger values defined by the manufacturer. The FACDFLT command resets all attributes, macros, and triggers to the original configuration. For a list of default values, see **"Default Factory Configuration" on page 3**.

The “FACDFLT ATTRIB” command allows the user to query the factory default values for all attributes. The “FACDFLT attribute_name” command allows the user to query the factory default value for the named attribute.

Syntax:

```
FACDFLT
FACDFLT ATTRIB
FACDFLT attribute_name
```

Example:

```
FACDFLT<CRLF>
OK><CRLF>
FACDFLT DENSEREADERMODE<CRLF>
DENSEREADERMODE=OFF<CRLF>
OK><CRLF>
FACDFLT ATTRIB<CRLF>
ANTS=1<CRLF>
TAGTYPE=EPCC1G2<CRLF>
FIELDSTRENGTH=30DB,30DB,30DB,30DB<CRLF>
RDTRIES=3<CRLF>
RPTTIMEOUT=0<CRLF>
IDTIMEOUT=100<CRLF>
ANTTIMEOUT=50<CRLF>
IDTRIES=1<CRLF>
ANTTRIES=3<CRLF>
WRTRIES=3<CRLF>
LOCKTRIES=3<CRLF>
SELTRIES=1<CRLF>
UNSELTRIES=1<CRLF>
INITTRIES=1<CRLF>
INITIALQ=4<CRLF>
SESSION=2<CRLF>
EPCC1G2PARAMETERS=11<CRLF>
SCHEDULEOPT=0<CRLF>
FIELDSEP=" "<CRLF>
CHKSUM=OFF<CRLF>
TIMEOUTMODE=OFF<CRLF>
NOTAGRPT=OFF<CRLF>
IDREPORT=OFF<CRLF>
TTY=OFF<CRLF>
DENSEREADERMODE=OFF<CRLF>
```

FLEXQUERY

Purpose: The FLEXQUERY selector is a field that can be added to any tag access command (read, write, protect, kill, etc.). It applies only to EPCC1G2 tag types. When the selector is included in the access command, the subsequent inventory operations will use the FLEXQUERY command rather than the standard EPCC1G2 QUERY command. FLEXQUERY is defined in the "ISO/IEC 18000-6 Second Edition" from December 1, 2010. It is intended primarily for use with battery assisted passive (BAP) and sensor tags. Tags that do not comply with this version of the ISO standard will not respond to FLEXQUERY.

Command Shortcut: `FLEXQUERY, FQ`

Syntax: `FLEXQUERY(<tag_type_selector>,<ss_reply>,<mrfid_reply>)`

Parameters: <tag_type_selector> = This parameter is a numeric value between 0 and 4095.

Interpretation	RFU	MRFID	Sensor Alarm	Full Function Sensor	Simple Sensor	RFU	RFU	RFU	Battery Assisted Passive	RFU	Passive NOTE 1 (RFU)
1	1	1	1	1	1	1	1	1	1	1	1
0: Inclusive		0: Disable	0: Disable	0: Disable	0: Disable				0: Disable		0: Disable
1: Exclusive NOTE 2		1: Enable as defined in ISO/IEC 29143	1: Enable	1: Enable	1: Enable				1: Enable		1: Enable

<ss_reply> = This parameter is a numeric value of zero or one. If one, it enables the simple sensor (SS) tag reply by setting the "SS Reply" bit in the FLEXQUERY command.

<mrfid_reply> = This parameter is a numeric value of zero or one. If one, it enables the Mobile RFID reply by setting the "MRFID Reply" bit in the FLEXQUERY command.

Examples: **Example 1:**

```
READ FQ(0x1c4,0,0)<CRLF>
HE280B0403C0000000C0269ED<CRLF>
OK><CRLF>
```

Example 2:

```
WRITE FQ(0x1c4,0,1) HEX(3:0,2)=h0000<CRLF>
HE280B0403C0000000C0269ED WROK<CRLF>
OK><CRLF>
```

HELP

Purpose: This command displays a list of all the BRI commands supported by the reader.

Syntax: `HELP`

Example: To show the list of available BRI commands, use this command:

```
OK>HELP<CRLF>
```

HWCC

This command returns the country code information that is stored in the reader device. This code is used to determine the region text returned from the HWREGION command. For more information, go to **"HWREGION" on page 48**.

HWID

Purpose: This command returns a unique identifier that represents the reader module. The response is a line of characters of arbitrary length determined by the reader.

Note: For the IF61 or 70 Series RFID, you can view the serial number of the module printed on the label of the device.

Syntax: `HWID`

Example: Readers with serial BRI in the module (IM4, IM5, IM11-OEM) reports both the hardware configuration string and serial number:

```
OK>HWID<CRLF>
CN-IM11A200002<CRLF>
SN-JRW00000000<CRLF>
OK>
```

Network readers report only the configuration string:

```
OK>HWID<CRLF>
CN-IF2A400014<CRLF>
OK>
```

HWPROD

Purpose: This command returns the product name of the reader.

Syntax: `HWPROD`

Example:

```
OK>HWPROD<CRLF>
IM5<CRLF>
OK>
```

HWREGION

Purpose: This command returns the region data for the reader.

Syntax: `HWREGION`

Example:

```
OK>HWREGION<CRLF>
FCC 915MHz CC014<CRLF>
OK>
```

Country Code Values and Region Data

Country Code	Region Data
CC002	ETSI 302 208 V1.2.1 865MHz CC002

Country Code Values and Region Data (continued)

Country Code	Region Data
CC003	ETSI 300-220 869 MHz CC003
CC004	FCC 915 MHz CC004
CC005	ETSI 300-220 869MHz CC005
CC008	South Korea 910 MHz CC008
CC012	ETSI 302-208 865MHz CC012
CC014	FCC 915MHz CC014
CC016	Taiwan 923MHz CC016
CC017	Australia 918MHz CC017
CC018	South Korea 910MHz CC018
CC019	Thailand 920MHz CC019
CC020	Australia 920MHz CC020
CC021	New Zealand 921Mhz CC021
CC022	Brazil 915MHz CC022
CC026	Hong Kong 910MHz CC026
CC027	Japan 953MHz CC027
CC028	Malaysia 919MHz CC028
CC030	Singapore 920MHz CC030
CC031	Singapore 920 MHz CC031
CC032	Asia 920MHz CC032
CC033	Indonesia 902MHz CC033
CC034	China 902MHz CC034
CC035	Philippines 918MHz CC035
CC036	South Africa 915MHz CC036
CC037	Israel 915MHz CC037

For information on how to return the CC value, see the HWCC command on **"HWCC" on page 48**.

If the BRI is unable to determine the country code for a particular module, the BRI will return the following message:

```
HWREGION<CRLF>
REGION UNKNOWN <CRLF>
OK><CRLF>
```

HWVER

Purpose: This command returns the board version level of the reader.

Syntax: `HWVER`

Example:

```
OK>HWVER<CRLF>
2B3<CRLF>
OK>
```

KILLTAG

Purpose: This command supports the EPC Class 1 Gen 2 KILL operation.

Syntax:
```
KILLTAG [flex_query_selector][TAGTYPE=<tagtype_list>]
[WHERE<data_condition>] [PASSWORD=<kill_password>]
```

Parameters: [flex_query_selector] = This parameter modifies the command for certain types of EPCC1G2 tags. For details, see **"FLEXQUERY" on page 47**.

[TAGTYPE] = This parameter represents the type of tag that is written.

[WHERE] = This parameter can be any expression defined in **"Data Conditions" on page 28**.

[PASSWORD] = This specific keyword is used to specify a password to access the data fields that are locked.

Examples
```
OK>KILLTAG WHERE EPCID=H3003000000FF8600000D056903
PASSWORD=H09FB6C13<CRLF>

OK>
```

LOCK

Purpose: This command is provided to maintain backward compatibility with previous BRI versions.

The command may only be used by ISO 18000-6B tags. If you are using EPC Global Class 1 Gen 2 tags, the LOCK command will function similarly to the WRITE command and will not protect any memory. In case lock control is required for other types of tags, use the command described on **"PROTECT" on page 52**.

Note: This command may eventually be removed when ISO 18000-6B tags are no longer supported. Applications are advised to start using the PROTECT command.

Syntax:
```
LOCK [flex_query_selector][DATA FIELD] [WRITE FIELD]
[TAGTYPE=<tagtype_list>] [WHERE <data_condition>]
[PASSWORD=<access_password>]
```

Parameters: [flex_query_selector] = This parameter modifies the command for certain types of EPCC1G2 tags. For details, see **"FLEXQUERY" on page 47**.

[DATA FIELD] = This parameter can be any data type defined in **"Data Field Definitions" on page 22**.

[WRITE FIELD] = This [WRITE FIELD] parameter consists of a list of data types that define the format of the data returned from a tag. Using data types INT, HEX, and STRING, specific data can be read from any memory location on a tag. Unlike the [READ FIELD] parameter, TAGID can be used in a [WRITE FIELD] parameter. The tag memory locations that store the tag identifier are locked at manufacturing time and cannot be changed. For details, see **"Understanding [READ FIELD] and [WRITE FIELD] Parameters" on page 85**.

[TAGTYPE] = This parameter represents the type of tag that is written.

[WHERE] = This parameter can be any expression defined in **"Data Conditions" on page 28**.

[PASSWORD] = This specific keyword is used to specify a password to access the data fields that are locked.

Errors: The following errors can be reported by the LOCK command.

Error 1:

```
LCKERR<CRLF>
OK><CRLF>
```

This error indicates that the tag failed to lock the data in the tag.

Error 2:

```
PVERR<CRLF>
OK><CRLF>
```

This error indicates that the data at the specified address has been previously locked.

Examples: `LOCK STRING(18,5)="HELLO"`

This LOCK command with a [WRITE FIELD] parameter will lock all tags found with the data `HELLO` starting at address 18 in the tag. An example of the response for a single tag is shown below:

```
LCKOK<CRLF>
OK><CRLF>
```

PING

Purpose: This command lets an application check if the reader is available and communicating as expected.

Note: The BRI PING command is not similar to the ICMP ping command.

Syntax: `PING<TIME | TIMESTAMP>`

Parameters: <TIMESTAMP> = This parameter allows you to get an approximate relation between the time stamp timer for the reader and the notion of absolute time by the application.

Example:

```
PING<CRLF>
OK><CRLF>
PING TIME<CRLF>
4535606
OK>
```

The returned time indicates that 4535.606 seconds have passed since the BRI session was started or the reader module was reset.

PLATDFLT

Purpose: This command resets all reader attributes to the platform default configuration. These values are different from the factory default settings. These settings are set through a platform configuration interface, such as SmartSystems.

Syntax: `PLATDFLT`

PRESET

Purpose: This command resets network readers like the IF2 and IF61. A reset performs a total software restart of the reader module. After the PRESET command is executed, the reader behaves as if external power was disconnected and then reconnected.

Note: For some readers like the IM4 or IM5, the PRESET command is the same as the RESET command. For details, see **"RESET" on page 61**. And, for other readers, like the IM11 installed in a CK70, the PRESET command is not supported and returns an ERR response.

Syntax: `PRESET<CRLF>`

PRINT

Purpose: This command lets you view the contents of a macro.

Command Shortcut: `PR`

Syntax: `PRINT $<NAME>`

Parameters: <NAME> = This parameter specifies the name of the macro.

Examples: **Example 1:**

This command causes the reader to return all attributes and command line parameter settings stored under the macro name MYREADMACRO:

```
PRINT $MYREADMACRO<CRLF>
```

The data is returned in a format that can be used in a subsequent BRI command.

Example 2:

The PRINT command can also be used to display any arbitrary strings.

```
PRINT MY STRING <CRLF>
MY STRING<CRLF>
OK><CRLF>
PRINT MY MACRO IS: $MYREADMACRO<CRLF>
MY MACRO IS: READ WHERE TAGID=HE2001050<CRLF>
OK><CRLF>
```

PROTECT

Purpose: This command lets you turn memory access locks ON or OFF in EPCglobal Gen 2 tags.

Use the ON keyword to lock and the OFF keyword to unlock.

Syntax:
```
PROTECT [flex_query_selector]<ON|OFF>[PERMANENT]<READ
FIELD>[TAGTYPE=<tagtype_list>]
[WHERE<data_condition>][PASSWORD=<access_password>]
```

Parameters: [flex_query_selector] = This parameter modifies the command for certain types of EPCC1G2 tags. For details, see **"FLEXQUERY" on page 47**.

[PERMANENT] = This reserved keyword specifies permanent locking and unlocking of memory access locks. If permanent lock flag is set, the specified memory bank can be read, but never be unlocked or written to again. If the permanent unlock flag is set, the specified memory bank can never be locked in the tag, and the data can always be read or written to at any time.

<READ FIELD> = This parameter can be any data type defined in **"Data Field Definitions" on page 22**. For more details, see **"Understanding [READ FIELD] and [WRITE FIELD] Parameters" on page 85**.

This parameter determines the part of the tag that is locked or unlocked. Note that when the PROTECT command is protecting memory bank 0, which is the password memory bank, it contains an ACCESS and a KILL password. Memory bank 0 contains 8 bytes of information and each password is 4 bytes long. These passwords can be protected independently. Also, if no memory bank is specified in the <READ FIELD>, the command will default to memory bank 1, the EPCID memory bank.

The following <READ FIELD> specifiers can be used to protect the ACCESS and KILL passwords:

HEX(0:4,4) represents the ACCESS password.

HEX(0:0,4) represents the KILL password.

Issuing the PROTECT command with these <READ FIELD> parameters will protect the specified password.

[TAGTYPE] = This parameter represents the type of tag that is written.

<data condition> = This parameter can be any expression defined in **"Data Conditions" on page 28**. The PROTECT command will operate on the set of tags determined by the [WHERE<data condition>] clause. If the WHERE clause is omitted, the command will operate on all tags that are in the field.

[PASSWORD] = This specific keyword is used to specify a password to access the data fields that are locked.

<access> = This is the EPC Class 1 Gen 2 access password expressed as a binary hexadecimal value. This password is required to execute a lock operation (LOCK, LOCKP, UNLOCK, UNLOCKP) and to execute a write operation on locked data.

A tag holds just one access password and therefore the same password applies to both operations executed on a tag.

The PROTECT command syntax allows the access password to be specified on a lock operation such as LOCK=<access> and a write operation such as PASSWORD=<access>. If both operations are executed then both access passwords must be equal.

(ON|OFF) = The ON keyword allows you to lock the specified memory bank. Once data in a memory bank is protected with the ON keyword, new data cannot be written to the locked memory bank.

The OFF keyword allows you to unlock the specified memory bank.

Errors: The following errors are reported by the PROTECT command.

Error 1:

```
PVERR<CRLF>
```

OK><CRLF>

PVERR indicates that an incorrect password has been supplied to the command, or that the memory space has already been locked using the ON keyword.

Error 2:

PWERR<CRLF>

OK><CRLF>

PWERR indicates that the tag does not have enough power to complete the PROTECT operation.

Examples: The examples below assume that the ACCESS password is H12345678, and the response to each command is LCKOK<CRLF>OK><CRLF>.

Example 1:

```
PROTECT ON HEX(1:4,12) PASSWORD=H12345678
```

This command will lock memory bank 1 on the EPC Global Gen 2 tag. Once it is locked, data can always be read, but new data cannot be written until a PROTECT OFF command is issued.

Example 2:

```
PROTECT ON PERMANENT HEX(1:4,12) PASSWORD=H12345678
```

This command will permanently lock memory bank 1 on the EPC Global Gen 2 tag. Once in, data can never be written into this memory space once protected with the PERMANENT keyword. Data can always be read from this memory bank 1 after execution of this command.

Example 3:

```
PROTECT OFF PERMANENT HEX(1:4,12) PASSWORD=H12345678
```

This command will permanently unlock memory bank 1 on the EPC Global Gen 2 tag. Data can always be read or written from or to memory bank 1 after execution of this command. Note that once a memory bank is permanently unlocked, any PROTECT command will be ignored if issues to the permanently unlocked memory bank.

Example 4:

Protecting the EPCID

```
WRITE HEX(0:4,4)=H12345678
PROTECT ON HEX(0:4,4) PASSWORD=H12345678
WRITE EPCID=H0102030405060708091011l2
PROTECT ON HEX(1:4,2) PASSWORD=H12345678
```

READ

Purpose: This command controls how tag information is collected and reported. The READ command supports two collection modes:

- In Singleshot mode, the reader executes the set number of IDTRIES or ANTTRIES, or continues reading until the IDTIMEOUT or ANTTIMEOUT timer expires, and returns all tags that are found. The number of tags found depends on the number of tags in the field. The singleshot mode may return NOTAG if no tags are present in the field of the reader and the NOTAGRPT

attribute is enabled. The duration of the read is determined by the attribute settings.

- In Continuous mode, the reader continuously collects tags and stores the tags in an internal tag list. If the tag list gets full, the oldest tag in the list is removed.

Singleshot mode is enabled by default. To enter continuous mode, you have to use the REPORT parameter.

Command Shortcut: `R or RD`

Syntax:

```
READ[flex_query_selector][data_field|literal]*
[ TAGTYPE=<tagtype_list>] [WHERE<data_condition>]
[PASSWORD=<access_password>] [REPORT=EVENT|NO|EVENTALL]
[STOP|POLL]
```

Parameters: [flex_query_selector] = This parameter modifies the command for certain types of EPCC1G2 tags. For details, see **"FLEXQUERY" on page 47**.

[data_field] = This parameter can be any data type defined in **"Data Field Definitions" on page 22**. For details, see **"Understanding [READ FIELD] and [WRITE FIELD] Parameters" on page 85**.

If the READ command does not contain a [READ FIELD], the BRI uses a default based on the tag type:

- For ISO 18000-6B tags, TAGID is used as the default.
- For all other tags, EPCID is used as the default.

[literal] = This parameter can be any text string surrounded by double quotes. The text string will be printed in the command response at the same point as it appeared in the command string, which lets you improve the readability of the data returned from the BRI. There is no limit to the number of [literals] you can include in a command. For details, see **"Understanding the [LITERAL] Parameter" on page 103**.

[TAGTYPE] = This parameter represents the type of tag that is written.

[WHERE] = This parameter can be any expression defined in **"Data Conditions" on page 28**.

[PASSWORD] = This specific keyword is used to specify a password to access the data fields that are locked.

[REPORT=*value*] = This reserved keyword determines if you are in Singleshot or Continuous mode:

- REPORT=DIRECT enables Singleshot. This value is equivalent to excluding a REPORT parameter.

- REPORT=EVENT enables Continuous mode with event messages. The collected tags are stored in the tag list, and a tag event message is immediately reported for each detected tag that is not already on the tag list. The event message is delayed if the RPTTIMEOUT attribute is set to a non-zero value; for help, see **"RPTTIMEOUT" on page 95**. The Tag event message is sent only once: when the tag is first added to the tag list. If a tag exits the field and is later identified again, no Tag event message is sent because the tag is already on the tag list. You can send the READ POLL command to clear the tag list. After you clear the tag list, if a tag is identified again, the tag is added to the tag list (because the list was empty) and the Tag event message is sent. If the tag list becomes full, the oldest tag is removed from the tag list.
- REPORT=NO enables Continuous mode with no event messages. The collected tags are stored in the tag list, but no event messages are reported. If the tag list becomes full, the oldest tag is removed from the list. You can send the READ POLL command to return the contents of the tag list.
- REPORT=EVENTALL enables Continuous mode with event messages. The collected tags are store in the tag list, and a tag event message is immediately reported for each detected tag that is not already on the tag list. In this mode, tags will continuously be reported as they are read.The event message is delayed if the RPTTIMEOUT attribute is set to a non-zero value; for help, see **"RPTTIMEOUT" on page 95**.

If the reader is in Continuous mode and a new command with REPORT=EVENT or REPORT=NO is received, the reader ends the collection process based on the previous command and restarts the collection process based on the new command.

[POLL] = You issue the READ POLL command with no other parameters. If the reader is in Continuous mode, the READ POLL command reports the contents of the tag list and removes each reported tag from the tag list. If the reader is in Singleshot mode, the READ POLL command returns OK>.

[STOP] = You issue the READ STOP command with no other parameters. If the reader is in Continuous mode, the READ STOP command clears the tag list, does not report the contents of the tag list, and enables Singleshot mode. If the reader is in Singleshot mode, no tags are reported, and the reader remains in Singleshot mode.

Note: The READ STOP command forces the reader to exit Continuous mode cleanly. All commands (except ATTRIBUTE, READ POLL, WRITEGPO, READGPI, TRIGGER, TRIGGERREADY, and PING) force the reader to exit Continuous mode.

Errors: These error codes are reported by the READ command.

Error 1:

```
ERR<CRLF>
OK><CRLF>
```

This error indicates that a general BRI error has occurred.

Error 2:

```
RDERR<CRLF>
OK><CRLF>
```

This error indicates that the data read from the tag was invalid.

Error 3:

```
MEMOVRN<CRLF>
OK><CRLF>
```

This error indicates that the address used in the read command was not available on the tag. This tag is only returned when the TAGTYPE attribute contains EPCC1G2.

Error 4:

```
DISPLAYERR<CRLF>
OK><CRLF>
```

This error indicates that the amount of data requested from the tag will not fit into the output buffer in the BRI service. Clients should execute additional read commands with smaller access requests. The BRI output buffer is 512 bytes.

Error 5:

The read command may also the error below when the NOTAGRPT attribute is ON.

```
NOTAG<CRLF>
OK><CRLF>
```

Examples: These examples demonstrate how to use the READ command.

Note: The INT, HEX, and STRING data types in the [READ FIELD] parameters in the some of these examples do not include the optional *memory_bank* parameter that applies only to EPCglobal Class 1 Gen 2 tags. For help understanding the memory bank, see the description of the *memory_bank* parameter on **"HEX(memory_bank:address, length)" on page 24**.

Example 1:

```
READ TAGID
```

This READ command with a TAGID parameter and no <DATA CONDITION> parameter finds all tags in the field and returns a tag ID for each tag found. Each tag ID is terminated with a <CRLF>. When all tags are returned, the BRI returns OK><CRLF>. If no tags are found, the BRI returns NOTAG<CRLF>OK><CRLF>. For example:

```
1234567890ABCDEFH<CRLF>
OK><CRLF>
```

Example 2:

```
READ TAGID WHERE TAGID=H1234567890ABCDEF
```

This READ command with a <DATA CONDITION> parameter defined above finds a tag with the specified tag identifier. If a tag with the specified TAGID is found, the BRI returns the tag identifier followed by OK><CRLF>. If the tag identifier is not found, the BRI returns NOTAG<CRLF>OK><CRLF>. For example:

```
1234567890ABCDEFH<CRLF>
OK><CRLF>
```

Reader attributes affect the command responses:

- If the IDREPORT attribute is enabled, each response is prefixed with the tag ID or EPC code. For help, see **"IDREPORT" on page 91**.

- If the NOTAGRPT attribute is disabled, the BRI returns OK><CRLF> instead of NOTAG<CRLF>OK><CRLF>. For help, see **"NOTAGRPT" on page 93**.

If you want to match any tag that contains a tag ID starting with H123456, you can use the following command:

```
READ TAGID WHERE TAGID=H123456??????????
```

The `??` wildcard character pair matches any character in that position. The question mark pairs must represent a two-character hex value:

- `TAGID=123??678` is not valid and causes an ERR response.
- `TAGID=12??5678` is valid.

Note: Wildcard characters do not work with EPC Global Gen 2 tags.

Also, if the TAGID being matched begins from the start of the TAGID information, you can leave off the question mark pairs. The following <DATA CONDITION> parameter is identical to the one above:

```
READ TAGID WHERE TAGID=H123456
```

The question mark pairs are implied by the command line parser since a TAGID data type requires up to 16 characters and only six are specified. If fewer than the required number of characters are supplied, implied question mark pairs are inserted to fill the remainder of the field data.

Example 3:

```
READ TAGID WHERE STRING(18,5)="HELLO"
```

This READ command with a <DATA CONDITION> parameter looks for data on the tag starting at tag memory address 18 through address 22 that matches the string `HELLO`. The command matches the text between the double quotes. The BRI returns the tag ID of each tag that contains this data string. Each tag that is found is followed by a <CRLF> sequence. When all tags are found matching the <DATA CONDITION> parameter, the BRI returns the tag identifier followed by OK><CRLF>. If no tags are seen matching this data, the BRI returns NOTAG<CRLF>OK><CRLF>. For example:

```
H1234567890ABCDEF<CRLF>
OK><CRLF>
```

Example 4:

```
READ INT(18,2)
```

This READ command with a [READ FIELD] parameter finds all tags and returns the integer value stored at tag memory addresses 18 and 19 on each tag found. The data is returned as a decimal integer value followed by a <CRLF> sequence. No tag ID information is returned in this case, only the data read from the specified address on each of the tags. When all tags are found and the data is returned, the BRI returns OK><CRLF>. If no tags are seen, the BRI returns NOTAG<CRLF>OK><CRLF>. If the [READ FIELD] parameter fails, the BRI returns RDERR<CRLF>OK><CRLF> for the result. For example:

```
1234<CRLF>
OK><CRLF>
```

If an error occurred in reading the data from the tag, the response is:

```
RDERR<CRLF>
OK><CRLF>
```

Example 5:

```
READ INT(18,2),INT(20,2)
```

This READ command with multiple [READ FIELD] parameters finds all tags and returns the data value stored at locations 18 and 19 and locations 20 and 21 on each tag found. Any [READ FIELD] parameter that fails to read returns RDERR for that parameter. When all the [READ FIELD] parameters and all tags have been processed, the BRI returns OK><CRLF>. If no tags are found, the BRI returns NOTAG<CRLF>OK><CRLF>.

Here are three example responses to this READ command.

This example response shows a successful read:

```
READ INT(18,2),INT(20,2)
1234 5678<CRLF>
OK><CRLF>
```

This example response shows an error reading the value `INT(10,2)`:

```
READ INT(18,2),INT(20,2)
1234 RDERR<CRLF>
OK><CRLF>
```

This example response shows how to add the [READ FIELD] TAGID to return the tag identifier for each tag read:

```
READ TAGID,INT(18,2),INT(20,2)
H1234567890ABCDEF 1234 5678<CRLF>
OK><CRLF>
```

Example 6:

```
READ "TAG ID:",TAGID
```

This READ command with a [LITERAL] parameter prints the string `TAG ID:` followed by a tag identifier and a <CRLF>. For example:

```
TAG ID:H1234567890ABCDEF<CRLF>
OK><CRLF>
```

Example 7:

```
READ INT(0:0,4) PASSWORD=H09FB6C13
```

This READ command accesses the EPCglobal Gen 2 password memory bank 0. This READ may require a password.

Example 8:

```
READ INT(0:0,4) WHERE EPCID=H3003000000FF8600000F0569
```

This read command accesses the EPCglobal Gen 2 password memory bank 0 that has not been locked. The <data condition> parameter specifies a unique EPCID to match for reading.

Example 9:

```
READ WHERE AFI=H95
```

This command only include tags where the AFI bits equal hexadecimal 0x95.

READGPI

Purpose: This command returns a value that indicates the current state of all the general purpose (GP) input lines available on a specific reader. The value returned is an integral representation of the bit field corresponding to the output pins available on the reader device. The READGPIO command is the same as the READGPI command, and has been included to maintain backwards compatibility with previous versions of the BRI. For details, see the documentation that shipped with the reader.

Note that regardless of the hardware, the BRI will return the following when all of the inputs on a reader are active:

```
READGPI<CRLF>
0<CRLF>
OK><CRLF>
```

Syntax:

```
READGPI<CRLF>
```

Examples: **Example 1:**

In this example, a reader has 2 general purpose outputs. The following table describes the possible values returned by the BRI for the READGPI and the associated states of each of the inputs.

Possible Values Returned for a Reader With 2 General Purpose Outputs

BRI Value	Pin 1 State	Pin 2 State
0	Active	Active
1	Active	Inactive
2	Inactive	Active
3	Inactive	Inactive

Example 2:

Use this command to check the state of all GP input lines in a reader:

```
READGPI<CRLF>
```

The response is formatted as follows for a reader with four GP input lines:

```
15<CRLF>
OK><CRLF>
```

The code indicates that all four inputs are ON.

REPEAT

Purpose: This command causes the last READ or WRITE command to be executed again.

When a READ or WRITE command is executed by the BRI, all command line parameters are stored in the reader. If a complex command has been sent, the REPEAT command provides a shortcut method of executing the command multiple times without sending the entire command sequence.

You can also use the REPEAT command to re-issue a previous READ or WRITE command that failed.

Command Shortcut: RPT

Syntax: REPEAT <VALUE><CRLF>

Parameters: <VALUE> = This parameter can be a number from 1 to 65534 and specifies the number of times to repeat the command. If <VALUE> is omitted, the REPEAT command will execute the last command one time. Also, if the last repeatable command was a continuous read, the repeat value is ignored and the command is executed once.

Examples: These examples demonstrate how to use the REPEAT command.

Example 1:

```
READ<CRLF>
H1234567890ABCDEF<CRLF>
OK><CRLF>
REPEAT 2<CRLF>
H1234567890ABCDEF<CRLF>
H1234567890ABCDEF<CRLF>
OK><CRLF>
```

Example 2:

```
ATTRIB IDTRIES;READ;ATTRIB IDTRIES<CRLF>
IDTRIES=1<CRLF>
H1234567890ABCDEF<CRLF>
IDTRIES=1<CRLF>
OK><CRLF>
REPEAT 2<CRLF>
IDTRIES=1<CRLF>
H1234567890ABCDEF<CRLF>
IDTRIES=1<CRLF>
IDTRIES=1<CRLF>
H1234567890ABCDEF<CRLF>
IDTRIES=1<CRLF>
OK><CRLF>
```

RESET

Purpose: This command performs a total software restart of the reader module.

In network readers like the IF2 and IF61, the RESET command only affects the embedded RFID reader module and does not reset the device itself. To completely reset the reader, use the PRESET command. For help, see **"PRESET" on page 52**.

Syntax: RESET<CRLF>

Examples: This command warm boots the reader:

```
RESET<CRLF>
OK><CRLF>
```

After the RESET command is executed, an event is reported to all current BRI sessions:

```
EVT:RESET<CRLF>
```

For help, see **"Understanding EVENT Messages" on page 110**.

SET

Purpose: This command lets you create both command macros and parameter macros, which are defined in **"Creating and Using BRI Macros" on page 113**. You can also use SET to list all macros stored in the non-volatile memory of the reader and to delete macros.

Syntax:

```
SET <NAME>="[BRI COMMAND],[READ FIELD]or[WRITE FIELD],
[LITERAL],[(ATTRIBUTE NAME=VALUE)...]TAGID,ANTENNA,
TIMESTAMP,WHERE <DATA CONDITION>"
```

Parameters: <NAME> = This parameter can be any alphanumeric string. There is no limit on the length of the <NAME> parameter, and it can be as short as one character. The first character must be a letter (A-Z or a-z). The first character cannot be a number (0-9). You cannot use BRI reserved keywords as macro names; for a complete list of reserved words, see **"Reserved Keywords" on page 17**.

If you do not include anything after the <NAME> parameter, the SET command displays all macros stored in the non-volatile memory of the reader.

If you do not specify anything after the equals sign, the SET command deletes the macro specified in the <NAME> parameter.

`"[BRI COMMAND],[READ FIELD]or[WRITE FIELD],
[LITERAL],[(ATTRIBUTE NAME=VALUE)...]TAGID,ANTENNA,
TIMESTAMP,WHERE <DATA CONDITION>"` = Everything you specify after the equals sign must be enclosed in one set of double quotes. Everything inside the double quotes is saved as the contents of the macro. If the macro contains [LITERAL] parameters, which must also be enclosed in double quotes, you must use the \ character to escape the embedded double-quotes around each [LITERAL] parameter.

To display all macros in memory, use the SET command with no parameters:

```
SET<CRLF>
<macro><CRLF>
<macro><CRLF>
OK><CRLF>
```

Examples: These examples demonstrate how to use the SET command.

Example 1:

The example below illustrates how to store a WHERE clause and TAGID parameter in a macro named MYREADMACRO.

```
SET MYREADMACRO="WHERE TAGID=H1234567890ABCDEF"<CRLF>
OK><CRLF>
```

To execute this macro definition with a READ command, use the command line below:

```
READ $MYREADMACRO<CRLF>
```

```
H1234567890ABCDEF<CRLF>
OK><CRLF>
```

Example 2:

This example includes the command and command line parameters.

```
SET YOURREADMACRO="READ TAGID WHERE INT (20,2)=2000"<CRLF>
OK><CRLF>
```

To execute this command, use the command line below:

```
$YOURREADMACRO<CRLF>
H1234512345123456<CRLF>
H2468246824682468<CRLF>
OK>
```

SWVER

Purpose: This command returns the current firmware version of the reader module.

Syntax: `SWVER`

Example:

```
OK>SWVER<CRLF>
9.04<CRLF>
OK>
```

TRIGGER

Purpose: This command creates, deletes, and displays trigger events that are based on the state of the GP inputs supported by the reader.

The maximum number of triggers you can create is 10. The available non-volatile memory must also be used for macros. If this memory space overflows, the BRI returns MERR.

Note: Intermec recommends that you do not create more than ten triggers.

Syntax:

```
TRIGGER [RESET|DELETEALL|<"NAME"> <GPIO|GPIOEDGE> <MASK>
<VALUE> FILTER<DELAY> [ACTION"MACRO_NAME"]]
```

Parameters: [RESET] = This reserved keyword completely resets the entire triggering system. This command deletes all triggers from memory and removes all trigger events from the event queue. This command returns the triggering system to a known state. You do not specify any other parameters when you issue a TRIGGER RESET command.

[DELETEALL] = This reserved keyword removes from memory all programmed triggers. This command does not delete queued trigger events from the reader. You do not specify any other parameters when you issue a TRIGGER DELETEALL command.

<NAME> = This parameter specifies the name of the trigger. You must enclose the name in double quotes. The affect of the <"NAME"> parameter depends on the optional parameters:

- When you include optional parameters such as GPIO or MASK, the trigger is created and stored in the reader with the name given in the <NAME> parameter. If a trigger already exists with the given name, it is updated with the new parameter information, and the trigger is reset if it is currently active.
- When you omit optional parameters, the trigger with the name specified is deleted.

[GPIO or GPIOEDGE MASK *value* FILTER *delay*] = These optional parameters are defined in the following description of how a trigger operates.

When a trigger has been created, it immediately enters the DETECTION state. In the DETECTION state, the GPIO inputs are scanned until a fire condition is detected:

- When GPIO is used, the fire condition occurs when the current GP input state masked (ANDed) with the *mask* parameter is equal to the *value*.
- When GPIOEDGE is used, the fire condition is the transition from the state where the input state masked (ANDed) with the *mask* parameter is not equal to the *value* to a state where the input state masked (ANDed) with the *mask* parameter is equal to the *value*.

When this input condition is true, a trigger event is stored on an internal queue and the trigger enters the FIRED state. The trigger remains in the FIRED state for the number of milliseconds given by the FILTER *delay* parameter, after which it re-enters the DETECTION state. If level triggering is used, when the DETECTION state is re-entered after expiration of the delay time, the input state may not have changed (the fire condition still exists) and that causes a new "firing" of the trigger.

GPIOEDGE trigger event messages, radio event messages, and tag event messages are sent directly without any internal queueing.

Trigger event messages from level triggered GPIO triggers are queued internally in the reader.

This queue has two states:

- BLOCKED
- READY

The queue is initially in the BLOCKED state. Every 200 milliseconds, the event queued is monitored to determine whether the queue is in the READY state and there is at least one event queued. If both conditions are met, then a queued event is reported asynchronously from the reader to the host and the queue is returned to the BLOCKED state. The host must issue a TRIGGERREADY command to transition the queue to the READY state. The reader stores up to ten events. If more than ten events are held, the oldest event is overwritten.

[ACTION] = This parameter allows the commands given by the macro to be executed upon detection of the fire condition.

For example, one trigger could start doing READs in continuous mode, while another event is also sent to the application about the trigger condition.This event could be used to inform the application to start reading the collected data. The application could then send a READ POLL followed by a READ STOP to retrieve the data collected since the initial trigger. This makes it possible to validate the contents of the macro, and if it is invalid, an ERR is returned by the TRIGGER command. Note that ACTIONs are neglected while the reader is in READER CONTINUOUS mode.

Examples: These examples demonstrate how to use the TRIGGER command.

The first two examples are set up as a dock-door solution. They define one trigger that waits for input 1 to go high and a second trigger that waits for input 2 to go high. The latter four examples show that trigger conditions can be configured to combine several inputs. The inputs 2 and 3 are used by these triggers and any combination of values on these generate a trigger event, but the event generated depends on the combined value.

Example 1:

```
TRIGGER RESET<CRLF>
OK><CRLF>
TRIGGER “Dock door #43” GPIOEDGE 1 1 FILTER 0<CRLF>
OK><CRLF>
TRIGGER<CRLF>
Dock door #43 GPIOEDGE 1 1 FILTER 0<CRLF>
OK><CRLF>
TRIGGER “Dock door #44” GPIOEDGE 2 2 FILTER 0<CRLF>
OK ><CRLF>
TRIGGER “Special 00” GPIOEDGE 12 0 FILTER 0<CRLF>
OK ><CRLF>
TRIGGER “Special 01” GPIOEDGE 12 4 FILTER 0<CRLF>
OK ><CRLF>
TRIGGER “Special 10” GPIOEDGE 12 8 FILTER 0<CRLF>
OK ><CRLF>
TRIGGER “Special 11” GPIOEDGE 12 12 FILTER 0<CRLF>
OK><CRLF>
```

Example 2:

The TRIGGER command with no parameters displays the currently programmed triggers as shown in the following example:

```
TRIGGER<CRLF>
Dock door #43 GPIOEDGE 1 1 FILTER 0<CRLF>
Dock door #44 GPIOEDGE 2 2 FILTER 0<CRLF>
Special 00 GPIOEDGE 12 0 FILTER 0<CRLF>
Special 01 GPIOEDGE 12 4 FILTER 0<CRLF>
Special 10 GPIOEDGE 12 8 FILTER 0<CRLF>
Special 11 GPIOEDGE 12 12 FILTER 0<CRLF>
OK><CRLF>
```

Example 3:

The TRIGGER DELETEALL command removes from memory all programmed triggers. This command does not delete queued trigger events from the reader.

```
TRIGGER DELETEALL<CRLF>
OK><CRLF>
TRIGGER<CRLF>
OK><CRLF>
```

Example 4:

The TRIGGER RESET command completely resets the entire triggering system. This command deletes all triggers from memory and removes all trigger events from the event queue. This command returns the triggering system to a known state.

```
TRIGGER RESET<CRLF>
OK><CRLF>
```

TRIGGERREADY Command

Responses: For details about EVENT messages, see **"Understanding EVENT Messages" on page 110**.

TRIGGERQUEUE

Purpose: This command lets an application determine if queued events are available. This command returns an integer value stating the number of currently queued trigger events available in the reader.

Command Shortcut: `TRIGGERQ`

Syntax: `TRIGGERQUEUE [FLUSH]<CRLF>`

Parameters: [FLUSH] = This reserved keyword deletes all trigger events from the queue.

Examples: These examples demonstrate how to use the TRIGGERQUEUE command.

Example 1:

This example shows that three trigger events are available in the queue, which means that three subsequent invocations of TRIGGERREADY can be sent without any blocking occurring:

```
TRIGGERQUEUE<CRLF>
3<CRLF>
OK><CRLF>
```

Example 2:

This example shows how to check how many trigger events are in the queue, remove all trigger events from the queue, and then check the queue again:

```
TRIGGERQUEUE<CRLF>
3<CRLF>
OK><CRLF>
TRIGGERQUEUE FLUSH<CRLF>
OK><CRLF>
TRIGGERQUEUE
0<CRLF>
OK><CRLF>
```

Responses: For details about EVENT messages, see **"Understanding EVENT Messages" on page 110**.

TRIGGERREADY

Purpose: This command is used by the application to transition the event queue for the reader to the READY state. When issued, this command enables the asynchronous reporting of the oldest trigger event in the event queue for the reader. Events are reported in the same order as they were originally queued.

If you send a TRIGGERREADY command, but no trigger event has been configured, the BRI ignores the command and returns OK><CRLF>.

Command Shortcut: `TRIGRDY`

Syntax: `TRIGGERREADY [CANCEL]<CRLF>`

Parameters: [CANCEL] = This reserved keyword terminates a TRIGGERREADY command that was issued if no event has occurred yet.

Responses: For details about EVENT messages, see **"Understanding EVENT Messages" on page 110**.

Examples: These examples demonstrate how to use the TRIGGERREADY command.

Example 1:

In this example, the reported GPIO state 11 (8+2+1) indicates that when the trigger fired, inputs 0, 1, and 3 were all high, and input 2 was low:

```
TRIGGERREADY<CRLF>
OK><CRLF>
EVT:TRIGGER Dock door #43 GPIO 11<CRLF>
```

Example 2:

In this example, you can use this command to terminate a TRIGGERREADY command that was issued if no event has occurred yet.

```
TRIGGERREADY CANCEL<CRLF>
```

TRIGGERWAIT

Purpose: This command is used by the application to transition the event queue for the reader to the READY state. The TRIGGERWAIT command is the same as the TRIGGERREADY command, and has been included to maintain backwards compatibility with previous versions of the BRI.

When issued, this command enables the asynchronous reporting of the oldest trigger event in the event queue for the reader. Events are reported in the same order as they were originally queued.

If you send a TRIGGERWAIT command, but no trigger event has been configured, the BRI ignores the command and returns OK><CRLF>.

Command Shortcut: `TRIGGERW, TRIGWAIT`

Syntax: `TRIGGERWAIT [CANCEL]`

Parameters: [CANCEL] = This reserved keyword terminates a TRIGGERWAIT command that was issued if no event has occurred yet.

Responses: For details about EVENT messages, see **"Understanding EVENT Messages" on page 110**.

Examples: These examples demonstrate how to use the TRIGGERWAIT.

Example 1:

In this example, the reported GPIO state 11 (8+2+1) indicates that when the trigger fired, inputs 0, 1, and 3 were all high, and input 2 was low:

```
TRIGGERWAIT<CRLF>
OK><CRLF>
EVT:TRIGGER Dock door #43 GPIO 11<CRLF>
```

Example 2:

In this example, you can use this command to terminate a TRIGGERWAIT command that was issued if no event has occurred yet.

```
TRIGGERWAIT CANCEL<CRLF>
```

VERSION

Purpose: This command displays general version information about the platform and reader module. This command has no parameters.

Command Shortcut: `VER`

Syntax: `VERSION`

Examples:

```
VERSION<CRLF>
IM5 RFID Reader Ver 9.11<CRLF>
Basic Reader Interface Version 3.01<CRLF>
FCC 915MHz CC014<CRLF>
Copyright (C) 2008 Intermec Technologies Corp.<CRLF>
OK>
```

WRITE

Purpose: This command stores user-specified information on the tag in the specified locations.

There are no limitations on ISO 18000-6B tags for the WRITE command.

For EPCglobal Class 1 Gen 2 tags, there is a limitation on the addresses and lengths of data that can be written. EPCglobal Class 1 Gen 2 tags support writing only to words or 16-bit values. As a result, you must write even-length values to even-byte addresses. This limitation is illustrated in the examples below.

The WRITE command may return NOTAG if no tags are present in the field of the reader and the NOTAGRPT attribute is enabled.

Command Shortcut: `W or WR`

Syntax:
```
WRITE [flex_query_selector] [WRITE_MODIFIER]<data_field> [[WRITE
MODIFIER]DATA FIELD]* [TAGTYPE=<tagtype_list>]
[WHERE<data_condition>] [PASSWORD=<access_password>]
```

Parameters: [flex_query_selector] = This parameter modifies the command for certain types of EPCC1G2 tags. For details, see **"FLEXQUERY" on page 47**.

[data_field] = This parameter can be any data type defined in **"Data Field Definitions" on page 22**. For more details, see **"Understanding [READ FIELD] and [WRITE FIELD] Parameters" on page 85**.

At least one data field is required for a write command to be executed. Data fields must include the value to be written.

To specify a value to be written, the data field should be followed immediately by an equals sign. See the examples section for examples.

[WRITE_MODIFIER] = This parameter can be used to select the type of write operation. This is for EPCC1G2 tags only. Valid write modifiers are BLOCK and STANDARD. BLOCK tells the reader to use EPCC1G2 block-write commands to write the data to the tag. STANDARD tells the reader to use standard EPCC1G2 write commands to write the data to the tag. If no modifier is specified then STANDARD is used. If a data field is specified without a modifier, the modifier from the preceding data field is used.

[TAGTYPE] = This parameter represents the type of tag that is written. This is a shortcut provided to eliminate the need for changing the TAGTYPE attribute for a single write command.

[WHERE] = This parameter can be any expression defined in **"Data Conditions" on page 28**.

[PASSWORD] = This specific keyword is used to specify a password to access the data fields that are locked.

Errors: These error codes are reported by the WRITE command.

Error 1:

`WRERR<CRLF>`

`OK><CRLF>`

This error indicates that the data being written to the tag failed.

Error 2:

```
MEMOVRN<CRLF>
OK><CRLF>
```

This error indicates that the address used in the WRITE command was not available on the tag. This is only returned when the TAGTYPE attribute contains EPCC1G2.

Error 3:

```
PVERR<CRLF>
OK><CRLF>
```

This error indicates that an incorrect password has been provided when writing to an EPCglobal Gen 2 tag, or the address being written has previously been locked.

Error 4:

```
PWERR<CRLF>
OK><CRLF>
```

This error indicates that the tag did not have enough power to complete the WRITE command when writing to an EPCglobal Gen 2 tag.

Examples: These examples demonstrate how to use the WRITE command.

Example 1:

```
WRITE INT(10,2)=3 HEX(12,2)=H12CD TAGTYPE=G2
```

The data field parameters specified above will write at tag memory address 10, and a value of 3 will be written. The data 0x12 and 0xCD will be written to locations 12 and 13 respectively.

Example 2:

```
WRITE STRING(18,5)="HELLO" TAGTYPE=G2
```

This WRITE command with a [WRITE FIELD] parameter writes all tags found with the data HELLO starting at address 18 in the tag. Each tag written responds with the status of the WRITE command terminated with a <CRLF>. When all tags are written, the BRI returns WROK<CRLF>OK><CRLF>. If no tags are written, the BRI returns NOTAG<CRLF>OK><CRLF>. Here is an example response when a single tag is written:

```
WROK<CRLF>
OK><CRLF>
```

For an EPCglobal Class 1 Gen 2 tag, this example returns the ADERR error response because the data to be written does not have an even length:

```
WRITE STRING(10,5)="HELLO"<CRLF>
ADERR<CRLF>
```

Here is a valid WRITE command for an EPCglobal Class 1 Gen 2 tag:

```
WRITE STRING(10,4)="GOOD"<CRLF>
WROK<CRLF>
OK><CRLF>
```

Example 3:

```
WRITE STRING(10,5)="HELLO" TAGTYPE=G2
WHERE TAGID=H1234567890ABCDEF
```

This WRITE command with a TAGID parameter in the WHERE clause and the specified [WRITE FIELD] parameter writes the text string HELLO starting at tag memory address 10 to the tag whose tag identifier matches that specified in the command. When the tag is found and written correctly, the BRI returns the status of the WRITE command terminated by a <CRLF>. When all tags are found and successfully written, the BRI returns WROK<CRLF>OK><CRLF>. If the tag was not found, the BRI returns NOTAG<CRLF>OK><CRLF>. Here is an example of a successful response to this WRITE command:

```
WROK<CRLF>
OK><CRLF>
```

If you want to write to any tag that has a tag ID starting with H123456, you can use the following command:

```
WRITE STRING(18,5)="HELLO" WHERE TAGID=H123456??????????
WROK<CRLF>
OK><CRLF>
```

For an EPCglobal Gen 2 tag the proper valid command, you can use the following command:

```
WRITE STRING(10,4)="TEST" WHERE TAGID=H12345678<CRLF>
WROK<CRLF>
OK><CRLF>
```

Example 4:

```
WRITE STRING(10,4)="GOOD" WHERE STRING(10,3)="BAD"
```

This WRITE command with a <DATA CONDITION> and a [DATA FIELD] parameter looks for tags with data on a tag starting at address 10 through address 12 that matches 3 characters of the string BAD. The BRI writes the data GOOD starting at address 10 through address 13.

The BRI returns WROK<CRLF>OK><CRLF> for each successfully written tag. If an error occurs during the WRITE command, the BRI returns WRERR in the field that failed to write. If no tags are seen matching the data condition, the BRI returns NOTAG<CRLF>OK><CRLF>. Here is an example successful response to this WRITE command:

```
WROK<CRLF>
OK><CRLF>
```

Example 5:

```
WRITE STRING(10,4)="GOOD",STRING(14,2)="OK"
```

This WRITE command with two [WRITE FIELD] parameters writes two strings to all tags. After writing the data GOOD starting at address 10 and OK starting at address 40, the BRI returns the status of the WRITE command when the last write has completed. The BRI returns WRERR if any of the data was not successfully written or with WROK if the write was successful. If no tags are seen matching the data condition, the BRI returns NOTAG<CRLF>OK><CRLF>.

Here are two example responses to this WRITE command.

In this example, both fields were correctly written:

```
WROK WROK<CRLF>
OK><CRLF>
```

In this example, the second write (writing FINE starting at address 30) failed:

```
WROK WRERR<CRLF>
OK><CRLF>
```

Example 6:

```
WRITE TAGID STRING(18,4)="GOOD",STRING(30,4)="FINE"
```

This WRITE command presents two command parameters that can be used with the WRITE command. Normally, TAGID is associated with the READ command or a <DATA CONDITION>. However, you might want to know which tags have been written. Specifying TAGID on the command line causes the BRI response to return the tag identifier for each tag that is written.

The BRI command shown above writes the data GOOD starting at address 18 and FINE starting at address 30 and responds with the tag identifier of any tags written, followed by WROK for the successfully written fields, followed by OK><CRLF> when the last write has completed. If an error occurs during the write command, the BRI returns WRERR for the field that was not successfully written. If no tags are seen matching the data condition, the BRI returns NOTAG<CRLF>OK><CRLF.

One other write error that can occur is a privilege error. This error is caused by a reader attempting to write to tags that it does not own. Each reader that has privileges enabled is only capable of writing to owned tags. If a privilege error is encountered, the BRI returns PVERR in place of WRERR.

Here are three example responses to this WRITE command.

In this example response, both fields were successfully written:

```
H1234567890ABCDEF WROK WROK<CRLF>
OK><CRLF>
```

In this example response, there was an error writing FINE to the tag:

```
H1234567890ABCDEF WROK WRERR<CRLF>
OK><CRLF>
```

In the this example response, a privilege error was encountered and no data was written to the tag:

```
H1234567890ABCDEF PVERR PVERR<CRLF>
OK><CRLF>
```

Note: The INT, HEX, and STRING data types in the [WRITE FIELD] parameters in the some of the previous examples do not include the optional *memory_bank* parameter that applies only to EPCglobal Class 1 Gen 2 tags. For help understanding the memory bank, see the description of the *memory_bank* parameter on **"HEX(memory_bank:address, length)" on page 24**.

Example 7:

```
WRITE STRING(3:18,4)="TEST" TAGTYPE=EPCC1G2 PASSWORD=H9F5BE634
```

This WRITE command writes EPC Class1 Gen2 User Memory bank that was previously locked.

Example 8:

```
WRITE EPCID=H300300FB9600000010260090 PASSWORD=H9F5BE634
```

This WRITE command writes EPCglobal Gen 2 the EPCID memory bank 1 that was previously locked. It assumes that there is mechanism in place that restricts just one tag in the field of view of the reader to insure that multiple tags are not created with the same EPCID.

Example 9:

```
WRITE HEX(3:0,2)=h0000 BLOCK HEX(3:2,2)=H0000 HEX(3:4,2)=H0000
STANDARD HEX(3:6,2)=H0000
```

This WRITE uses a standard EPCC1G2 write command to write to location (3:0,2). It uses an EPCC1G2 block-write command to write to (3:2,2) and (3:4,2) and then switches back to a standard write for (3:6,2).

WRITEGPO

Purpose: This command sets the state of all the GP output lines available on a specific reader. Each reader has a unique set of GP output lines that are described in the documentation shipped with the reader. The WRITEGPIO command is the same as the WRITEGPO command, and has been included to maintain backwards compatibility with previous versions of the BRI.

```
OK><CRLF>
```

Syntax: `WRITEGPO<[=VALUE]> | [PIN <ON|OFF>]>`

Command Shortcut: `WRGPI`

Parameters: <VALUE> = This parameter specifies a number between 0 and 15, assuming a maximum number of four GP output lines. A value of 0 turns all lines ON, and a value of 15 turns all lines OFF.

Bitwise Outputs

The <VALUE> parameter is an integral representation of the bit field corresponding to the output pins available on the reader device.

Note that regardless of hardware, the BRI will enable all of the outputs with the following command is entered:

```
WRITEGPO=0<CRLF>
```

Direct Pin Outputs

When an output pin is specified, the BRI is capable of modifying the state of an individual output without an application having to calculate the bit mask (this calculation is impossible to do when multiple applications are modifying the output, as it is not possible to read the output state).

Examples: **Example 1:**

In this example, a reader has 2 general purpose outputs. The following table describes the possible values returned by the BRI for the WRITEGPO and the associate states of each of the outputs.

Possible Values Returned for a Reader With 2 General Purpose Outputs

BRI Value	Pin 1 State	Pin 2 State
0	On	On
1	On	Off
2	Off	On
3	Off	Off

Example 2:

The following example turns on the first output on a device, and then turns it off.

```
WRITEGPO 1 ON<CRLF>
OK><CRLF>
WRITEGPO 1 OFF<CRLF>
OK><CRLF>
```

BRI Extensions for NXP Tags

This section lists BRI extensions that are specific to NXP RFID tags.

NXPALARM

Purpose: An extension to the READ command. When a READ command is executed with NXPALARM, the reader executes a read operation where only NXP tags with the EAS bit set are identified. This bit can be enabled when using the NXPEAS command.

Note: No other data fields can be requested when executing the READ command with the NXPALARM extension.

Responses are not generated for each tag that is identified with the EAS bit enabled. Only a single alarm message will be generated, even if multiple NXP tags are identified with the EAS bit set.

If EVENT is specified as the report type, an NXPALARM event is generated once an NXP tag with the EAS bit is identified.

If EVENTALL is specified, an NXPALARM event is continuously generated while an NXP tag with the EAS bit set is in the reader field.

The READ STOP command should be used to stop the READ NXPALARM operation when in continuous mode.

Syntax:

```
READ [flex_query_selector]NXPALARM [TAGTYPE=EPCC1G2]
[REPORT=EVENT|EVENTALL]
```

Examples:

```
READ NXPALARM
ALARM<CRLF>
OK><CRLF>

READ NXPALARM REPORT=EVENT
OK><CRLF>
EVT:NXP ALARM<CRLF>

READ NXPALARM REPORT=EVENT
OK><CRLF>
EVT:NXP ALARM<CRLF>
EVT:NXP ALARM<CRLF>
EVT:NXP ALARM<CRLF>
EVT:NX PALARM<CRLF>
[...]
```

NXPCONFIG

Purpose: An extension to the WRITE command. WRITE NXPCONFIG modifies the 16 NXP configuration bits in the memory of a tag. For more information, see the NXP documentation.

To modify the NXPCONFIG field of a tag, a non-zero access password is required, and no other writable data fields may be specified. If an incorrect password is entered, WRERR may be returned.

Syntax:

```
WRITE [flex_query_selector]NXPCONFIG=<hex value> [,<hex
value>]*[TAGTYPE=EPCC1G2] [WHERE <data_condition>]
PASSWORD="password"
```

Examples: These examples are equivalent.

Example 1:

```
WRITE NXPCONFIG=H0000 NXPCONFIG=H0000 PASSWORD=h12345678
```

Example 2:

```
WRITE NXPCONFIG=H0000,H0000 PASSWORD=h12345678
```

NXPEAS

Purpose: An extension to the WRITE command. WRITE NXPEAS is used to control the EAS EEPROM of an NXP tag. When the EAS system bit of an NXP tag is set, the tag responds to the custom NXPALARM command. For more information, see the **"NXPALARM" on page 74**.

Syntax:

```
WRITE [flex_query_selector]NXPEAS=<ON |OFF> [TAGTYPE=EPCC1G2]
[WHERE <data_condition>] PASSWORD="password"
```

NXPREADPROTECT

Purpose: An extension to the WRITE command. WRITE NXPREADPROTECT prevents the reading of the EPCID by returning all zeros.

Syntax:
```
WRITE [flex_query_selector]NXPREADPROTECT=<ON|OFF>
[TAGTYPE=EPCC1G2] [WHERE <data_condition>] PASSWORD="password"
```

BRI Extensions for Fujitsu Tags

This section lists BRI extensions that are specific to Fujitsu RFID tags.

FJAREAREADLOCK

Purpose: An extension to the WRITE command. WRITE FJAREAREADLOCK implements the Fujitsu AreaReadLock command. AreaReadLock specifies the AreaReadLock status in the control memory of the tag.

Syntax:
```
WRITE
[flex_query_selector]FJAREAREADLOCK(<group>,<action>,<mask>,<pas
sword>) [TAGTYPE=<tagtype_list>] [WHERE <data_condition>]
```

<group> = An integer from 0 to 31.

<action> = An integer from 0 to 65535.

<mask> = An integer from 0 to 65535.

<password> = A 32-bit block group password.

Example:
```
WRITE FJAREAREADLOCK(0,0xffff,0xffff,0x11223344)<CRLF>
H3035307B2831B380B0000C8F LCK0K<CRLF>
OK><CRLF>
```

FJAREAWRITELOCK

Purpose: An extension to the WRITE command. WRITE FJAREAWRITELOCK implements the Fujitsu AreaWriteLock command. AreaWriteLock specifies the AreaWriteLock status in the control memory of the tag, and requires a password.

Syntax:
```
WRITE[flex_query_selector]
FJAREAWRITELOCK(<group>,<action>,<mask>,<password>)
[TAGTYPE=<tagtype_list>] [WHERE <data_condition>]
```

<group> = An integer from 0 to 31.

<action> = An integer from 0 to 65535.

<mask> = An integer from 0 to 65535.

<password> = A 32-bit block group password.

Example:
```
READ FJAREAWRITELOCK(0,0xffff,0xffff,0x11223344)<CRLF>
H3035307B2831B380B0000C8F LCK0K<CRLF>
OK><CRLF>
```

FJAREAWRITELOCKWOPWD

Purpose: An extension to the WRITE command. WRITE FJAREAWRITELOCKWOPWD implements the Fujitsu AreaWriteLockWoPwd command. AreaWriteLockWoPwd specifies the AreaWriteLockWoPwd status in the control memory of the tag.

Syntax:
```
WRITE
[flex_query_selector]FJAREAWRITELOCKWOPWD(<group>,<action>)
[TAGTYPE=<tagtype_list>] [WHERE <data_condition>]
```

<group> = An integer from 0 to 31.

<action> = An integer from 0 to 65535.

Example:
```
WRITE FJAREAWRITELOCKWOPWD(0,0xffff)<CRLF>
H3035307B2831B380B0000C8F LCK0K<CRLF>
OK><CRLF>
```

FJBURSTERASE

Purpose: FJBURSTERASE implements the BurstErase command which erases (writes "0") to multiple words in the memory of a tag. If an attempt is made to erase locked memory, PVERR is returned.

Syntax: FJBURSTERASE [flex_query_selector](<bank>:<address>, <length>) [, (bank:<address>,<length>)]* [TAGTYPE=<tagtype_list>] [WHERE <data_condition>] [PASSWORD=<access_password>]

Examples: **Example 1:**
```
FJBURSTERASE (3:0,128)<CRLF>
H3035307B2831B380B0000C8F ERASEOK<CRLF>
OK><CRLF>
```
Example 2:
```
FJBURSTERASE (3:128,128)<CRLF>
H3035307B2831B380B0000C8F PVERR<CRLF>
OK><CRLF>
```

FJBURSTWRITE

Purpose: An extension to the WRITE command. WRITE FJBURSTWRITE implements the Fujitsu BurstWrite command. FJBURSTWRITE is a write modifier, such as BLOCK and STANDARD. <data_field> uses the same syntax as for other BRI write commands. FJBURSTWRITE applies to <data_fields> of type INT, STRING, and HEX. The BurstWrite command operates on even word addresses and even word lengths while BRI uses byte addresses and byte lengths. The offsets and lengths specified in <data_field> must be evenly divisible by 4 to satisfy the requirements of the BurstWrite command.

If FJBURSTWRITE attempts to write to locked memory PVERR is returned.

Syntax:
```
WRITE [flex_query_selector]FJBURSTWRITE <data_field>
[data_field]* [TAGTYPE=<tagtype_list>] [WHERE <data_condition>]
[PASSWORD=<access_password>]
```

Examples: **Example 1:**

```
WRITE FJBURSTWRITE HEX(3:0,4)=H12345678<CRLF>
H3035307B2831B380B0000C8F WROK<CRLF>
OK><CRLF>
```

Example 2:

```
WRITE FJBURSTWRITE HEX(3:128,4)=H12345678<CRLF>
H3035307B2831B380B0000C8F PVERR<CRLF>
OK><CRLF>
```

FJCHGAREAGROUPPWD

Purpose: An extension to the WRITE command. WRITE FJCHGAREAGROUPPWD implements the Fujitsu ChgAreaGroupPwd command, and changes the password of a given group.

Syntax: `WRITE [flex_query_selector]FJCHGAREAGROUPPWD(<group >,<oldpassword>,<newpassword>) [TAGTYPE=<tagtype_list>] [WHERE <data_condition>]`

<group> = An integer from 0 to 31.

<oldpassword> = A 32-bit block group password.

<newpassword> = A 32-bit block group password.

Examples:

```
WRITE FJCHGAREAGROUPPWD (0,0x0,0x11223344)<CRLF>
H3035307B2831B380B0000C8F WROK<CRLF>
OK><CRLF>
```

FJCHGBLOCKGROUPPWD

Purpose: An extension to the WRITE command. WRITE FJCHGBLOCKGROUPPWD implements the Fujitsu ChgBlockGroupPwd command, and changes the password of a given block group.

Syntax: `WRITE [flex_query_selector]FJCHGBLOCKGROUPPWD(<group >,<oldpassword>,<newpassword>) [TAGTYPE=<tagtype_list>] [WHERE <data_condition>]`

<group> = An integer from 0 to 31.

<oldpassword> = A 32-bit block group password.

<newpassword> = A 32-bit block group password.

Examples:

```
WRITE FJCHGBLOCKGROUPPWD (0,0x0,0x11223344)<CRLF>
H3035307B2831B380B0000C8F WROK<CRLF>
OK><CRLF>
```

FJCHGBLOCKLOCK

Purpose: An extension to the WRITE command. WRITE FJCHGBLOCKLOCK implements the Fujitsu ChgBlockLock command. ChgBlockLock changes the BlockLock flags in a BlockGroup with a password.

Syntax:
```
WRITE [flex_query_selector]
FJCHGBLOCKLOCK(<group>,<action>,<mask>,<password>)
[TAGTYPE=<tagtype_list>] [WHERE <data_condition>]
```

<group> = An integer from 0 to 31.

<action> = An integer from 0 to 65535.

<mask> = An integer from 0 to 65535.

<password> = A 32-bit block group password.

Examples:
```
WRITE FJCHGBLOCKLOCK (0,0xffff,0xffff,0x11223344)<CRLF>
H3035307B2831B380B0000C8F LCKOK<CRLF>
OK><CRLF>
```

FJCHGWORDLOCK

Purpose: An extension to the WRITE command. WRITE FJCHGWORDLOCK implements the Fujitsu ChgWordLock command. This command changes one of two WordLock flags using the associated block group password.

Syntax:
```
WRITE [flex_query_selector]FJCHGWORDLOCK (<bank>:<address>,
<password>)=<payload> [,
(<bank>:<address>,<password>)=<payload>]*
[TAGTYPE=<tagtype_list>] [WHERE <data_condition>]
```

<bank> = Memory bank, must be 3, other values are reserved for future use.

<address = An integer memory byte address (must be evenly divisible by 4).

<password> = A 32-bit block group password.

<payload> = An integer from 0 to 15.

Examples:
```
WRITE FJCHGWORDLOCK (0,0x11223344)=0xf<CRLF>
H3035307B2831B380B0000C8F LCKOK<CRLF>
OK><CRLF>
```

FJREADBLOCKLOCK

Purpose: An extension to the READ command. READ FJREADBLOCKLOCK implements the Fujitsu ReadBlockLock command. ReadBlockLock read the BlockLock flags in a BlockGroup.

Syntax:
```
READ [flex_query_selector]FJREADBLOCKLOCK(<group>) [,
FJREADBLOCKLOCK(<group>)] [TAGTYPE=<tagtype_list>] [WHERE
<data_condition>]
```

<group> = An integer from 0 to 31.

Examples:
```
READ FJREADBLOCKLOCK(0)<CRLF>
H3035307B2831B380B0000C8F HOOOO<CRLF>
H3035307B2831B380B0000C90 HFFOO<CRLF>
OK><CRLF>
```

BRI Extensions for Impinj Monza 4 Tags

This section lists BRI extensions that are specific Impinj Monza 4 tags. The Impinj Monza 4 tags implement a 16-bit QT status field and a custom command to read and write the QT field.

The most significant bit (QT_SR) controls the range of the response of a tag. When set, the tag reduces the strength of its backscattered signal to reduce the range which the tag can be read.

The second significant bit (QT_MEM) controls whether the tag presents the public or private memory map to the reader. When set, the tag exposes its public memory map.

For more information on the Impinj Monza 4 tags, see the product specification for Impinj Monza 4 tags.

READ IMPINJQT

Purpose: Use this command to read the QT status. The result is a 16-bit hexadecimal value representing the QT status bits from the tag.

Syntax:
```
READ [flex_query_selector]IMPINJQT [TAGTYPE=<tagtype list>]
[WHERE <data condition>] [PASSWORD=<"access_password">]
```

Example: In this example, the first tag has both the QT_SR and the QT_MEM bits cleared (set for normal-range and private memory map). The second tag has both bits set (set for short-range and public memory map).

```
READ IMPINJQT
H300833B2DDD90140000000000000000000 H0000
H300833B2DDD90140000000000000000001 HC000
```

WRITE IMPINJQT

Purpose: Use this command to write to the QT status field.

Syntax:
```
WRITE [flex_query_selector]IMPINJQT=H<Data> {PERMANENT}
[TAGTYPE=<tagtype list>] [WHERE <data condition>]
[PASSWORD=<"access_password">]
```

<Data> = Contains a 16-bit value representing the contents of the QT status field to be written. The most significant bit is the QT_SR which controls the short-range/ normal-range setting of the tag. The next most significant bit is the QT_MEM bit which controls the public/private memory map setting of the tag.

Example: **Example 1:**

```
WRITE IMPINJQT=H0000
H300833B2DDD90140000000000000000000 WROK
```

Temporarily set the tag for normal range and private memory map.

Example 2:

```
WRITE IMPINJQT=HC000
H300833B2DDD90140000000000000000000 WROK
```

Temporarily set the tag for short range and public memory map.

Example 3:

```
WRITE IMPINJQT=HC000 PERMANENT
H300833B2DDD90140000000000000000000 WROK
```

Permanently set the tag for short range and public memory map.

BRI Extensions for EM Microelectronics Tags

The EM4325 tag from EM Microelectronics implements the following custom tag command extensions.

READ EMM_GETUID

Purpose: Use this command to get the UID from the tag with a single command. The UID can be 64, 80, or 96 bits long depending on the tag. The EMM_GETUID is a data field that can be used in a READ command.

Syntax:

```
READ [flex_query_selector] EMM_GETUID [TAGTYPE=<tagtype list>]
[WHERE <data condition>] [PASSWORD=<"access_password">]
```

Example:

```
READ EMM_GETUID
H300833B2DDD90140000000000000000000 HE280B0403C0000000C0269ED
```

READ EMM_GETSENSOR

Purpose: This command allows a reader to get the UID and sensor data from the tag with a single command. The UID can be 64, 80, or 96 bits long depending on the tag. The EMM_GETSENSOR is a data field that can be used in a READ command.

Syntax:

```
READ [flex_query_selector] EMM_GETSENSOR(<SendUID>,<NewSample>)
[TAGTYPE=<tagtype list>] [WHERE <data condition>]
[PASSWORD=<"access_password">]
```

<SendUID> = Integer value of 0 or 1 which indicates whether the tag should include its UID in the response.

<NewSample> = Integer value of 0 or 1 which indicates whether the tag should initiate a new measurement or just return the result of the last measurement.

Examples:

Example 1:

```
READ EMM_GETSENSOR(0,0)
H300833B2DDD9014000000000 H07F8000000000000
```

Example 2:

```
READ EMM_GETSENSOR(1,0)
H300833B2DDD9014000000000
HE280B0403C0000000C0269ED07F8000000000000
```

Example 3:

```
READ EMM_GETSENSOR(0,1)
H300833B2DDD9014000000000
HE280B0403C0000000C0269ED07F8000000000000
```

WRITE EMM_SENDSPI

Purpose: This command provides the interface to support SPI master operation of the EM4325 chip. It allows the reader to send SPI commands to an SPI slave device attached to the EM4325. The EMM_SENDSPI is a data field that can be used in a WRITE command.

Syntax:
```
WRITE [flex_query_selector] EMM_SENDSPI(<SPI_response_size>,
<SPI SCLK>, <SPI initial delay>, <SPI byte delay>)= H<Data>
[TAGTYPE=<tagtype list>] [WHERE <data condition>]
[PASSWORD=<"access_password">]
```

<SPI_response_size>, <SPI SCLK>, <SPI initial delay>, <SPI byte delay> = Numeric values corresponding directly to the parameters in the SPI packet portion of the command packet.

Examples: **Example 1:**

```
WRITE EMM_SENDSPI(0,0,0,0)=H1234
H300833B2DDD90140000000000000000000 WOK
```

Example 2:

```
WRITE EMM_SENDSPI(2,0,0,0)=H1234
H300833B2DDD90140000000000000000000 H5678
```

WRITE EMM_RESETALARMS

Purpose: This command allows a reader to reset the tag alarm conditions for Aux, Under Temp, and Over Temp. The EMM_RESETALARMS is a data field that can be used in a WRITE command.

Syntax:
```
WRITE [flex_query_selector] EMM_RESETALARMS [TAGTYPE=<tagtype
list>] [WHERE <data condition>] [PASSWORD=<"access_password">]
```

Example:
```
WRITE EMM_RESETALARMS
H300833B2DDD90140000000000000000000 WROK
```

Generic Tag Access Command

The generic tag access command can be used to invoke any vendor custom EPCC1G2 command that meets the following guidelines:

- The command is valid in the open or secured state.
- The command is formatted as follows:

	Command Code	Command Date	RN	CRC-16
# Bits	variable	variable	16	16
Details	Normally 16 bits - E0XX	0 or more	handle	

- When the command executes successfully, the tag responses as follows:

	Header	Data	RN	CRC-16
# Bits	1	variable	16	16
Details	0	0 or more	handle	

- When the command executes unsuccessfully, the tag responses as follows:

	Header	Error Code	RN	CRC-16
# Bits	1	8	16	16
Details	1	error code	handle	

- If the tag reply takes longer than T1, the reply uses the extended preamble as if TRext=1 regardless of the TRext value in the Query that initiated the inventory round.
- If the tag reply occurs within T1, the reply uses the preamble as determined by TRext in the Query that initiated the inventory round.

ACCESSCMD

Purpose: This command sends a custom tag access command to a tag and reports the resulting tag response. The ACCESSCMD is a data field that can be used in a READ command.

The result displays the hexadecimal value of the response with the handle and CRC-16 bits stripped. The result data is left justified. That is, the most significant (left most) bit of the command field represents the first bit to be transmitted to the tag.

Syntax:

```
READ [flex_query_selector] ACCESSCMD(<command_length>,
<response_length>, <milliseconds>)=H<command> [TAGTYPE=<tagtype
list>] [WHERE <data condition>] [PASSWORD=<"access_password">]
```

<command_length> = The number of bits in the command excluding the handle and the CRC-16.

<response_length> = The number of bits expected in the successful response to the command. This length does not include the handle and CRC-16.

<milliseconds> = The maximum number of milliseconds that the tag takes to respond to the command. For a standard EPCC1G2 write operation, this value is 20 indicating that the tag responds within 20 milliseconds. Intermec readers currently support a maximum wait time of about 33 milliseconds. When this value is zero, the reader expects the tag to respond within T1 with the preamble as specified in the Query that started the inventory round. When this value is non-zero, the reader expects the tag to respond with the extended preamble as if TRext=1.

<command> = The command data. The reader appends the tag's handle and the CRC-16. The command data is left justified. That is, the most significant (left most) bit of the command field represents the first bit to be transmitted to the tag.

Examples: **Example 1**

This example demonstrates the use of a standard EPCC1G2 read command (code C2) to read the first word of the EPCID of the tag. The equivalent BRI read command would be "r hex(1:4,2)".

```
READ ACCESSCMD(26,17,0)=HC2408040
H3005FB63AC1F3681EC88046A H180280
READ HEX(1:4,2)
H3005FB63AC1F3681EC88046A H3005
```

The following table shows the derivation of the command data C2408040. The command is 26 bits long, 8 Command Code, 2 MemBank, 8 WordPtr, and 8 WordCount. The handle and CRC are omitted from the command line and automatically appended to the command by the reader. The command must be comprised of an even number of bytes. In this case, C240804 would be enough to specify all the pertinent information but the value must be padded with an extra zero (C2408040) to make it an even number of bytes.

	Command Code	Membank	WordPtr	WordCount	RM	CRC-16
# Bits	8	2	EBV	8	16	16
Details	11000010 (Read)	01 (EPC)	00000010	00000001	handle	

The following table shows the derivation of the tag response as reported by the BRI. Note that the data value of 3005 is the same as the equivalent "r hex(1:4,2)" but it is shifted right by the header bit to be reported as 180280. The reader does not report the handle and CRC-16 from the response.

	Header	Data	RN	CRC-16
# Bits	1	16	16	16
Details	1	00110000000101	handle	

Example 2

```
READ ACCESSCMD(26,17,0)=HC2408800
H3005FB63AC1F3681EC88046A H8180
```

Command:

	Command Code	Membank	WordPtr	WordCount	RM	CRC-16
# Bits	8	2	EBV	8	16	16
Details	11000010 (Read)	01 (EPC)	00000010	10000000 (too many)	handle	

Error response:

	Header	Error Code	RN	CRC-16
# Bits	1	8	16	16
Details	1	00000011 (memory overrun)	handle	

Understanding [READ FIELD] and [WRITE FIELD] Parameters

Error checking occurs when a command is executed. If the syntax of the parameter is invalid, an error is reported. If the error occurs during the execution of a command, the appropriate response is returned from the BRI.

When reading or writing data from or to a tag, the entire address space of the tag is available.

For example, suppose a [READ FIELD] parameter was specified as `INT(126,4)` and the maximum tag memory address was 127. When the BRI executes this command, it returns RDERR because this [READ FIELD] parameter is asking to read four memory locations starting at address 126 (trying to read locations 126, 127, 128, and 129). This would try to read data two tag memory locations past the end of tag memory.

[READ FIELD] Examples

To read a STRING value 15 characters long at address 30:

```
READ STRING(30,15)
```

To read a HEX value eight characters long at address 100:

```
READ HEX(100,8)
```

To read a four-byte INT value on the tag at address 18:

```
READ INT(18,4)
```

To read a STRING value of five characters, a HEX value of seven characters, an INT value four-bytes long, and an INT value located at addresses 20, 25, 32, and 36 respectively:

```
READ STRING(20,5),HEX(25,7),INT(32,4),INT(36,1)
```

[WRITE FIELD] Examples

Use this command to write a STRING value that is eleven characters long to address 18 with the data `HELLO WORLD`:

```
WRITE STRING(18,11)=”HELLO WORLD”
```

Use this command to write a four-byte INT value to address 23 with data `1,234,567`:

```
WRITE INT(23,4)=1234567
```

Use this command to write three one-byte INT values and two 2-byte INT values to addresses 20, 30, 40, 50, and 52 with data the `A`, `10`, `20`, `600` and `2000` hexadecimal, respectively:

```
WRITE NT(20,1)='A',INT(30)=10,INT(40,1)=20,INT(50,2)=600,
INT(52,2)=H2000
```

Understanding the <ATTRIBUTE NAME> Parameter

This section describes the reader attributes you specify in the <ATTRIBUTE NAME> parameter in ATTRIBUTE commands. For help using the ATTRIBUTE command, see **"ATTRIBUTE" on page 36**.

You can use this command to display the current values for the reader attributes on your reader:

```
ATTRIBUTE<CRLF>
```

The BRI returns a list of attributes that varies based on the current setting for the TIMEOUTMODE attribute:

- If TIMEOUTMODE is enabled, then ANTTIMEOUT and IDTIMEOUT appear in the list.
- If TIMEOUTMODE is disabled, then ANTTRIES and IDTRIES appear in the list. (TIMEOUTMODE is disabled by default.)

The reader attributes are all reserved keywords, as listed in **"Keywords Reserved for Reader Attributes" on page 20**.

ANTENNAS or ANTS

Sets the antenna sequence to be used during READ and WRITE commands. This parameter is reader-dependent because different readers have different numbers of antennas.

```
ATTRIBUTE ANTS=n,n,n,n,n,n,n,n
```

where *n* is a number between 1 and 4 which identifies an antenna.

You can specify up to eight *n* values. This attribute is limited to values up to the number of antennas supported on the reader. Default is 1.

This example tells the reader to turn on the second antenna:

```
ATTRIBUTE ANTS=2
```

This example tells the reader to read or write tags using the antenna pattern specified:

```
ATTRIBUTE ANTS=1,2,1,3,1,4,1<CRLF>
```

ANTTIMEOUT

Sets the maximum time period (ms) during which each antenna is used for a READ or WRITE command. The maximum value for this attribute is 65534. Default is 50.

For help setting this attribute, see **"Understanding the Timeouts and Tries" on page 99**.

Note: This attribute applies only if TIMEOUTMODE is enabled. Otherwise, ANTTRIES applies.

ANTTRIES

Sets the number of times each antenna is used for a READ or WRITE command. Range is 1 to 254. Default is 3.

For example:

```
ATTRIBUTE ANTTRIES=4
```

For help setting this attribute, see **"Understanding the Timeouts and Tries" on page 99**.

Note: This attribute applies only if TIMEOUTMODE is disabled. Otherwise, ANTTIMEOUT applies.

BAUD

Note: If your BRI application communicates with the reader over a TCP connection, you cannot modify this attribute from the default. For details, see **"BRI TCP Applications" on page 3**.

Sets the reader baud rate for serial readers. You can set this parameter to 19200, 38400, 57600, or 115200. Default is 115200.

For example:

```
ATTRIBUTE BAUD=115200<CRLF>
OK><CRLF>
```

The baud rate change takes effect after the OK><CRLF> is transmitted. This allows the host time to change its baud rate before sending the next BRI command.

BROADCASTSYNC

Enables or disables the transmission of BroadcastSync commands. BROADCASTSYNC commands are sent periodically to distribute the current time to the tags. The time is determined by the value of the UTCTIME attribute. The value represents the number of seconds between BroadcastSync commands. If the value is zero, BroadcastSync commands will not be sent. If the value is non-zero, the commands will be sent periodically when the reader is identifying tags (read, write, erase, etc). When the reader is reset, this attribute defaults to zero. The user application must reset this value anytime the reader powers up or anytime the reader issues a BRI EVT:RESET. The range of value is 0 to 65535. Default is 0.

BTPWROFF

Sets the time period (in seconds) for which the Bluetooth radio (if available) will search for a Bluetooth connection. If no connection is created during the configured time period, the Bluetooth radio will be turned off completely to save power in mobile applications. The range of value is 30 to 3600 seconds. Default is 300 seconds.

CHANNEL

Determines the transmit channel that will be used for non-hopping 915 MHz readers. This attribute is not available on other readers. Attempts to query or set this attribute on readers that do not support CHANNEL will return an error. The range and default depend on the configured country code.

CHKSUM

Note: If your BRI application communicates with the reader over a TCP connection, you cannot modify this attribute from the default. For details, see **"BRI TCP Applications" on page 3**.

Configures the reader to include a checksum value in returned data. The checksum is two characters sent just prior to the command delimiter. For more details, see **"Using Message Checksums" on page 16**.

You can set this attribute to ON or OFF. Default is OFF.

This example enables transmission of the response checksum data:

```
ATTRIBUTE CHKSUM=ON<CRLF>
```

This example disables transmission of the response checksum data:

```
ATTRIBUTE CHKSUM=OFFC9<CRLF>
```

Note: If the CHKSUM attribute is enabled (`CHKSUM=ON`), the TTY and ECHO attributes are automatically disabled (`TTY=OFF` and `ECHO=OFF`).

DENSEREADERMODE

Enables or disables the dense reader mode operation. This mode only applies when the TAGTYPE list contains only the EPCglobal Class 1 Gen 2 tagtype. These tags respond with Miller Sub carrier encoded data instead of FM0 encoded data. You can set this attribute to ON or OFF.

For the IP30 and IM11, the default is ON.

For all other readers, the default is OFF.

ECHO

Note: If your BRI application communicates with the reader over a TCP connection, you cannot modify this attribute from the default. For details, see **"BRI TCP Applications" on page 3**.

Allows each command character to be returned back to the host device. This attribute can be helpful if you are typing BRI commands. You can set this attribute to ON or OFF. Default is OFF.

This example enables command echoing to the host device:

```
ATTRIBUTE ECHO=ON
```

This example disables command echoing to the host device:

```
ATTRIBUTE ECHO=OFF
```

Note: If the CHKSUM attribute is enabled, you cannot enable ECHO. If you attempt to enable the ECHO attribute, the BRI returns ERR.

EPCC1G2PARAMETERS

This numeric attribute allows the user to select a set of EPCC1G2 protocol parameters. The value must be one of the IDs listed in the responses to CAP EPCC1G2PARAMETERS. An alias for this attribute is EPCC1G2PARMS. For example:

```
ATTRIB EPCC1G2PARMS=1<CRLF>
OK><CRLF>
ATTRIB EPCC1G2PARMS<CRLF>
EPCC1G2PARMS=1<CRLF>
OK><CRLF>
```

FIELDSEP

This attribute allows the user to change the output format character used in the BRI. The default output field separator is the ASCII space character (0x20). Default is the space character. Note that this attribute can be modified when using a TCP connection.

The following table shows the values that are allowed for the FIELDSEP attribute.

FIELDSEP Attribute Values

Common Name	BRI Representation
Space	" "
Comma	","
Colon	":"
Semicolon	";"
Tab	"\t"
Caret	"^"
Tilde	"~"

The following example shows how to change the BRI output character to a tab:

```
ATTRIB FIELDSEP="\t"
```

FIELDSTRENGTH

Controls the RF power level for each antenna. You can specify a different value for each antenna between 15 and 30 dB. Default is 30 dB.

Two input modes are supported for the FIELDSTRENGTH attribute, you can enter a FIELDSTRENGTH value using a numeric scale or dB value. It is recommended that you use a dB value. The numeric scale is deprecated and is supported for backward compatibility of older versions of the BRI.

To enter values in dB, you must follow the numerical values with "dB". Valid values are 15 to 30 dB. For example:

```
OK>ATTRIBUTE FIELDSTRENGTH=30dB, 15dB, 16dB, 25dB<CRLF>
```

Reducing the signal by 3dB is equivalent to cutting the output power in half. For example, 27 dB is half as strong as 30 dB, and 24 dB is half as strong as 27 dB. 30 dB represents 1 watt of output power. Some regulatory domains limit the reader output power to less than 1 watt. If a reader is configured for one of these domains, entering 30 dB will result in maximum output power which will be less than 1 watt.

For entering in a numerical scale, the valid values are 25 to 100. You enter in these values in increments of 5. If you enter in values that are not multiples of 5, the value that was entered will be rounded down to the nearest multiple. Every 5 steps on the numerical scale is equivalent to 1 dB as shown in the following table.

Numerical Values and Their Equivalent dB Values

Numeric Value	Equivalent dB value
25	15 dB
30	16 dB
35	17 dB
40	18 dB
45	19 dB
50	20 dB

Numerical Values and Their Equivalent dB Values (continued)

Numeric Value	Equivalent dB value
55	21 dB
60	22 dB
65	23 dB
70	24 dB
75	25 dB
80	26 dB
85	27 dB
90	28 dB
95	29 dB
100	30 dB

IDREPORT

Configures the BRI to return tag identifiers when a command is executed. This applies to both ISO and EPC tag types:

- For ISO tags, the tag identifier corresponds to TAGID.
- For EPC tags, the tag identifier corresponds to EPCID.

You can set this attribute to ON or OFF. Default is OFF.

Note: This attribute default for the IF61 Fixed Reader is ON.

This example enables the display of each tag identifier for each READ or WRITE command:

```
ATTRIBUTE IDREPORT=ON
```

This example disables the display of each tag identifier for each READ or WRITE command:

```
ATTRIBUTE IDREPORT=OFF
```

IDTIMEOUT

Sets the maximum time period (ms) during which attempts are made to find tags before a response is returned to a READ or WRITE command. The maximum value for this attribute is 65534. Default is 100.

If you notice that all your antennas are not being used, it is possible that the reader does not have time to cycle through all the antennas before the timeout expires. You might want to increase the timeout.

For help setting this attribute, see **"Understanding the Timeouts and Tries" on page 99**.

Note: This attribute applies only if TIMEOUTMODE is enabled. Otherwise, IDTRIES applies.

IDTRIES

Sets the number of times an attempt is made to find tags before a response is returned to a READ or WRITE command. Range is 1 to 254. Default is 1.

For help setting this attribute, see **"Understanding the Timeouts and Tries" on page 99**.

Do not set the IDTRIES attribute to 0. The BRI returns ERR if you attempt to set the attribute to 0.

This example tells the RFID reader identify algorithm to execute three times in order to find all the tags that may be in the field:

```
ATTRIBUTE IDTRIES=3<CRLF>
```

Note: This attribute applies and only if TIMEOUTMODE is disabled. Otherwise, IDTIMEOUT applies.

INITIALQ

Establishes the initial Q parameter value used by the Query command for EPCglobal Class 1 Gen 2 tags only. Range is 0 to 15. Default is 4.

If you know that there is only one tag in the field, you should set this attribute to 0 for the best possible performance.

This attribute applies only to EPCglobal Class 1 Gen 2 tags.

INITTRIES

Sets the initialization tries variable in the reader. Range is 1 to 254. Intermec recommends that you leave this attribute set to the default, which is 1.

For help setting this attribute, see **"Understanding the Timeouts and Tries" on page 99**.

This example sets the INITRIES attribute to 1:

```
ATTRIBUTE INITTRIES=1<CRLF>
```

Do not set the INITTRIES attribute to 0. The BRI returns ERR if you attempt to set this attribute to 0.

LBTCHANNEL

Determines which channels will be the default transmit channel when executing the Listen Before Talk algorithm. LBTCHANNEL sets the default transmit channel of the available ETSI 302-208 channels. When you enable LBT scanning, the channel scan sequence starts with this LBT channel. When LBT scanning is disabled, (as in the 4 channel mode) the LBT channel is the only channel used. The range for 10 channel mode is 4 to 13. The default for 10 channel mode is 8, and for 4 channel mode the default is 7.

With 865 MHz readers, channels 4 to 13 are available. Default is 8.

With 953 MHz readers (special build), channels 1 to 9 are available. Default is 5.

LBTSCANENABLE

LBT scanning is enabled, by default in ETSI 10 Channel mode in accordance with 302-208.

Note: LBT scanning is permanently disabled in ETSI 4 channel mode in accordance with 302-208 v1.2.1.

When LBT scanning is enabled, the algorithm scans the available ETSI 302-208 channels for a free transmit channel.

In continuous read mode, the scan sequence begins with the channel specified by LBTCHANNEL and every third channel is checked (for example, 8, 11, 4, 7, 10, 13, 6, 9, 12, 5) until a free channel is found. If a free channel is not found, LBT repeats the scan sequence.

In single-shot read mode, LBT scanning goes through all available channels at once. If no free channel is found, the reader will report "NOTAG" and abort the inventory operation.

When LBT scanning is disabled, the reader does not scan for a free transmit channel, and the transmit channel is set by the LBTCHANNEL BRI attribute.

The valid values in 4 channel mode are 4, 7, 10, 13.

If LBTSCANENABLE is set to ON, the LBT algorithm will find a free channel from the 10 allowed.

If LBTSCANENABLE is set to OFF, the transmit channel will be set by the attribute LBTCHANNEL.

LOCKTRIES

Sets the number of times the lock algorithm is executed before a response is returned to a LOCK command. Range is 1 to 254. Default is 3.

This example tells the RFID reader lock algorithm to execute up to two times in order to lock data on the specified tags:

```
ATTRIB LOCKTRIES=2<CRLF>
```

NOTAGRPT

Allows the BRI to send a NOTAG message to notify you when no tags were found to operate on. You can set this attribute to ON or OFF. Default is OFF.

This example enables NOTAGRPT, so that the NOTAG message is sent:

```
ATTRIBUTE NOTAGRPT=ON
```

This example disables NOTAGRPT, so that no message is sent:

```
ATTRIBUTE NOTAGRPT=OFF
```

Note: By default, NOTAGRPT is disabled. However, the examples in this manual assume that the NOTAGRPT attribute is enabled. For details about other assumptions, see **"Conventions Used in This Manual" on page 5**.

QUERYSEL

Specifies the contents of the "Sel" field for EPCC1G2 "Query" commands. Valid values are from 0 to 4. Default is 4.

QUERYSEL Value Descriptions

Value	Description
0	All tags
1	All tags
2	Tags with SL not asserted
3	Tags with SL asserted
4	Reader determines Sel value

You can use this attribute with the EPCC1G2 Select Logic to select a sub-population of tags. For more information, see **"Using EPCC1G2 Select Logic in Data Conditions" on page 33**. However, for normal operations that do not use the EPCC1G2 Select Logic, leave this attribute set to the default value of 4.

QUERYTARGET

Specifies the "Target" field for EPCC1G2 "Query" commands. Valid values are A and B. Default is A.

QUERYTARGET Value Descriptions

Value	Description
A	Tags are reset to the A state (if UNSELTRIES is not equal to zero) and EPCC1G2 queries target tags in the A state.
B	Tags are reset to the B state (if UNSELTRIES is not equal to zero) and EPCC1G2 queries target tags in the B state.

You can use this attribute with the EPCC1G2 Select Logic to select a sub-population of tags. For more information, see **"Using EPCC1G2 Select Logic in Data Conditions" on page 33**. However, for normal operations that do not use the EPCC1G2 Select Logic, leave this attribute set to the default value of A.

RDTRIES

Sets the number of times an attempt is made to read data from a tag before a response is returned to a READ command. Range is 1 to 254. Default is 3.

This example tells the RFID reader to execute the read algorithm up to a maximum of five times on each [READ FIELD] specified in a READ command line:

```
ATTRIBUTE RDTRIES=5<CRLF>
```

RPTTIMEOUT

Sets the time period (ms) to delay the reporting of tag events when the reader is in Continuous mode (REPORT=EVENT). Range is 0 to 65534. Default is 0.

If RPTTIMEOUT is set to 0, there is no delay. A report is sent for each tag found in the ID round directly after the completion of the ID round.

If RPTTIMEOUT is set to a non-zero value, a timer is loaded with the value upon:

- a continuous operation starts, such as READ REPORT=EVENT command.
- a new RPTTIMEOUT value is set with the ATTRIBUTE command.
- a new report is available and the timer expired flag is set.

When the started timer expires, the timer is stopped and a new timer expired flag is set. The timer expired flag is cleared each time the time is started.

SCHEDULEOPT

Determines how antennas are switched during the inventory process. This attribute controls the behavior of the inventory scheduling parameters such as ANTTIMEOUT, ANTTRIES, IDTIMEOUT, and IDTRIES. This attribute can have a value of either 0, 1, or 2. Default is 0. **"Understanding the Timeouts and Tries" on page 99**, describes the behavior of timeout and tries attributes when SCHEDULEOPT is 0.

SCHEDULEOPT Value Descriptions

Value	Description
0	Legacy "normal" BRI operation
1	Simplified "normal" BRI operation
2	Simplified "normal" BRI operation with EPCC1G2 A/B toggling

When SCHEDULEOPT is set to 1, the TIMEOUT attribute is ignored. The ANTTIMEOUT and ANTTRIES attribute work together to determine how much time is spent on a given antenna. When ANTTIMEOUT is 0, the value of ANTTRIES defines the number of inventory rounds run on an antenna before stepping to the next antenna. When ANTTIMEOUT is not 0, the value of ANTTRIES is ignored and ANTTIMEOUT defines the amount of time spent on an antenna before stepping to the next antenna.

The IDTIMEOUT and IDTRIES attributes work together to determine how much time is spent trying to read tags. When IDTIMEOUT is zero, the value of IDTRIES defines the number of attempts used to read tags before a response is returned from a read or write command. When IDTIMEOUT is not zero, the value of IDTRIES is ignored and IDTIMEOUT defines the amount of time attempting to read tags before a response is returned.

When SCHEDULEOPT is set to 2, inventory behaves just like SCHEDULEOPT=1 except that any consecutive inventory rounds on the same antenna will toggle between targeting tags in the A state and tags in the B state. This value only applies to EPCC1G2 tags and is useful when dealing with tags that exhibit excessive persistence. For example, some battery powered EPCC1G2 tags have infinite persistence in sessions 1, 2, and 3. Once inventoried from A to B, these tags will stay in the B state and may not be identified if the reader does not run an inventory round targeting B tags.

SELTRIES

Sets the number of times a group select is attempted. A group select is the command used to start the identify process. Range is 1 to 254. Default is 1.

```
ATTRIBUTE SELTRIES=1
```

Note: This attribute can be used for EPCglobal Class 1 Gen 2 tags, but is not recommended.

SESSION

Sets the command session parameter to a corresponding EPCglobal Class 1 Gen 2 air protocol command. You can set this attribute to 0, 1, 2, or 3. Default is 2.

The following example specifies an EPCglobal Class 1 Gen 2 Select command:

```
ATTRIBUTE SESSION=0<CRLF>
```

For details about SESSION, see the EPCglobal Inc. specification, *EPC™ Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz–960 MHz Version 1.1.0*.

TAGTYPE

Defines the types of RFID tags used in an application. Certain performance improvements can be realized if it is known ahead of time what types of tags are in use. Follow this syntax:

```
ATTRIBUTE TAGTYPE=identifier[,identifier ...]
```

where *identifier* is one of the values listed in the next table.

TAGTYPE Identifiers for Each Air Protocol

Identifier	Air Protocol	Description
MIXED	ISO 18000-6B and UCODE119 only	G1, G2 and v1.19 tags only (MIXED mode is provided for backward compatibility.)
ISO6BG1 or G1	ISO 18000-6B	ISO6B G1 tags
ISO6BG2 or G2	ISO 18000-6B	ISO6B G2 tags
ICODE119	UCODE119	Phillips v1.19 tags (ISO 18000-6B emulating EPC tag IDs)

TAGTYPE Identifiers for Each Air Protocol (continued)

Identifier	Air Protocol	Description
UCODE119 or V119	UCODE119	Phillips v1.19 tags (ISO 18000-6B emulating EPC tag IDs)
EPCC1G1	EPCglobal Class 1 Gen 1	EPC global UHF Gen 1
EPCC1G2	EPCglobal Class 1 Gen 2	EPC global UHF Gen 2 tags (Default)
ATA	ATA	American Trucking Association (full frame)
ATAHALF	ATA	American Trucking Association (half frame)

Note: The identifiers G1 and G2 remain for backward compatibility. G1 is synonymous with ISO6BG1 and G2 is synonymous with ISO6BG2.

The tagtype capabilities of the reader can be queried with the CAP TAGTYPE command. You can also specify up to eight TAGTYPE *identifiers*. The order of the *identifiers* determines the order that the selected protocols are run. Do not specify the same air protocol twice as this will negatively impact performance.

The following example specifies that the application will use both ISO 18000-6B and EPC Class 1 Gen 2 tags:

```
OK> ATTRIBUTE TAGTYPE=ISO6BG2,EPCC1G2
```

The following example specifies that the application will use Phillips v1.19 tags (ISO 18000-6B emulating EPC tag IDs):

```
OK>ATTRIBUTE TAGTYPE=ICODE119
```

TIMEOUTMODE

Establishes whether to use “timeout” attributes or “tries” attributes. You can set this attribute to ON or OFF. Default is OFF.

- If TIMEOUTMODE is enabled, the IDTIMEOUT and ANTTIMEOUT attributes apply and are displayed when the ATTRIBUTE command is issued. This means that timeouts control how long the identify algorithm and antennas are used during each READ or WRITE command.
- If TIMEOUTMODE is disabled, the IDTRIES and ANTTRIES attributes apply and are displayed when the ATTRIBUTE command is issued. This means that counters control how many times the identify algorithm and antennas are used during each READ or WRITE command.

TTY

Note: When TTY is disabled, which is the default setting, only <LF> is recognized as a command terminator; <CR> is ignored. For details, see **“BRI TCP Applications” on page 3**.

Sets the command line delimiter used by the BRI. Cause the BRI server to echo BRI commands and to accept carriage-return as well as line-feed as end of line characters. This can be useful for manually typing BRI commands. You can set this attribute to ON or OFF. Default is OFF.

When TTY is enabled, be aware of these effects:

- The BRI command parser understands both the <CR> character (0x0D) and the <LF> character (0x0A) as line terminators. Therefore, any command followed by <CRLF> results in two OK> responses: the first is for the <CR> character, and the second is for the <LF> character.
- BRI command responses are terminated with OK>. The <CRLF> is omitted.
- For multi-line responses (for example, when multiple tags return after a READ command), each line up to the OK> line is always terminated by <CRLF> whether TTY is enabled or disabled.
- If both the TTY and CHKSUM attributes are enabled, the BRI automatically disables the CHKSUM attribute (`CHKSUM=OFF`).

When TTY is disabled, be aware of these effects:

- The BRI command parser understands only the `<LF>` character (0x0A) as the line terminator. For example, the HyperTerminal by default sends only `<CR>` to terminate a line. Therefore, if TTY is disabled and you are using HyperTerminal, you should select **Properties** > **Settings** > **ASCII Setup** and check the "Send line ends with line feeds" check box to ensure that every command is followed by a <LF> termination character. Or, you can press **Ctrl**-**J** (<LF>) to terminate a command, instead of pressing **ENTER** (<CR>).
- BRI command responses are terminated with OK><CRLF>.
- For multi-line responses (for example, when multiple tags return after a READ command), each line up to the OK> line is always terminated by <CRLF> whether TTY is enabled or disabled.

UNSELTRIES

Sets the number of times a group unselect is attempted. Range is 0 to 254. Default is 1.

```
ATTRIBUTE UNSELTRIES=1
```

UTCTIME

Sets the value of the UTC time that is sent in BroadcastSync commands. The time is in seconds since the epoch of January 1, 1970 at 00:00:00. The reader increments this attribute once per second so reading the value will return the current UTC time. This attribute will default to zero whenever the reader is reset. The user application should set the value from a reliable clock source anytime the reader powers up or anytime the reader issues a BRI EVT:RESET. Range is 0to 4294967296. Default is 0.

XONXOFF

This attribute enables or disables the flow control for a serial connection. The default value for XONXOFF is OFF.

Any changes to XONXOFF will take affect after the BRI has completed the response from the ATTRIBUTE command.

WRTRIES

Sets the number of times an attempt is made to write data to a tag on each [WRITE FIELD] before a response is returned to a WRITE command. Range is 1 to 254. Default is 3.

This example tells the RFID reader to execute the write algorithm up to a maximum of three times to write data to the specified tags:

```
ATTRIBUTE WRTRIES=3<CRLF>
```

Understanding the Timeouts and Tries

This section will help you determine how to set these attributes:

- ANTTIMEOUT
- IDTIMEOUT
- ANTTRIES
- IDTRIES
- INITTRIES

You need to know if the TIMEOUTMODE attribute is enabled or disabled:

- If TIMEOUTMODE is enabled, then you set ANTTIMEOUT and IDTIMEOUT. For help, see the next section, "Setting IDTIMEOUT and ANTTIMEOUT."
- If TIMEOUTMODE is disabled, then you set ANTTRIES, IDTRIES, and INITTRIES. For help, see **"Setting IDTRIES and ANTTRIES" on page 100** and **"Setting INITTRIES" on page 102**.

Setting IDTIMEOUT and ANTTIMEOUT

When you choose values for IDTIMEOUT and ANTTIMEOUT, you need to determine which attribute should have the larger value.

When IDTIMEOUT <= ANTTIMEOUT

If you set IDTTIMEOUT to a value smaller than or equal to ANTTIMEOUT, then ANTTIMEOUT is a flag to indicate that the reader spends X IDTIMEOUT on each antenna before moving to the next antenna. The total read time is approximately equal to IDTIMEOUT multiplied by the number of antennas selected.

Suppose these attributes are set:

- IDTIMEOUT=200
- ANTTIMEOUT=600
- ANTS=1,2
- INITTRIES=1

The reader performs these steps:

1. Turn on antenna one.
2. Read tags for 200 milliseconds.
3. Switch to the next antenna.
4. Read tags for 200 milliseconds.

The total ID time = 200 * 2 = 400 milliseconds.

When IDTIMEOUT > ANTTIMEOUT

If you set IDTIMEOUT to a value greater than ANTTIMEOUT, there is a difference between IDTIMEOUT and IDTRIES. Rather than keep track of how much time is spent on each antenna, the IDTIMEOUT is the total ID time. It does not matter how many antennas you select.

For example, if you set IDTIMEOUT to 500, the reader spends a total of 0.5 seconds looking for tags. The reader switches antennas if it spends ANTTIMEOUT without seeing any new tags. ANTTIMEOUT is reset when the reader switches antennas. In this scenario, it is possible that the reader will not use all the antennas.

Suppose these attributes are set:

- IDTIMEOUT=500
- ANTTIMEOUT=100
- ANTS=1,2,3,4
- INITTRIES=1

The reader performs these steps:

1. Turn on antenna one.
2. Reader finds tags during the first 200 milliseconds.
3. For the next 100 milliseconds, the reader sees no new tags.
4. Switch to antenna two.
5. Reader finds new tags for the next 200 milliseconds.
6. Reader has searched for a total of 500 milliseconds so it stops.

The total ID time = 500 milliseconds.

In this example, the reader never used antenna three and antenna four because the timeout expired before it had a chance to use them.

Setting IDTRIES and ANTTRIES

When you choose values for IDTRIES and ANTTRIES, you need to determine which attribute should have the larger value.

When IDTRIES < ANTTRIES

If you set IDTRIES to a value smaller than ANTTRIES, the reader executes all IDTRIES on antenna one before cycling to the next antenna. In this case, ANTTRIES functions as a flag to indicate this mode. Its actual value has no meaning. As long as ANTTRIES is larger than IDTRIES, this mode is active.

The total number of IDTRIES is IDTRIES multiplied by the number of selected antennas.

Suppose these attributes are set:

- IDTRIES=2
- ANTTTRIES=6
- ANTS=1,2
- INITTRIES=1

The reader performs these steps:

1. Turn on antenna one.
2. Read no new tags.
3. Read 1 new tag.
4. Switch to the next antenna.
5. Read no tags.
6. Read no tags.

The total number of ID tries = 2 * 2 = 4.

When IDTRIES >= ANTTRIES

If you set IDTRIES to a value greater than or equal to ANTTRIES, the reader performs a total of X IDTRIES on each antenna. However, if there are Y ANTTRIES in a row with no new tags found, the reader cycles to the next antenna. The reader comes back to any antenna that has not completed all X IDTRIES and finishes them using the same rules. The reader resets ANTTRIES whenever it switches antennas.

The total number of IDTRIES is equal to IDTRIES multiplied by the number of selected antennas.

Suppose these attributes are set:

- IDTRIES=4
- ANTTTRIES=2
- ANTS=1,2
- INITTRIES=1

In this example, the reader executes four ID tries per antenna. However, if it sees two ID tries in a row without any new tags, then the reader switches to the next antenna. The reader resets the counter for the next antenna.

The reader performs these steps:

1. Read 1 new tag.
2. Read no new tags.
3. Read no new tags.
4. Switch to the next antenna.
5. Read 2 new tags.
6. Read 4 new tags.

7. Read no new tags.
8. Read 1 new tag.
9. Switch back to antenna one.
10. Read no new tags.
11. Read 2 tags.

The total number of ID tries = 2 * 2 = 4.

Setting INITTRIES

When you choose the value for INITTRIES, you need to decide if you want INITTRIES to be 1 or greater than 1:

- The previous examples assume that INITTRIES is set to 1.
- This section describes how the reader operates when INITTRIES is greater than 1. In this case, the INITTRIES attribute tells the reader how many resets (initialize) commands should be sent. However, this does not mean how many are sent at one time. INITTRIES actually works as a multiplier of IDTRIES and ANTTRIES.

Suppose these attributes are set:

- INITTRIES=3
- IDTRIES=2
- ANTTTRIES=6
- ANTS=1,2

The reader performs these steps:

1. Turn on antenna one.
2. Send an initialize command on antenna 1 and 2.
3. Read no new tags.
4. Read 1 new tag.
5. Switch to the next antenna.
6. Read no tags.
7. Read no tags.
8. Send an initialize command on antenna 1 and 2.
9. Read 3 new tags.
10. Read 2 new tags.
11. Switch to the next antenna.
12. Read 1 tag.
13. Read 2 tags.
14. Send an initialize command on antenna 1 and 2.
15. Read 2 new tags.
16. Read 2 new tags.

17 Switch to the next antenna.

18 Read no tags.

19 Read no tags.

The total number of ID tries = 3 * (2 * 2) = 12.

You can see that INITTRIES worked to multiply the total number of cycles the reader ran. This is the case no matter how you set IDTRIES and ANTTRIES.

Note: You cannot set INITTRIES, IDTRIES, or ANTTRIES to zero.

Understanding the [LITERAL] Parameter

[LITERAL] parameters can improve the readability of the data returned from a BRI command. A [LITERAL] parameter is a quoted text string that can be placed anywhere on a BRI command line. There is no limit to the number of [LITERAL] parameters that you can specify. The maximum length of a [LITERAL] parameter is 255 characters.

This example demonstrates various uses of [LITERAL] parameters:

Command: `READ "PRICE $" ,INT(18,2,) "QUANTITY" ,INT(20,2), "SIZE", STRING(22,8)<CRLF>`

Response: `PRICE $25 QUANTITY 10 SIZE "LARGE"<CRLF>`

`OK><CRLF>`

Reading and Writing STRING Fields

When writing a string field to a tag, the length field specifies the total length of the string data. When data is returned from STRING fields, the data returned contains information only up to the length specified in the command.

The following five examples show the details associated with reading and writing STRING fields.

Consider the first example:

Suppose you want to define STRING data on a tag starting at location 18 for a length of 20 bytes. With a field of this length, the string can store 20 characters. This WRITE command would write the entire field:

`WRITE STR(18,20)="abcdefhgijklmnopqrst"<CRLF>`

The string is 20 characters long. This would fill the entire defined field in the tag.

Consider the second example:

If the data in a STR(address,length) data type is specified as follows, the characters `abcde` are written to locations 18 through 22. The characters `fg` are not written:

`WRITE STR(18,5)="abcdefg"<CRLF>`

Consider the third example:

This READ command reads the entire string field of the first example. The command stops reading after 20 characters.

```
READ STR(18,20)<CRLF>
"abcdefghijklmnopqrst"<CRLF>
OK><CRLF>
```

Consider the fourth example:

This READ command reads 25 characters from tag memory. The WRITE command in the first example wrote 20 ASCII characters up to location 37. The characters from 38 to 42 were not changed by that WRITE command. They may not represent printable ASCII. Characters that are not printable ASCII are returned as hex characters in the form \x*nn*. Suppose the characters on the tag are a period (.) followed by two carriage-return and line-feed characters. For example:

```
READ STR(18,25)<CRLF>
"abcdefghijklmnopqrst.\x0d\x0a\x0d\x0a"<CRLF>
OK><CRLF>
```

Consider the fifth example:

Smaller sections of this same data could be read out with this command:

```
READ STR(18,5)<CRLF>
"abcde"<CRLF>
OK><CRLF>
```

Understanding ACCESS and KILL Passwords

EPC Global Gen 2 tags contain a security feature that is controlled by the password values stored in memory bank 0. These ACCESS and KILL passwords that are contained in the EPC Global Gen 2 tags are 4 bytes in length:

- ACCESS passwords control how data is read or written to EPC Global Gen 2 tags. The ACCESS password is stored in the second 4 bytes of memory bank 0 and can be specified using the syntax `HEX(0:4,4)`.
- KILL passwords are used by the KILLTAG command to kill an EPC Global Gen 2 tag. The KILL password is stored in the first 4 bytes of memory bank 0 and can be specified using the syntax `HEX(0:0,4)`.

By default, the passwords in memory bank 0 are set to 0. When the password values contain a 0, the tags can be read and written by the PROTECT command. The READ, WRITE, PROTECT, and KILLTAG commands each have an option for specifying a password.

Once the ACCESS and KILL passwords have been set in a tag, the ability to read, write, protect, and kill tags requires the use of a password to complete the operation.

For ACCESS passwords, there are different rules in regards to read and write operations once the access password has been set.

Memory Bank 3 User Memory

For memory bank 3, if a non-zero ACCESS password has been written to a tag, you must specify an ACCESS password in the WRITE or PROTECT command in order to write or protect memory bank 3. If the correct password is not given, the commands will return a PVERR indicating that an invalid password has been given. The READ command can be performed without inputting a password.

The example below illustrates writing the ACCESS password first, then performing the READ, WRITE, and PROTECT operations.

```
WRITE HEX(0:4,4)=H11223344<CRLF>
WROK<CRLF>
OK>

READ HEX(3:0,4) PASSWORD=H11223344<CRLF>
H99001122<CRLF>
OK><CRLF>

READ HEX(3:0,4) PASSWORD=H55667788<CRLF>
PVERR<CRLF>
OK><CRLF>

WRITE HEX(3:0,4)=HAABBCCDD PASSWORD=H11223344<CRLF>
WROK><CRLF>
OK><CRLF>

WRITE HEX(3:0,4)=HAABBCCDD PASSWORD=H55667788<CRLF>
PVERR<CRLF>
OK><CRLF>

PROTECT OFF HEX(3:09,4) PASSWORD=H11223344<CRLF>
LCKOK<CRLF>
OK>

PROTECT ON HEX(3:0,4) PASSWORD=H55667788<CRLF>
PVERR<CRLF>
OK><CRLF>
```

Memory Bank 2 TID Memory

For memory bank 2, if a non-zero ACCESS password has been written to a tag, you must specify an ACCESS password in the READ, WRITE, or PROTECT command in order to read, write, or protect memory bank 3. If the correct password is not given, the commands will return a PVERR indicating that an invalid password has been given.

The example below illustrates writing the ACCESS password first, then performing the READ, WRITE, and PROTECT operations.

```
WRITE HEX(0:4,4)=H11223344<CRLF>
WROK<CRLF>
OK>

READ TAGID PASSWORD=H11223344<CRLF>
H66778899<CRLF>
OK><CRLF>

READ TAGID PASSWORD=H55667788<CRLF>
PVERR<CRLF>
OK><CRLF>

WRITE TAGID=HAABBCCDD PASSWORD=H11223344<CRLF>
WROK<CRLF>
OK><CRLF>

WRITE TAGID=HAABBCCDD PASSWORD=H55667788<CRLF>
PVERR<CRLF>

PROTECT OFF HEX(2:0,4) PASSWORD=H11223344<CRLF>
LCKOK<CRLF>
OK><CRLF>

PROTECT ON HEX(2:0,4) PASSWORD=55667788<CRLF>
PVERR<CRLF>
OK><CRLF>
```

Memory Bank 1 EPCID Memory

For memory bank 1, if a non-zero ACCESS password has been written to a tag, you must specify an ACCESS password in the WRITE command in order to write to memory bank 1. Note that in memory bank 1, the data can always be read with a correct ACCESS password or without an ACCESS password. If the correct password is not given, the commands will return a PVERR indicating that an invalid password has been given.

The example below illustrates writing the ACCESS password first, then performing the READ, WRITE, and PROTECT operations.

```
WRITE HEX(0:4,4)=H11223344<CRLF>
WROK<CRLF>
OK><CRLF>

READ EPCID<CRLF>
H112233445566778899001122<CRLF>
OK><CRLF>

READ EPCID PASSWORD=H11223344<CRLF>
H112233445566778899001122<CRLF>
OK><CRLF>

WRITE EPCID=H00112234455667788990011 PASSWORD=H11223344
WROK<CRLF>
OK><CRLF>

WRITE EPCID=H001122334455667788990011 PASSWORD=H55667788<CRLF>
PVER<CRLF>
OK><CRLF>

PROTECT OFF HEX(1:4,12) PASSWORD=H11223344<CRLF>
LCKOK<CRLF>
OK><CRLF>

PROTECT ON HEX(1:4,12) PASSWORD=H55667788<CRLF>
PVERR<CRLF>
OK><CRLF>
```

Memory Bank 0 PASSWORD Memory

If a non-zero ACCESS password has been written to an EPC global Gen 2 tag, the ACCESS password must be specified in the WRITE command in order to write to memory bank 0.

In memory bank 0, two unique passwords exist, the ACCESS and the KILL password. Even though they are located on the same memory bank, these passwords are treated independently. If the correct password is not supplied, the READ, RITE, and PROTECT commands will return PVERR indicated that an invalid password has been supplied.

If you are using the PROTECT ON PERMANENT option, both the ACCESS and KILL passwords can no longer be overwritten using the WRITE command as in the other memory banks. Also, the password data can no longer be read using the READ command. Permanently locking a password allows it to be completely hidden, and any attempts to use a WHERE clause to select tags based on stored information in memory bank 0 will be unsuccessful. This stops a user from finding out either byte by byte or bit by bit what the stored passwords contained.

The example below shows writing the ACCESS password first, then performing the READ, WRITE, and PROTECT operations.

```
WRITE HEX(0:4,4)=H11223344<CRLF>
WROK<CRLF>
OK><CRLF>

READ HEX(0:4,4) PASSWORD=H11223344<CRLF>
H11223344<CRLF>
OK><CRLF>

READ HEX(0:4,4) PASSWORD=H55667788<CRLF>
PVER<CRLF>
OK><CRLF>

WRITE HEX(0:4,4)=H11223344 PASSWORD=H11223344<CRLF>
WROK><CRLF>
OK><CRLF>

WRITE HEX(0:4,4)=H11223344 PASSWORD=H55667788<CRLF>
PVERR<CRLF>
OK><CRLF>

PROTECT OFF HEX(0:4,4) PASSWORD=H11223344<CRLF>
PVERR<CRLF>
OK><CRLF>

PROTECT ON PERMANENT HEX(04:4,4) PASSWORD=H11223344<CRLF>
```

```
LCKOK<CRLF>
OK><CRLF>

READ HEX (04:4,4)<CRLF>
PVERR<CRLF>
OK><CRLF>

WRITE HEX(0:4,4)=H11223344 PASSWORD=H11223344<CRLF>
PVERR<CRLF>
OK><CRLF>
```

Understanding Error and Success Responses

There are two types of errors:

- Command-level errors, which indicate that the entire command failed.
- Field-level errors, which indicate that only part of the command failed.

The following table describes the reader error and success responses.

Reader Error and Success Responses

Response	Description	Command-Level or Field-Level Error?
ADERR	Response to a WRITE command for an EPCglobal Class 1 Gen 2 tag when you did not write an even-length value to an even-byte addresses.	Field-level
CKERR	Response to a command with an invalid checksum. If you receive this response in a BRI application that communicates with the reader over a TCP connection, this indicates a command transport error (even though checksums are disabled).	Command-level
ERASEERR	Response to an erase tag field command that was not successful.	Field-level
ERASEOK	Response to an erase tag field command that was successful.	Not an error.
ERR	Response to a command that was not successful.	Command-level
MERR	Response to any command that causes an “out of memory” error.	Command-level
NOTAG	Response to a READ or WRITE command when no tags are found and when the NOTAGRPT attribute is enabled.	Not an error
PVERR	Response when memory is locked.	Field-level
PWERR	Response when the WRITE command failed because the tag had low power.	Field-level
RDERR	Response to a read tag field command that was not successful.	Field-level
WRERR	Response to a write tag field command that was not successful.	Field-level
WROK	Response to a write tag field command that was successful.	Not an error

Understanding EVENT Messages

A reader may emit up to seven types of event messages:

- TRIGGER event messages are the result of input state changes from general purpose input devices, see **"TRIGGER" on page 63** for more information.
- TRIGGERACTION event messages are shown when a trigger executes an action macro. Any output generated by the macro will be sent to the BRI client in the form of a trigger action event. Note that if an action generates a response completion message `OK>` then it will also be included in the trigger action events.
- RADIO event messages are generated internally by the reader and provide information to the user about the state of the reader. These events are related to regulatory compliance issues imposed by the country where the reader is being used. When a reader duty cycle time has expired, it will not perform tag operations until the duty cycle time is available again.
- TAG event messages are detected as a result of READ commands. The <TYPE_SPECIFIC_PART> is identical to the response defined by the read command.
- BATTERY event messages are generated internally by the platform hardware and provide information about the state of the battery associated with the platform. These events are generated when the battery status changes or when the Bluetooth or USB connection on the IP30 is established.

  "Charged" means that there is more than 80% battery life remaining, "operational" means that there is between 80% to 20% battery life remaining, and "low" means that there is less than 20% battery life remaining.
- THERMAL event messages are generated internally by the reader and provide information to the user about the state of operating temperature associated with the reader. When the reader is in an over temperature state, it will stop operating on tags. Once the reader has returned to a normal operating temperature, tag operations will resume normal operation.
- RESET event messages are generated internally by the reader and provide information to the user when the reader has been reset, which is typically caused by the RESET command being executed.

Syntax: All event messages use the following format:

```
EVT:<TYPE> <TYPE_SPECIFIC_PART>
The <TYPE_SPECIFIC_PART> depends on the <TYPE>:
EVT:TRIGGER <NAME> <EVENT> <DATA>
EVT:TRIGGERACTION "TRIGGER_NAME" MACRO_OUTPUT
EVT:RADIO DUTY_CYCLE TIMELEFT <TIME>
EVT:TAG <TAG DATA>
EVT:BATTERY LOW|OPERATIONAL|CHARGED
EVT:THERMAL OVERTEMP|NORMAL <TEMPERATURE>
EVT:RESET
EVT: BRISERVICE NOSESSIONS
EVT: ANTENNA FAULT <ANTENNA NUMBER>
EVT: ANTFAULT <ANTENNA NUMBER>
```

Fields: <TYPE> = This field may be TRIGGER, TRIGGERACTION, RADIO, TAG, BATTERY, THERMAL, or RESET.

<NAME> = This field specifies the name associated with the event, like the name of the trigger.

<EVENT> = This field identifies the event that occurred.

<DATA> = This field contains data, like the state of the GP inputs on the reader.

Examples: **Example 1:**

The following is an example of a TRIGGER event message:

```
EVT:TRIGGER ExampleTRIGGER GPIO 15<CRLF>
```

Example 2:

The following is an example of a TRIGGERACTION event message:

```
EVT:TRIGGERACTION "GPIO_1" H6861681<CRLF>
EVT:TRIGGERACTION "GPIO_1" OK><CRLF>
```

Example 3:

The following is an example of a RADIO event message:

```
EVT:RADIO DUTY_CYCLE TIMELEFT xxx<CRLF>
```

The event message above is generated when the reader must turn off the radio in order to comply with country regulations. The response is type RADIO with the name DUTY_CYCLE. The event was TIMELEFT and the *xxx* is a decimal value in seconds representing the remaining that the radio can be turned ON and actively looking for tags.

Example 4:

The following is an example of the TAG event message from "READ TAGID REPORT=EVENT":

```
EVT:TAG H112210101122334411221122<CRLF>
```

Example 5:

The following is an example of the TAG event message from "READ ANT EPCID COUNT TIME":

```
EVT:TAG 3 H1122101011223344112211221 22 34537<CRLF>
```

Example 6:

The event below is generated from the BATTERY event message when low level battery capacity has been detected:

```
EVT:BATTERY LOW<CRLF>
```

Example 7:

The following is an example of the THERMAL event message when an over-temperature condition in the reader has been detected:

```
EVT: THERMAL OVERTEMP <degrees>
```

Example 8:

The following is an example of the ANTENNA FAULT event message that is generated for every READ, WRITE, etc. operation that is executed on a faulty, disconnected, or mismatched antenna:

```
EVT: ANTENNA FAULT 2<CRLF>
```

Example 9:

The following is an example of the ANTENNA FAULT event message that is generated for every READ, WRITE, etc. operation that is executed on a faulty, network reader:

```
EVT: ANTFAULT 2<CRLF>
```

Understanding the Format of BRI Command Responses

This section illustrates the variety of command responses available. All BRI command responses are formatted in the same order that the BRI command is presented to the reader. You can include information from multiple tags in the field, and you can use [LITERAL] parameters to make the responses easier to read.

Command: `READ TAGID, ANTENNA, TIMESTAMP<CRLF>`

Response: `H1234561234561234 1 12345<CRLF>`

`OK><CRLF>`

Command: `READ ANTENNA TAGID<CRLF>`

Response: `2 H1234512345123451<CRLF>`

`OK><CRLF>`

Command: `READ "TAG:"TAGID "TIME="TIMESTAMP<CRLF>`

Response: `TAG:H1234567890ABCDEF TIME=12345<CRLF>`

`OK><CRLF>`

Command: `READ TAGID<CRLF>`

This READ command finds four tags in the field.

Response:
```
HABCDEF1111111111<CRLF>
HABCDEF2222222222<CRLF>
HABCDEF3333333333<CRLF>
HABCDEF4444444444<CRLF>
OK><CRLF>
```

Command:
```
READ STR(18,5)<CRLF>
```

This command reads a string from the tag and finds three tags in the field.

Response:
```
"HELLO"<CRLF>
"CLOSE"<CRLF>
"FIELD"<CRLF>
OK><CRLF>
```

Command:
```
READ "PRICE ", INT(28,2), "QUANTITY ", INT(30,2)<CRLF>
```

This READ command reads two numeric fields from a tag and finds three tags in the field. This command also shows how to make the response more readable using [LITERAL] parameters PRICE and QUANTITY. One tag also returns a read error for the second field.

Response:
```
PRICE 89 QUANTITY 100<CRLF>
PRICE 139 QUANTITY 12<CRLF>
PRICE 39 QUANTITY RDERR<CRLF>
OK><CRLF>
```

Creating and Using BRI Macros

Macros provide a shorthand method for defining and sending complex BRI commands without sending all of the command line parameters each time the command is used. Macros let you create shorter, more convenient commands.

You can send two types of macros:

- A command macro contains a BRI command or sequence of BRI commands to be executed.
- A parameter macro contains a list of parameters to be used when executing a BRI command.

Your macro can contain any BRI commands, command line parameters, and reserved keywords.

Macros are stored in the non-volatile memory of the reader.

The following sections explain how to create, invoke, list, view, and delete macros. If you prefer looking at command descriptions:

- see **"SET" on page 62**. The SET command lets you create, list, and delete macros.
- see **"PRINT" on page 52**. The PRINT command lets you view the contents of a macro.

Creating a Command Macro

You create a command macro, which contains both a command and all its parameters, with the SET command. For example:

```
SET MYREADMACRO="READ TAGID WHERE INT(20,2)=2000"<CRLF>
```

Follow these guidelines when naming a macro:

- The name is an alphanumeric string that can be as short as one character; there is no length limit for the name.
- You cannot use BRI reserved keywords as macro names. For a list of reserved keywords, see **"Reserved Keywords" on page 17**.
- The first character must be a letter (A to Z or a to z).
- The first character cannot be a number (0 to 9).

If the macro contains [LITERAL] parameters, which must be enclosed in double quotes, you must use the \ character to escape the embedded double quotes.

To execute a command macro, see the next section, "Executing a Command Macro."

Executing a Command Macro

You execute a command macro by typing the name of the macro preceded by a $. Suppose a macro MYREADMACRO was defined, you invoke it with this command:

```
$MYREADMACRO
```

The macro MYREADMACRO is expanded by the BRI and replaced with the text string stored in the macro MYREADMACRO.

Creating a Parameter Macro

Suppose these are the parameters for a READ command used several times in your application:

```
READ "TAG ID:" ,HEX(0,8), "NAME:", STRING(18,20) WHERE INT(18)=1
AND INT(22) != 100 AND STRING(30)="ADDRESS"
```

You can store all those parameters in a parameter macro called MYREADPARAMS with this command:

```
SET MYREADPARAMS ="\"TAG ID:\", HEX(0,8),
\"NAME:\",STRING(18,20) WHERE INT(18)=1 AND INT(22) != 100 AND
STRING(30)=\"ADDRESS\""
```

Note how the \ character is used to escape the embedded double quotes for the [LITERAL] parameters.

To execute a READ command using this parameter macro, see the next section, "Executing a Command With a Parameter Macro."

Executing a Command With a Parameter Macro

You execute a command with a parameter macro by typing the command followed by a $ and the macro name.

For example, you can execute the READ command using the parameters stored in MYREADPARAMS with this command:

```
READ $MYREADPARAMS
```

Listing All Macros Stored in Memory

Macros are stored in the non-volatile memory of the reader. To display a list of all the macros currently stored in memory, use the SET command with no parameters:

```
SET<CRLF>
```

The BRI returns a list like this one:

```
MYREADMACRO=READ TAGID WHERE INT(20,2)=2000<CRLF>
MYREADPARAMS=\"TAG ID:\",HEX(0,8), \"NAME:\",STRING(18,20) WHERE
INT(18)=1 AND INT(22) != 100 AND STRING(30)=\"ADDRESS\"<CRLF>
OK><CRLF>
```

Displaying the Contents of a Macro

To display the contents of a macro, use the PRINT command as follows:

```
PRINT $<NAME><CRLF>
```

where <NAME> is the name of the macro.

For example, use this command to view the contents of the MYREADMACRO macro:

```
PRINT $MYREADMACRO
```

The data is returned in a format that can be used in a subsequent BRI command.

Deleting a Macro

To delete a macro from memory, use the SET command with no data after the macro name as follows:

```
SET <NAME>=<CRLF>
```

where <NAME> is the name of the macro.

For example, to delete MYREADMACRO, use this command:

```
SET MYREADMACRO=<CRLF>
```

5

Reader-Specific Platform Specifications

This chapter lists the reader-specific platform specifications that you should be aware of. This chapter contains these sections:

- **Reader-Specific Platform Specifications**
- **ITRFxxx01 Readers**
- **Readers That Contain the IM5 Module**
- **Readers That Contain the IM4 Module**

Reader-Specific Platform Specifications

If a reader deviates from the BRI specification, it is noted in this chapter. The differences can include enhanced or reduced feature sets. This chapter outlines the reader-specific implementations for these readers:

- ITRF*xxx*01 readers
- Readers that contain the IM5 module
- Readers that contain the IM4 module with software version 5.*xx* (or later)

For a complete list of readers that implement all or part of the BRI, see your Intermec representative.

ITRFxxx01 Readers

This section describes how ITRF*xxx*01 readers deviate from the BRI specification. The ITRF*xxx*01 name is used to indicate both the ITRF41501 reader and the ITRF24501 reader.

BRI Disabled by Default

By default, the BRI is disabled on the ITRF*xxx*01 reader. You must use a separate utility to enable the BRI in the reader.

Features Not Implemented

Attribute XON/XOFF is not implemented fully. The reader will stop and start data transmission in response to XON/XOFF commands. When the attribute is enabled, the reader will never send XON/XOFF characters to the host.

Buffer Sizes

Be aware of these buffer size limits on the ITRF*xxx*01 reader:

- Maximum command line buffer size is 255 characters including expanded macros.
- Maximum macro buffer size is 255 characters.
- Maximum macro name size is 32 characters.
- Maximum [LITERAL] lengths are 255 characters.
- Maximum length for HEX and STRING data type definitions are 104 characters.
- Maximum memory storage for commands, fields, literals and macros is 4092 bytes.

Memory Management

The ITRF*xxx*01 reader has limited memory available. The maximum available memory is 4092 bytes. This memory must be shared with macros, commands, literals and BRI response functions. Flexibility has been provided to allow the application to efficiently use the limited memory resources.

Intermec recommends that no more than 3000 bytes of memory be used for macro storage. If you follow this recommendation, the BRI will operate correctly and have sufficient memory resources for all possible command lines. If memory resources become too low in the BRI, the responses may not return all the data specified in a BRI command.

Error Responses

The MERR error is reported when there is not enough memory to store macros or process commands. The ITRF*xxx*01 reader has a total 4092 bytes of available RAM. Since all command processing and macro storage is kept in this limited RAM space, it is possible that a command will fail with this MERR for no apparent reason.

If you receive the MERR error, you should delete some macros to increase the amount of available memory. For help, see **"Listing All Macros Stored in Memory" on page 115** and **"Deleting a Macro" on page 115**.

Antennas

The ITRF*xxx*01 reader has four antennas that can be controlled with the ANTS attribute.

GPIO

The ITRF*xxx*01 reader has four general purpose input ports and four general purpose output ports.

The WRITEGPI command requires you to keep track of the current status of the output lines in the application software. For help, see **"WRITEGPO" on page 73**.

Reader Attributes

In Japan, the following attributes have unique defaults:

- `TTY=ON` is set to `TTY=OFF` for all Japan country codes.
- `ECHO=ON` is set to `ECHO=OFF` for all Japan country codes.
- `IDREPORT=ON` is set to `IDREPORT=OFF` for all Japan country codes.

BRI Commands

The following BRI commands are not implemented in the ITRF*xxx*01 BRI:

- TRIGGER
- TRIGGERREADY
- TRIGGERQUEUE

In this example, the WHERE expression is evaluated exactly as described in **"Using AND/OR Logic in Data Conditions" on page 30**:

```
READ INT(18) WHERE INT(19)=1 OR INT(20)=1 AND INT(21)=3
```

However, if you change the order of the expression, the resulting tags selected are evaluated as if parentheses were used:

```
READ INT(18) WHERE INT(20)=1 AND INT(21)=3 OR INT(19)=1
```

Since an OR follows an AND, the expression is evaluated using parentheses:

```
READ INT(18) WHERE ( INT(20)=1 AND INT(21) ) OR INT(19)=1
```

Implied parentheses are applied any time an OR keyword is followed by an AND keyword.

EVENT Responses

The ITRF*xxx*01 reader does not support the event responses, as described in **"Understanding Error and Success Responses" on page 109**.

Phillips V1.19 TAG Type

The ITRF*xxx*01 reader does not support the Phillips V1.19 tag type, as described in **"TAGTYPE" on page 96**.

Readers That Contain the IM5 Module

RFID readers that contains the IM5 module deviate from the BRI specification as described in this section.

For a complete list of RFID readers that contain the IM5 module, see your Intermec representative.

Buffer Sizes

Be aware of these buffer size limits on readers that contain the IM5 module and architecture older than IF61 v2.00:

- Maximum command line buffer size is 255 characters including expanded macros.
- Maximum macro buffer size is 255 characters.
- Maximum macro name size is 32 characters.
- Maximum [LITERAL] lengths are 255 characters.
- Maximum length for HEX and STRING data type definitions are 104 characters.

Antennas

Readers that contain the IM5 module have four antennas that can be controlled with the ANTS attribute.

GPIO

Readers that contain the IM5 module have four general purpose input ports and four general purpose output ports. The WRITEGPI command requires you to keep track of the current status of the output lines in the application software. For help, see **"WRITEGPO" on page 73**.

Reader Attributes

The default for the attribute INITTRIES is 1. Intermec recommends that you leave the INITRIES value set to 1 for readers that contain the IM5 module. For more information, see **"INITTRIES" on page 92**.

FACDFLT Command

The FACDFLT command restores the reader to its factory default configuration, as described in **"Resetting the Reader to the Factory Default Configuration" on page 3**. However, the reader remains in BRI mode of operation.

Readers That Contain the IM4 Module

RFID readers that contains the IM4 module with software version 5.*xx* (and higher) support the BRI. These readers deviate from the BRI specification as described in this section.

For a complete list of RFID readers that contain the IM4 module, see your Intermec representative.

WRITEGPI Command

The WRITEGPI command sets the state of all the GP output lines available on a reader. For more information, see **"WRITEGPO" on page 73**.

Readers that contain the IM4 module have two GP output lines, which you can set to 0 (OFF) or 1 (ON). The WRITEGPI command can include a value from 0 to 3, as shown in the next table.

Choosing a Value for WRITEGPI

	Which GP Output Lines are Turned ON?	
Value	**1**	**2**
0		
1	ON	
2		ON
3	ON	ON

READGPI Command

The READGPI command returns a value that indicates the current state of all the GP input lines available on a reader. For more information, see **"READGPI" on page 60**.

Readers that contain the IM4 module have two GP input lines, which are numbered from 1 to 2. Each GP input line can be 0 (OFF) or 1 (ON).

Interpreting the Value Returned by READGPI

	Which GP Input Lines are Turned ON?	
Value	**1**	**2**
0		
1	ON	
2		ON
3	ON	ON

The response is formatted as follows for a reader that has two GP input lines, indicating that all input are ON:

```
3<CRLF>
OK><CRLF>
```

TRIGGER Command

The TRIGGER command creates, deletes, and displays trigger events that are based on the state of the GP inputs supported by the reader. For more information, see **"TRIGGER" on page 63**.

For readers that contain the IM4 module, the GPIO mask value can only contain values 0, 1, 2, or 3.

Antennas

Readers that contain the IM4 module have one antenna, and affects these programming elements:

- ANTENNA data type definition
- ANTENNA or ANTS attribute

ANTENNA Data Type Definition

The ANTENNA data type definition indicates which antenna primarily located the tag. ANTENNA is also a read-only value reported during the execution of READ and WRITE commands. For more information, see **"ANTENNA" on page 22**.

For readers that contain the IM4 module, the ANTENNA data type definition always returns a value of 1.

ANTENNAS or ANTS Attribute

The ANTENNA or ANTS attribute sets the antenna sequence to be used during READ and WRITE commands. For more information, see **"ANTENNAS or ANTS" on page 86**.

For readers that contain the IM4 module, you can specify the value 1 once in this attributes command. For example:

ATTRIB ANTS=1<CRLF>

Sensing an Over-Temperature Condition

When the application requests a tag operation (such as identify, read, or write) and the temperature of the IM4 reader is within normal limits, the IM4 returns normal status and data, if found.

If the temperature is out of range (approximately 70 °C or 158 °F at the RF power amplifier), the IM4 returns an over-temperature event to indicate this condition.

The IM4 returns events related to the over-temperature condition in the following format:

`EVT:THERMAL OVERTEMP xxC<CRLF>`

`EVT:THERMAL NORMAL xxC<CRLF>`

where *xx* is the current temperature in Celsius.

While the IM4 is in an over-temperature condition:

- The reader will return a NOTAG response.
- Only the tag-reading functions are inhibited. All other functions are possible.
- The time required for the IM4 to return to a normal temperature depends on the ability of the environment to dissipate the heat.

When the IM4 returns to a normal operating temperature, it returns normal status in response to commands.

Note: The IM4 has a built-in thermal-control self-protection circuit. Because many environmental variations exist, it is impossible to predict exactly when the IM4 may go into thermal-protection mode. The IM4 was designed for up to 1 Watt of continuous power at 100% duty cycle, and care has been taken to rid the IM4 of excess heat, ensuring that users rarely see this scenario.

Index

A

ACCESS and KILL passwords
- ACCESS password, description, **104**
- KILL password, description, **104**
- Memory Bank 0 PASSWORD Memory, **108**
- Memory Bank 2 TID Memory, **106**
- Memory Bank 3 User Memory, **105**, **107**

actions
- EPCC1G2 select logic, **33**

ADERR response, description, **109**
AFI data field, **23**
AND keyword, **31**
AND logic, grouping expressions, **32**
AND/OR logic
- keywords, **31**
- operators, **31**

ANTENNA data field, **22**
ANTENNA data type definition, **122**
ANTENNAS or ANTS attribute, **86**, **123**
ANTTIMEOUT attribute, **87**
ANTTRIES attribute, **87**
archictecture of BRI, **2**
asynchronous event handler, **12**
asynchronous event messages, **11**
ATTRIBUTE command, **36**
attributes
- ANTENNAS or ANTS, **86**
- ANTTIMEOUT, **87**
- ANTTRIES, **87**
- BAUD, **87**
- BROADCASTSYNC, **88**
- BTPWROFF, **88**
- CHANNEL, **88**
- CHKSUM, **88**
- DENSEREADERMODE, **89**
- ECHO, **89**
- EPCC1G2PARAMETERS, **89**
- FIELDSEP, **89**
- FIELDSTRENGTH, **90**
- IDREPORT, **91**
- IDTIMEOUT, **91**
- IDTRIES, **92**
- INITIALQ, **92**
- INITTRIES, **92**
- LBTCHANNEL, **92**
- LBTSCANENABLE, **93**
- LOCKTRIES, **93**
- NOTAGRPT, **93**
- QUERYSEL, **94**
- QUERYTARGET, **94**
- RDTRIES, **94**
- RPTTIMEOUT, **95**
- SCHEDULEOPT, **95**
- SELTRIES, **96**
- SESSION, **96**
- TAGTYPE, **96**
- TIMEOUTMODE, **97**
- TTY, **97**
- UNSELTRIES, **98**
- UTCTIME, **98**
- WRTRIES, **99**
- XONXOFF, **98**

attributes list, reserved keyword list, **20**

B

Basic Reader Interface
- access methods, **2**
- conventions used, **5**
- default configuration, **3**
- general features, **2**
- Hyperterminal, using to connect, **2**
- overview, **2**
- telnet, using to connect, **2**
- usage scenarios, **2**
- version, identifying, **4**

BATTERY event messages, **110**
BAUD attribute, **87**
binary constants, **22**
BIT data field, **23**
BLOCKPERMALOCK command, **38**
BLOCKPERMALOCK READ command, **40**
BRI command list, **17**
BRI command processor thread, **13**
BRI command response types
- command error response, **11**
- normal command response, **11**

BRI command responses, understanding, **112**
BRI command, message type, **11**
BRI commands
- reserved keyword list, **17**

BRI event handler, **14**
BRI Extensions
- EMM tags
  - READ EMM_GETSENSOR, **81**
  - READ EMM_GETUID, **81**
  - WRITE EMM_RESETALARMS, **82**
  - WRITE EMM_SENDSPI, **82**
- Fujitsu tags
  - FJAREAREADLOCK, **76**
  - FJAREAWRITELOCK, **76**
  - FJAREAWRITELOCKWOPWD, **77**
  - FJBURSTERASE, **77**

FJBURSTWRITE, 77
FJCHGAREAGROUPPWD, 78
FJCHGBLOCKGROUPPWD, 78
FJCHGBLOCKLOCK, 79
FJCHGWORDLOCK, 79
FJREADBLOCKLOCK, 79
IMPINJ tags
READ IMPINJQT, 80
WRITE IMPINJQT, 80
NXP tags
NXPALARM, 74
NXPCONFIG, 75
NXPEAS, 75
NXPREADPROTECT, 76
BRI logic interface, description, 16
BRI message layer, programming, 11
BRI asynchronous event handler, 12
BRI command processor, 12
BRI message types, 11
asynchronous event messages, 11
BRI command, 11
BRI Command Response, 11
response handler, 12
BRI transport layer, programming, 12
transport initialization, 13
transport operation, 12
BRI. See Basic Reader Interface
BRISERVICE event messages, 111
BRIVER command, 4, 40
BROADCASTSYNC attribute, 88
BTPWROFF attribute, 88

C

CAPABILITIES command, 41
CHANNEL attribute, 88
CHKSUM attribute, 88
CKERR response, description, 109
command error response, description, 11
command macro
description, 113
example, 114
executing, 114
guidelines, 114
macros, deleting, 115
macros, displaying, 115
stored macros, listing macro, 115
command parameters list, 18
command parameters, reserved keyword list, 18
command processor, 12
command processor thread, 13
command response types
command error response, 11
normal command response, 11
command responses, understanding, 112
command/response structure, BRI logic interface, 16
command-level errors
CKERR, 109
description, 109
ERR, 109
MERR, 109
commands
ATTRIBUTE, 36
BLOCKPERMALOCK, 38
BLOCKPERMALOCK READ, 40
BRIVER, 4, 40
CAPABILITIES, 41
conventions used, 36
DIAGNOSTICS, 44
ERASE, 45
FACDFLT, 45, 121
FACDLFT commands, 3
FLEXQUERY, 47
HELP, 47
HWCC, 48
HWID, 48
HWPROD, 48
HWREGION, 48
HWVER, 49
KILLTAG, 50
LOCK, 50
PING, 51
PLATDFLT, 51
PRESET, 52
PRINT, 52
PROTECT, 52
READ, 54
READGPIO, 60, 122
REPEAT, 60
RESET, 61
SET, 62
SWVER, 63
TRIGGER, 63, 122
TRIGGERQUEUE, 66
TRIGGERREADY, 67
TRIGGERWAIT, 68
VERSION, 68
WRITE, 69
WRITEGPIO, 73, 121
constants
binary constants, 22
hex constants, 22
integer constants, 22
octal constants, 22
string constants, 22
conventions used for BRI commands, 36
conventions used in manual, 5

COUNT data field, 23

D

data conditions
AND/OR logic
grouping expressions, 32
operators, 31
using, 30
defined, 28
EPCC1G2 select logic
actions, 33
targets, 33
example, 31
limitations, 29
multi-protocol condition usage, 34
native tag selector logic
operators, 29
using, 29
data field definitions
AFI data field, 23
ANTENNA data field, 22
BIT data field, 23
COUNT data field, 23
description, 22
EPCID data field, 23
HEX data field, 24
INT data field, 25
ISOUII data field, 26
PC data field, 27
RSSI data field, 27
STRING data field, 27
tag types supported, 30
TAGTYPE data field, 28
TIME data field, 28
default configuration, values, 3
defaults used in manual, 5
DENSEREADERMODE attribute, 89
DIAGNOSTICS command, 44
documentation, third party
documents, xii

E

ECHO attribute, 89
EMM tags
READ EMM_GETSENSOR, 81
READ EMM_GETUID, 81
WRITE
EMM_RESETALARMS, 82
WRITE EMM_SENDSPI, 82
EPC Global Gen 2 tag, ACCESS and
KILL passwords, 104
EPCC1G2 select logic, described, 33
EPCC1G2PARAMETERS
attribute, 89
EPCglobal Class 1 Gen 2 tag
memory bank parameter
description, 25
EPCID data field, 23
ERASE command, 45
ERASEERR response,
description, 109
ERASEOK response, description, 109
ERR response, description, 109
error and success responses, 109
ADERR, 109
CKERR, 109
ERASEERR, 109
ERASEOK, 109
ERR, 109
MER, 109
NOTAG, 109
PVERR, 109
PWERR, 109
RDERR, 109
WRERR, 109
WROK, 109
Ethernet, accessing with, 2
event handler thread, 14
event messages, 110
BATTERY, 110
BRISERVICE, 111
examples, 111
formats, 111
NOSESSIONS, 111
RADIO, 110
RESET, 110
TAG, 110
THERMAL, 110
TRIGGER, 110
TRIGGERACTION, 110
understanding, 110

F

FACDFLT command, 3, 45, 121
factory default configuration
resetting, 3
factory default configuration,
values, 3
field-level errors
ADERR, 109
description, 109
ERASEER, 109
PVERR, 109
PWERR, 109
RDERR, 109
WRERR, 109
FIELDSEP attribute, 89
FIELDSTRENGTH attribute, 90
FJAREAREADLOCK, 76
FJAREAWRITELOCK, 76
FJAREAWRITELOCKWOPWD, 77
FJBURSTERASE, 77
FJBURSTWRITE, 77

FJCHGAREAGROUPPWD, 78
FJCHGBLOCKGROUPPWD, 78
FJCHGBLOCKLOCK, 79
FJCHGWORDLOCK, 79
FJREADBLOCKLOCK, 79
FLEXQUERY command, 47
Fujitsu tags
  FJAREAREADLOCK, 76
  FJAREAWRITELOCK, 76
  FJAREAWRITELOCKWOPWD, 77
  FJBURSTERASE, 77
  FJBURSTWRITE, 77
  FJCHGAREAGROUPPWD, 78
  FJCHGBLOCKGROUPPWD, 78
  FJCHGBLOCKLOCK, 79
  FJCHGWORDLOCK, 79
  FJREADBLOCKLOCK, 79

G

general features, Basic Reader Interface, 2
Generic, 82
grouping expressions, AND/OR logic, 32

H

HELP command, 47
hex constants, 22
HEX data field, 24
HWCC command, 48
HWID command, 48
HWPROD command, 48
HWREGION command, 48
HWVER command, 49
Hyperterminal, using to connect, 2

I

IDREPORT attribute, 91
IDTIMEOUT attribute, 91
IDTRIES attribute, 92
IM4 module
  ANTENNA data type definition, 122
  antennas, 122
  ANTENNAS or ANTS attribute, 123
  description, 121
  over-temperature condition, 123
  READGPIO command, 122
  TRIGGER command, 122
  WRITEGPIO command, 121
IM5 module
  antennas, 121
  buffer sizes, 120
  description, 120
  FACDFLT command, 121
  GPIO, 121
  reader attributes, 121
IMPINJ tags
  READ IMPINJQT, 80
  WRITEIMPINJQT, 80
INITIALQ attribute, 92
INITTRIES attribute, 92
input message thread, 13
INT data field, 25
integer constants, 22
ISOUII data field, 26
ITRFxx01 Readers
  antennas, 119
  BRI commands not implemented, 120
  buffer sizes, 118
  default settings, 118
  description, 118
  error responses, 119
  EVENT responses, 120
  GPIO, 119
  implemented features, 118
  memory management, 119
  reader attributes, 119
  tag types supported, 120

K

keywords
  AND, using, 31
  OR, using, 31
KILL and ACCESS passwords, 107
  ACCESS password, description, 104
  KILL password, description, 104
  Memory Bank 0 PASSWORD Memory, 108
  Memory Bank 2 TID Memory, 106
  Memory Bank 3 User Memory, 105
KILLTAG command, 50

L

LBTCHANNEL attribute, 92
LBTSCANENABLE attribute, 93
LITERAL parameter
  example, 103
  understanding, 103
LOCK command, 50
LOCKTRIES attribute, 93
logic interface, description, 16

M

macro
  command macro
    creating, 114
    description, 113
    example, 114
    executing, 114
    guidelines, 114
    macros, deleting, 115

macros, displaying, 115
stored macros, listing, 115
parameter macro
creating, 114
description, 113
example, 114
executing, 115
guidelines, 114
macros, deleting, 115
macros, displaying, 115
stored macros, listing, 115
manuals, third party documents, xii
Memory Bank 0 PASSWORD Memory, 108
Memory Bank 2 TID Memory, 106
Memory Bank 3 User Memory, 105
memory_bank parameter
description, 25
values, 25
MER response, description, 109
message checksums
enabling, 16
examples, 16
using, 16
message layer
programming, 11
BRI asynchronous event handler, 12
BRI command processor, 12
response handler, 12
message layer, programming
BRI message types, 11
multi-protocol condition usage, 34
multi-threaded implementation
BRI command processor thread, 13
BRI event handler, 14
input message thread, 13

N

native tag selector logic
using, 29
normal command response, description, 11
NOSESSIONS event messages, 111
NOTAG response, description, 109
NOTAGRPT attribute, 93
NXP tags
NXPALARM, 74
NXPCONFIG, 75
NXPEAS, 75
NXPREADPROTECT, 76
NXPALARM, 74
NXPCONFIG, 75
NXPEAS, 75
NXPREADPROTECT, 76

O

octal constants, 22
operators
AND/OR logic, 31
native tag selector logic, 29
OR keyword, 31
OR logic, grouping expressions, 32
over-temperature condition, 123
overview, Basic Reader Interface, 2

P

parameter macro
description, 113
example, 114
executing, 115
guidelines, 114
macros, deleting, 115
macros, displaying, 115
stored macros, listing, 115
PC data field, 27
PING command, 51
PLATDFLT command, 51
PRESET command, 52
PRINT command, 52
PROTECT command, 52
PVERR response, description, 109
PWERR response, description, 109

Q

QUERYSEL attribute, 94
QUERYTARGET attribute, 94

R

RADIO event messages, 110
RDERR response, description, 109
RDTRIES attribute, 94
READ command, 54
READ EMM_GETSENSOR, 81
READ EMM_GETUID, 81
READ IMPINJQT, 80
reader attributes
changing, 36
list, 20
reading, 38
reader, resetting, 3
READGPIO command, 60, 122
REPEAT command, 60
reserved keywords
BRI command list, 17
command parameters list, 18
error and success response list, 21
reader attributes list, 20
RESET command, 61
RESET event messages, 110
resetting, default configuration, 3
response handler, 12
response/command structure, 16
RFID applications, programming, 8

BRI asynchronous event handler, 12
BRI command processor, 12
BRI command processor, programming, 12
BRI message layer, programming, 11
response handler, 12
RFID configuration, managing, 8
RFID resource kit, using, 9
software structure, 9
RFID configuration
integrating, 8
managing, 8
SmartSystems Foundation, using, 8
RFID Resource Kit
downloading, 9
using, 9
RPTTIMEOUT attribute, 95
RSSI data field, 27

S

SCHEDULEOPT attribute, 95
SELTRIES attribute, 96
serial interface, accessing with, 2
SESSION attribute, 96
SET command, 62
SmartSystems Foundation, 8
software structure
description, 9
illustrated, 10
specifications, list of, xii
string constants, 22
STRING data field, 27
STRING fields
description, 103
example, 103
reading and writing, 103
success and error responses, 109
ADDERR, 109
CKERR, 109
ERASEERR, 109
ERASEOK, 109
ERR, 109
MER, 109
NOTAG, 109
PVERR, 109
PWERR, 109
RDERR, 109
WRERR, 109
WROK, 109
success response list, 21
SWVER command, 63

T

TAG event messages, 110
tag types, data fields supported, 30
TAGTYPE attribute, 96
TAGTYPE data field, 28
targets, EPCC1G2 select logic, 33
TCP connection
choosing TCP port, 3
using, 3
TCP interface, accessing with, 2
telnet, using to connect, 2
THERMAL event messages, 110
third-party documents, list of, xii
TIME data field, 28
TIMEOUTMODE attribute, 97
transport initialization, 13
transport layer, programming, 12
transport operation, 12
TRIGGER command, 63, 122
TRIGGER event messages, 110
TRIGGERACTION messages, 110
TRIGGERQUEUE command, 66
TRIGGERREADY command, 67
TRIGGERWAIT command, 68
TTY attribute, 97

U

UNSELTRIES attribute, 98
usage scenarios, Basic Reader Interface, 2
UTCTIME attribute, 98

V

VERSION command, 68

W

Wi-Fi, accessing with, 2
WRERR response, description, 109
WRITE command, 69
WRITE EMM_RESETALARMS, 82
WRITE EMM_SENDSPI, 82
WRITE IMPINJQT, 80
WRITEGPIO command, 73, 121
WROK response, description, 109
WRTRIES attribute, 99

X

XONXOFF attribute, 98

Intermec

Worldwide Headquarters
6001 36th Avenue West
Everett, Washington 98203
U.S.A.

tel 425.348.2600

fax 425.355.9551

www.intermec.com

Basic Reader Interface Programmer Reference Manual

P/N 937-000-012