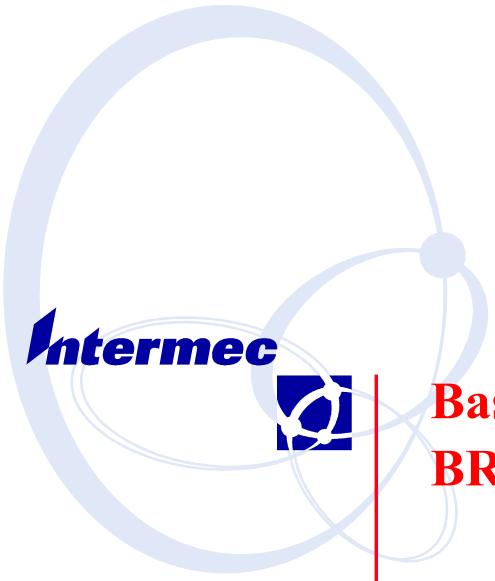

Basic Reader Interface
BRI User Manual

Intellitag Developer's Kit

Table of Contents

1.0 Requirements

1. Personal Computer with
    a. Serial communications port
    b. Terminal Emulation Program (i.e. HyperTerminal).

2. Dumb Terminal with
    a. Serial communications port capable of 115200 bits per second

3. Pennsylvania Reader with version 3.15 firmware or higher
    a. Connect serial interface to Com Port 1 on computer using an appropriate serial cable.
    b. Configure serial baud rate for 115200 bits per second 8,N,1 if not already set.

2.0 Operation

Turn on power to the reader. If this is the first time the reader has been used, run the BRI Setup utility included with the Intellitag™ IDK. This will enable the BRI interface in the reader. Upon power up of a reader with an active BRI interface the reader responds with an initial version message and an OK> prompt. The following commands outlined in this document may now be used.

3.0 BRI Installation Instructions

Before using the BRI for the first time, you must install new firmware. Once this is complete, you can toggle between using the BRI and the API with out having to change firmware.

1. Start the Itrfldr.exe application. This will allow you to download new firmware to the reader.
2. The first time you run this software, you will need to auto detect the reader to verify the baud rate. Select the *Communications Setup* button. A window will open. Select the *Search* button. Once the auto detect process is complete, you are ready to down load the firmware.

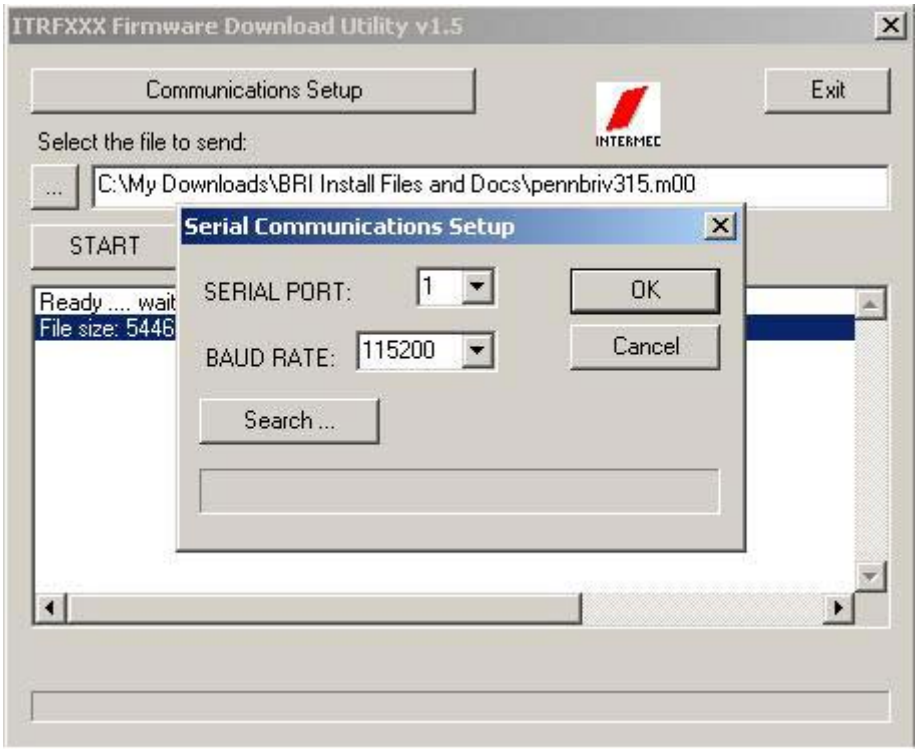

Figure 1: Setup Communications

3. To select the firmware, click the "…" button located below the "Select the file to send" label. Select the firmware named: pennbriv315.m00.
4. Click the Start button do begin downloading firmware. Make sure you do not cycle power or disconnect any cables during the download process. See figure 2 to see what the screen should look like if the download is in process.

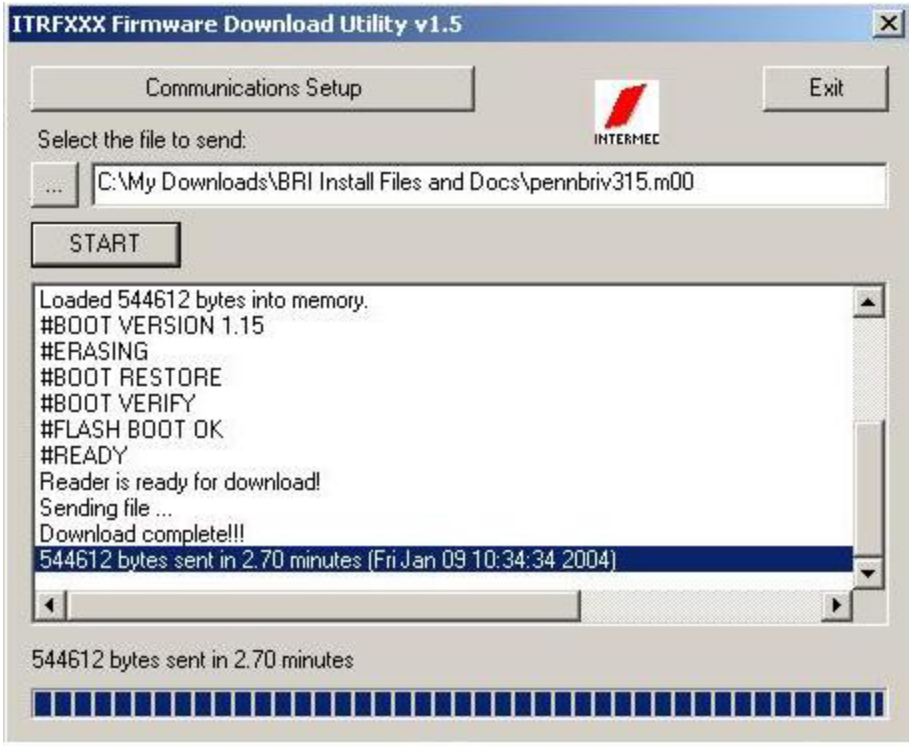

Figure 2: Firmware Download in Process

5. When the download is complete the screen should look like figure 3.

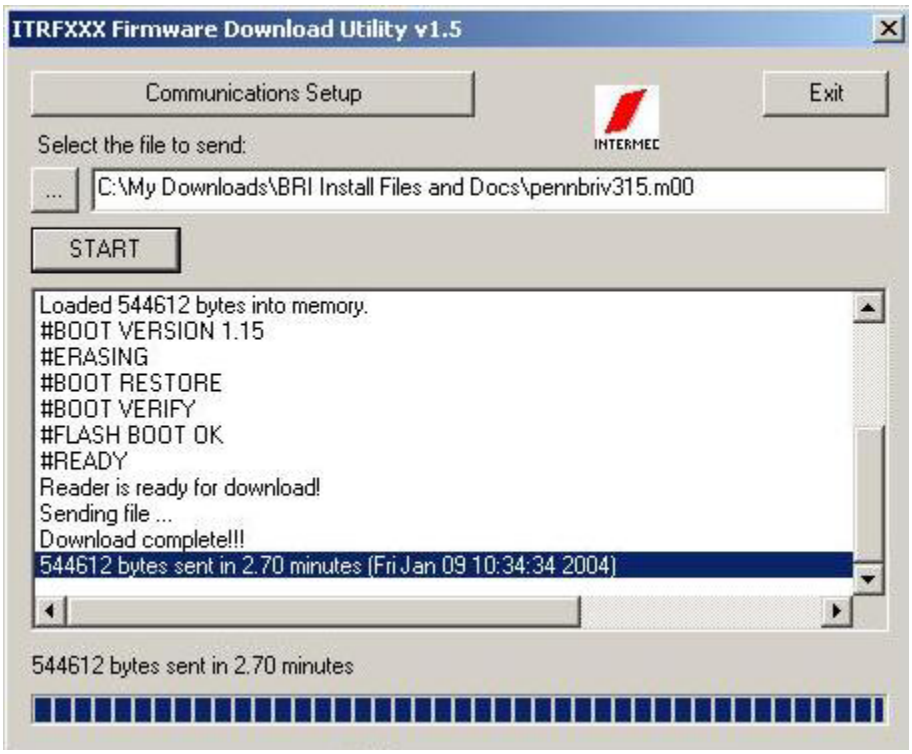

Figure 3: Download Complete

6. Close the download application, Itrfldr.exe.
7. Double click on the file “brisetup.exe”. This will install an application which you will use to when ever you want to activate the BRI in the reader.
8. After install the application, *SetBRI Utility*, you should find a shortcut to it on your Windows START button.

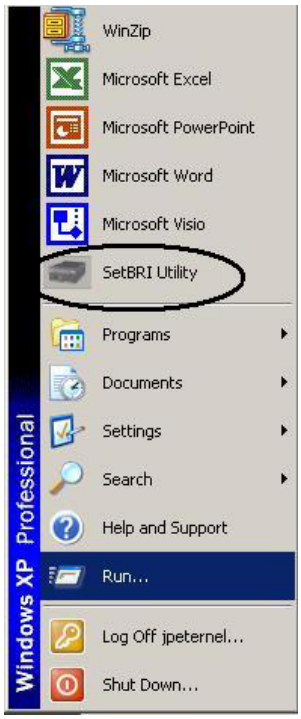

Figure 4: Shortcut to BRI Utility

9. Select the shortcut to activate the utility program.
10. Select the com port number and baud rate for your reader (default is 115200).

Figure 5: BRI Utility

11. Click the *Set BRI* button to activate the BRI.
12. The BRI should now be active in the reader. A green check mark will appear on the application screen if the set BRI succeeds.

Figure 6: Set BRI Success

13. The final step is to CYCLE POWER on the reader. You can also issue the RESET command. You are now ready to use the BRI.
14. To deactivate the BRI, you must set the attribute attribBRI = 0. To reactivate the BRI, you must run the SetBRI Utility.

4.0 Commands

Following is the list of commands provided by the Basic Reader Interface (BRI). The command set is case insensitive. The command "READ" is interpreted by the program the same as "Read".

The "READ" and "WRITE" commands are used for identifying, reading, and writing tags. These two commands are the only two active commands required to harness the entire capabilities of the Intellitag™ feature set. It is recommended that the reader peruse the sections on read, write, and select arguments first; before working with the more advanced capabilities of these commands.

4.1 READ

The read command is the most fundamental command of the BRI. It allows the user to identify, filter, and read data from tags with a minimal command interface. The structure of the command format is illustrated below.

READ { *read field* { , *read field* } } { WHERE *select field* { , *select field* } }

As seen in the above format, there are two optional arguments to the read command.

In its simplest form, since all arguments are optional, the READ command is sufficient by itself.

Example:

```
OK> Read
H1111111111111111
OK>
```

When the READ command is entered without any arguments, the reader will return the Tag IDs of all tags in the RF field of view of the reader.

The first optional argument indicates one or more read fields, separated by commas. The first argument is separated from the command by one or more spaces. When read fields are specified, the command will return the fields specified after each ID that is reported.

Example:

```
OK>Read Hex(0)
H1111111111111111 H1111111111111111
OK>
```

The "Hex(0)" argument placed in the command causes the reader to return the hexadecimal string data in the tag memory beginning at decimal address location 0

following the Tag ID. In the example above, the data read happens to be the Tag ID. The reader will default with a read length of 8 bytes if it is not specified.

If the user wishes to specify the read length, this may be accomplished by including the length within the parentheses following the address location with a comma as shown below.

OK> Read Hex(0,4)
H1111111111111111 H11111111
OK>

Notice that the read data is reduced from 8 bytes (16 hexadecimal characters), to 4 bytes (8 hexadecimal characters).

The second optional argument indicates one or more select fields, also separated by commas. If select fields are specified, the command returns only those tags that match the select fields.

Example:

OK> Read Where TagID = H11
H1111111111111111
OK>

If other tags are in the field with the first byte equal to 0x11, they will also be returned.

OK> Read Where TagID = H11
H1111111111111111
H1122222222222222
OK>

Both read field and select field arguments may be included in a single statement by simply separating the arguments with one or more spaces as shown below.

Example:

OK> Read Hex(0) Where TagID = H11
H1111111111111111
OK>

If multiple arguments of either type are included, it is important to remember to separate the arguments by a comma.

If a wildcard is to be used for the select fields for read or write hexadecimal commands, it is possible to include the character “?” in the compare field as shown below.

Example:

OK> Read Hex(0) Where TagID = H??11
H1111111111111111
H2211222222222222
OK>

More detail on the wildcard function is provided under the hexadecimal select field type description later in this document.

4.2 WRITE

The write command is the only other active command of the BRI. It allows the user to identify, filter and write data to tags with minimal command interface. The structure of the command format is illustrated below. There are two optional arguments for the write command.

WRITE { *write field* { , *write field* } } { WHERE *select field* { , *select field* } }

Using the WRITE command without any arguments, the WRITE command may be used to provide a response with all the tag IDs of all tags in the field of view of the reader as does the READ command. As there was no write data specified in the command, no data will be written to the tags.

Example:

OK> Write
H1111111111111111
OK>

The first optional argument indicates one or more write fields, separated by commas. The first argument is separated from the command by one or more spaces. When write fields are specified, the command will return the fields specified after each ID that is reported.

Example:

OK> Write Hex(20) = HDEADBEEF
H1111111111111111 WROK
OK> Read Hex(20,4)
H1111111111111111 HDEADBEEF
OK>

If there is an error, the reader will return “WRERR” following the Tag ID instead of “WROK”.

The “Hex(20) = HDEADBEEF” argument placed in the command causes the reader to write the hexadecimal string data into tag memory beginning at decimal address location 20. In the example above, the data write was successful.

A write length can be included and is required when writing integers. This is detailed in the write field description toward the end of this document.

Similar to the read command, the second optional argument indicates one or more select fields, also separated by commas. If select fields are specified, the command returns only those tags the match the select fields.

Example:

OK> Write Hex(20) = HDEADBEEF Where TagID = H11
H1111111111111111 WROK
H1122222222222222 WROK
OK>

If the WRITE command is executed without any write fields, but with select fields, the IDs of all tags matching the select fields will be returned, but no writes are performed.

Write fields are required for any data to be written to any tags. If write fields are specified, but no select fields are specified, the WRITE command performs the write to all tags within its RF field of view. If write fields and select fields are specified, data is written to only those tags matching the selected fields.

4.3 REPEAT

This command will re-execute the last READ or WRITE command specified. There is no argument required.

4.4 SET

SET *identifier = string*

The SET command is used to create a macro shortcut. This allows the user to store and reference a complete or partial command sequence using a simple macro identifier name. Prefix a macro with a ‘$’ in order to use a stored macro within a command.

Note that the string assigned to the macro must be enclosed within quotes.

Example:

OK> SET mycmd = “READ WHERE TAGID = H11”
OK>

This macro internally expands to "READ WHERE TAGID = H11" in the reader. To use this macro, simply enter the name you assigned the macro preceded with a "$" as shown below:

```
OK>$mycmd
H1111111111111111
OK>
```

To delete a macro from memory, enter the SET command without including any assigned value.

Example:

```
OK>SET mycmd =
OK>
```

The macro has now been erased.

The BRI does not support nested macros, or macros within macros.

4.5 PRINT

PRINT *text…*

The print command simply echoes whatever is typed beyond the PRINT keyword to the console. It will expand macros, so it is handy for checking the output of commands that use macros. The example below concatenates the two macros, "MyRds" and "MySels".

Example:

```
OK>SET MyRds = "hex(20)"
OK>SET MySels = "WHERE int(11,2) = 1"
OK> PRINT READ $MyRds $MySels
READ hex(20) WHERE int(11,2) = 1
OK>
```

4.6 ATTRIBUTE

ATTRIBUTE *attribute = value[s]*

The ATTRIBUTE command allows the user to set parameters specific to the reader platform being used by the BRI. The following attributes are common to all readers supporting the BRI interface.

4.6.1 IDTRIES attribute

IDTRIES = *n*

This attribute is used to set the maximum number of identify attempts the reader will use. Its argument is a numeric value in the range of 1 – 255.

Example:

OK>ATTR IDTRIES = 1
OK>

Note the ATTRIBUTE command (short form ATTR) precedes the attribute to be set.

3.6.2 RDTRIES attribute

RDTRIES = *n*

This attribute is used to set the maximum number of read attempts the reader will use. Its argument is a numeric value in the range of 1 – 255.

Example:

OK>ATTR RDTRIES = 1
OK>

4.6.3 WRTRIES attribute

WRTRIES = *n*

This attribute is used to set the maximum number of write attempts the reader will use. Its argument is a numeric value in the range of 1 – 255.

Example:

OK>ATTR WRTRIES = 1
OK>

4.6.4 ANTTRIES attribute

ANTTRIES = *n*

This attribute is used to set the maximum number of identify attempts per antenna the reader will use. Its argument is a numeric value in the range of 1 – 255.

Example:

OK>ATTR ANTTRIES = 1
OK>

4.6.5 ANTENNAS attribute

ANTENNAS = *n, n, n, n, n, n, n, n*

This attribute is used the set the antennas scan array within the reader. Eight numeric arguments separated by commas are required. The reader scans the list from left to right. The Pennsylvania reader recognizes antenna numbers 1 to 4. Use the value zero in any position within the array which you desire to skip.

Example:

Create an array with antennas 2 and 3 only.

OK>ATTR ANTENNAS = 2,3,0,0,0,0,0,0
OK>

4.6.6 FIELDSEP attribute

FIELDSEP = *string*

The FIELDSEP attribute is used to set the field separator the reader will use. By default, this sequence is a space character. The field separator is displayed in-between the read fields of a tag transaction. It is not recommended to change this unless you have a specific need.

Example:

Set a field separator consisting of a dash

OK>ATTR FIELDSEP = "-"
OK>

Set the field separator to the default setting

OK>ATTR FIELDSEP = " "
OK>

4.6.7 NOTAG attribute

NOTAG = *string*

The NOTAG attribute is used to display a message when no tags are present before the reader and when the reader is in automation mode. The default message is “No Tag”.

Example:
Set the NOTAG attribute message to “No Tags Present”

OK>ATTR NOTAG = “No Tags Present”
OK>

Disable sending a NOTAG attribute message

OK>ATTR NOTAG = “”
OK>

4.7 VERSION

OK>VERSION

This command has no parameters and commands the reader to return the current software versions installed in the reader.

4.8 FACDFLT

OK>FACDFLT
OK>

This command has no parameters and commands the reader to reset all parameters back to the FACTORY DEFAULT settings. This command will set the attributes to the following values after it is executed. Note that this command will disable the BRI mode of operation. It will be necessary to run the SetBRI Utility to return to the BRI mode of operation.

The default attribute settings are shown below:

TTY=ON
ECHO=ON
BAUD=115200
FIELDSEP=” “
IDTRIES=3
RDTRIES=3

WRTRIES=3
LOCKTRIES=3
INITTRIES=3
ANTTRIES=3
BRI=OFF
ANTS=1,0,0,0,0,0,0,0

5.0 Data Types

These are the three data types recognized by the BRI. These are *integers, hexadecimal strings*, and *strings*. These types are used in read, write, select fields, and are useful for the majority of data needs.

There are also keywords that are used to indicate data type, but here we are illustrating how these 'types' appear. Integer types are expressed in decimal, hexadecimal, octal, and character constant.

Note that octal numbers always start with 0, whereas decimal numbers start with 1-9.

5.1 Integers

An integer can be specified as

A decimal number	26
A hexadecimal number	0x1A
An octal number	032

The forth method, rarely used is the character constant.

Character constants can represent any printable ASCII character as well as any 8 bit value, printable or not. Character constants can be specified using common ASCII escape sequences, much as is done in the 'C' programming language. The backslash character is the special character used for this purpose. If you wish to specify a back slash character, type it twice as in the following examples.

'\n'	Represent the 'newline' character	10	(0x0A)
'\t'	Represents the 'tab' character	9	(0x09)
'\v'	Represent the 'vertical tab' character	11	(0x0B)
'\b'	Represents the 'backspace' character	8	(0x08)
'\r'	Represent the 'carriage return' character	13	(0x0D)
'\f'	Represents the 'formfeed' character	12	(0x0C)
'\a'	Represents the 'BEL' character	7	(0x07)
'\\'	Represents the 'backslash character	92	(0x5C)
'\''	Represents the 'single quote' character	39	(0x27)
'\"'	Represent the 'double quote' character	34	(0x22)

'\0*ddd*'	Represent any character as an octal value
'\0x*dd'*	Represent any character as an hexadecimal value

Integers are used wherever a numeric value is required within a given command or argument.

5.2 Strings

Strings are simply a sequence of characters enclosed in double quotes.

Example:

“This is a String”

Strings can contain non-printable and other character in the same fashion as character constants. See character constants for escape sequences.

“This is a string with an embedded carriage return\r”;

Strings are used as arguments in some commands.

5.3 Hexadecimal Strings

Hexadecimal strings are a special string type composed of hexadecimal digits. These should not be confused with hexadecimal integers that express numbers.

The hex string is formed by at least two hexadecimal digits preceeded with an H character to denote it as a hexadecimal string. A hexadecimal string can be of any length, but most command limit hex string arguments to 8 bytes (16 hexadecimal characters).

Example:

H11223344

The above example specifies a hexadecimal string representing 4 consecutive byte values of 0x11, 0x22, 0x33, and 0x44.

5.3.1 Special Case – Wildcards

Hex strings can contain ‘??’ question mark pairs to indicate ‘any value’. This is equivalent to using a wildcard value. Question marks are only valid when using a hex string as an argument to a select field. Question marks are not valid within read or write fields.

Example:

H55??2233

The above example represents a hex string of 4 bytes that matches 0x55, (anything), 0x22, and 0x33. This is only valid in select field arguments.

6.0 Read and Write Command Arguments

6.1 Read Field Arguments

Read Arguments are used to specify the data to be returned from tags and how it is to be presented. This is where the keywords INT, STRING, and HEX begin to be understood. Multiple read arguments can be specified by separating them with commas.

6.1.1 INT Read Field

INT(*n*,*n*)

An integer argument is specified using the INT keyword followed by two numbers enclosed in brackets. The first number indicates the address offset in the tag where the integer number to be retrieved is. The second number indicates the size of the number in bytes – valid sizes are 1 to 4, allowing integers of 8 to 32 bits. The comma and size are optional, if left out, the integer defaults to a one byte size.

Example:

OK> READ int(20,2)
H1122334455667788 65123
OK>

This example returns a 16 bit (2 byte) number from location 20. In this example, the value returned is 65123.

OK>READ int(134)
H1122334455667788 109
OK>

This example did not specify a byte size, so a size of 1 was used by default. The value returned in this example is 109.

6.1.2 HEXADECIMAL STRING Read Field

HEX(*n*,*n*)

A hex string read field argument is the HEX keyword followed by two numbers separated by a comma and enclosed in brackets. The number indicates the address offset within the tag where the hexadecimal string to be retrieved is located. The second number indicates the length of the hexadecimal string to be retrieved (in bytes). Valid lengths are from 1 to

the Maximum tag data. The comma and length are optional, if left out, the size defaults to 8 bytes.

Example:

```
OK>READ HEX(202,2)
H1122334455667788 HBEEF
OK>

OK>READ HEX(200)
H1122334455667788 HDEADBEEFDEADBEEF
OK>
```

6.1.3 STRING Read Field

STRING(*n*,*n*)

A string read field argument is the STRING keyword followed by two numbers separated by a comma and enclosed in brackets. The second number indicates the length of the string to be retrieved (in bytes). Valid lengths are from 1 to the Maximum tag data. The comma and length are optional, if left out, the size defaults to 8 bytes.

Example:

```
OK> READ STRING(50,5)
H1122334455667788 Hello
OK>

OK>READ STR(50)
H1122334455667788 HelloBye
OK>
```

Note: You should only specify string read fields arguments where you understand the data at the specified address is to be presented to be in a printable form. The BRI program will send all characters within the string. If there are control characters within the string retrieved, the console may scroll or tab or display special characters.

6.1.4 Multiple Read Arguments

Multiple read fields can be specified for a read command.

Example:

```
OK>READ int(10,3), HEX(200), STR(80,7)
```

H1122334455667788 16777215 HFEEDBEEFFEEDBEEF TESTSTR
OK>

6.2 Write Field Arguments

Write field arguments are used to specify data to be written to a tag. The keywords INT, STRING, and HEX are again used. The reader responds with the Ids of the tags written followed by a WROK or WRERR for each field written. WROK indicates a successful write. WRERR indicates a failed write.

6.2.1 Integer Write Field Argument

INT(*n*,*n*) = *n*

The arguments to an integer write field are address offset, size and the value to be written. All arguments are required.

Example:

OK>WRITE INT(25,1) = 44;
H1122334455667788 WROK
OK>READ int(25)
H1122334455667788 44
OK>

6.2.2 Hexadecimal String Write Field Argument

HEX(*n*) = *hex string*

The arguments to the hexadecimal string write field are the address offset and the value to be written. A length is not required as it is implicit in the length of the given value to be written. The hexadecimal length may be any size from 1 to the Maximum tag data bytes.

Example:

OK>WRITE HEX(20) = H8877
H1122334455667788 WROK
OK>READ HEX(20,2)
H1122334455667788 H8877
OK>

This example writes a 2 byte hexadecimal string at location 20.

6.2.3 String Write Field Argument

The arguments to the string write field are the address offset and the value to be written. A length is not required as it is implicit in the length of the given string to be written.

Example:

OK>WRITE STR(20) = “Hello”
H1122334455667788 WROK
OK>READ STR(20,5)
H1122334455667788 Hello
OK>

6.3 Select Field Arguments

Select field arguments are used to narrow the set of tags, or choose specific tags when using the READ or WRITE commands. Select fields also use INT, HEX and STRING arguments, along with the logical operators supported by Intellitag™ RFID tags. Selecting and Unselecting are supported. Unselect will be covered later in this section. Note that optional select field arguments to the READ and WRITE commands always start with the keyword WHERE.

6.3.1 Operators

These are the logical operators used for select field comparisons.

=	equals
!=	not equals
<	less than
>	greater than

6.3.2 Integer Select Field

INT(*n*,*n*) *operator n*

The integer arguments are again the address offset, the size in bytes, the logical comparison operator and the integer to compare to.

Example:

```
OK>READ WHERE INT(20,2) = 65535
H1122334455667788
OK>
```

The sample example could be performed using any of the remaining comparison operators such as !=, <, or >, with different results.

6.3.3 Hexadecimal String Select Field

HEX(*n*) *operator hexstr*

The hexadecimal string select argument is arguably the most powerful select argument of all. This argument can accept hexadecimal strings with 'wildcard' values. The arguments to the hexadecimal string write field are the address offset, logical comparison operator, and the hexadecimal string to be compared to.

Example:

```
OK>READ WHERE HEX(20) = HDEAD
H1122334455667788
OK>

OK>READ HEX(20,4) WHERE HEX(20) = H??ADBEEF
H1122334455667788 HDEADBEEF
OK>
```

Note that the hexadecimal string 'H??', a one byte wildcard, would match any tag and would therefore return all tags.

TAGID is a pseudonym/shortcut for HEX(0), which is a tag's ID area.

Example:

```
OK>READ WHERE TAGID = H??
H1122334455667788
OK>
```

In any of the above examples, any of the available logical comparison operators could have been used, with differing results.

6.3.4 String Select Field

STR(n) *operator string*

The string select arguments can be used to select tags using character strings. The arguments to the string select field include the address offset, the logical comparison operator to be used, and the string to be compared.

Example:

OK> READ WHERE STR(20) = “Hello”
H1122334455667788
OK>

Any of the available comparison operators may be used.

6.3.5 Multiple Select Fields

Multiple select fields can be specified by separating them with commas.

Example:

OK>READ WHERE INT(10,2) = 4004, hex(20) = HDEADBEEF

This example will return all tags that match ***either the first select field or the second select field***.

6.3.6 Advanced Select Operation

This section on advanced operations describes how to perform set subtraction using unselect in select fields.

Performing unselects allows you to subtract tags from the resulting tags of a preceding select field. This powerful mechanism allows you to obtain a specific resulting tag set, not just another additional result set. The operator used to unselect tags in a select field is ‘!’ or the keyword NOT. These must be included at the beginning of the select field.

Assume we have 4 tags with ID of:

H5511111111111111
H6622222222222222
H8844444444444444
H7733333333333333

Example:

```
OK>READ WHERE TAGID = H??, !TAGID = H77
H5511111111111111
H6622222222222222
H8844444444444444
OK>
```

In this example, all tags were select and tag H773333333333333 was removed from the result set.

By using select and unselect fields, along with the various comparison operators and various data types there is no limit to select and choose the exact tags needed for a given application.

6.3.7 Putting it all Together

READ and WRITE are designed to work with select fields. The result is that you READ or WRITE to the set of carefully specified tags. There are 3 more READ and WRITE command line parameters that can be used to provide formatted output and also retrieve additional information that occurred during the READ or WRITE command.

6.3.7.1 TIMESTAMP

Using the TIMESTAMP parameter on a READ or WRITE command line will cause the reader to return a time value in milliseconds at the time the tag was read or written. This time value starts at 0 each time the reader is powered up.

Example:

```
OK> READ TIMESTAMP HEX(20,4) WHERE TAGID = H??, !TAG=H77
H5511111111111111 4304320 HDEADBEEF
H6622222222222222 4304340 HDEADBEEF
H8844444444444444 4304360 HDEADBEEF
OK>
```

The time values above show the relative times when the tags were read. This parameter can be used with the WRITE command in the same way.

6.3.7.2 ANTENNA

Using the ANTENNA parameter on a READ or WRITE command line will cause the reader to return the antenna number that was used during the read or write command. The Pennsylvania reader has 4 antennas and will report an antenna value that ranges from 1 to 4.

Example:

```
OK> READ ANTENNA HEX(20,4) WHERE TAGID = H??, !TAG=H77
H5511111111111111 1 HDEADBEEF
H6622222222222222 2 HDEADBEEF
H8844444444444444 4 HDEADBEEF
OK>
```

The response shown above indicates which antenna was used to read the tag during the read command. This parameter can be used with the WRITE command in the same way.

6.3.7.3 LITERALS

Literals can be used to format output to improve readability for the user. A LITERAL is any quoted text string that is included in a READ or WRITE command.

Example:

```
OK> READ “TIME=”TIMESTAMP, “ANTENNA=”ANTENNA, “DATA=”HEX(20,4)
H5511111111111111 TIME=4304320 ANTENNA=1 DATA=HDEADBEEF
H6622222222222222 TIME=4304340 ANTENNA=2 DATA=HDEADBEEF
H773333333333333   TIME=4304360 ANTENNA=3 DATA=HDEADBEEF
H8844444444444444 TIME=4304360 ANTENNA=4 DATA=HDEADBEEF
OK>
```

The above response shows how the data is formatted using the defined LITERALS. Literals can be used with the WRITE command in the same way.

7.0 Advanced Commands and Attributes

There are several additional commands and attributes that are available in the BRI. These commands and attributes should only be used with a thorough understanding of these additional BRI features.

7.1 Advanced Commands

The following additional commands are available in the BRI:
LOCK – Permanently store data in an RFID tag
READGPIO – Read the status of the 4 reader General Purpose input lines
WRITEGPIO – Write the values of the 4 General Purpose output lines

7.1.1 LOCK Command

The LOCK command is identical to the WRITE command described in section 3.2. The difference is a LOCK command causes all data written to a tag to be permanently stored. Once this command is executed the data written to the locations specified in the command can never be changed. It is important to understand the permanent nature of this command prior to using it.

The lock command is the only other active command of the BRI. It allows the user to identify, filter and lock data written to tags with minimal command interface. The structure of the command format is illustrated below. There are two optional arguments for the lock command.

LOCK { *write field* { , *write field* } } { WHERE *select field* { , *select field* } }

Using the LOCK command without any arguments, the LOCK command may be used to provide a response with all the tag IDs of all tags in the field of view of the reader as does the READ command. As there was no lock data specified in the command, no data will be written to the tags.

7.1.2 READGPIO Command

The READGPIO command allows the 4 available input ports on the reader to be read. The user is free to attach signals to the 4 input lines on the reader. These inputs can then be read by an application.

The READGPIO command has no additional command line parameters. The state of the input lines is returned as an integer value ranging from 0 to 15.

General Purpose Input	Value Returned
Input 1 active	1

Input 2 active	2
Input 3 active	4
Input 4 active	8

OK>READGPIO
15
OK>
The above response indicates that all 4 inputs are active.

7.1.3 WRITEGPIO Command

The WRITEGPIO command allows the 4 available output ports on the reader to be read. The user is free to attach signals to the 4 output lines on the reader. These outputs can then be written by an application.

The WRITEGPIO command has no additional command line parameters. The state of the output lines is set by writing an integer value ranging from 0 to 15.

General Purpose Output	Value Set
Output 1 active	1
Output 2 active	2
Output 3 active	4
Output 4 active	8

OK>WRITEGPIO 15
OK>
The above command will set all the General Purpose outputs ON.

7.2 Advanced Attributes

The following additional attribute values are available in the BRI:
TTY – Set the command line terminator to
<CR> (carriage return only) or
<CRLF> (carriage return, line feed sequence)
ECHO – Set the BRI to echo all commands back to the host
BAUD – Set the reader serial baud rate to a new value
XONXOFF – Enable XON/XOFF software protocol handshaking to the host device
CHKSUM – Enable the BRI to send and receive all commands with checksum value
IDREPORT – Enable or disable automatic reporting of all tag ids
NOTAGRPT – Enable or disable the transmission of a NO TAGS found message if no tags are found during a READ or WRITE command.
INITTRIES – Set the number of initialization tries in the reader
LOCKTRIES – Set the number of times a lock command is attempted
BRI – Set the operational mode of the reader to either Native mode or BRI mode
TAGTYPE – Set the reader to operate on a particular type of Intellitag™ tags

POWER – Set the output power of the reader to a value less than MAX or 100%

7.2.1 TTY attribute

TTY=ON/OFF

Setting the TTY attribute to OFF will set the command line terminator to <CR> (carriage return only). Setting the TTY attribute to ON will set the command line terminator to <CRLF>. These settings allow use with a dumb terminal or a terminal emulator program on a PC.

Example:
Set the TTY attribute message to ON for a <CRLF> command line terminator.

OK>ATTR TTY = ON
OK>

7.2.2 ECHO attribute

ECHO=ON/OFF

Setting the ECHO attribute to OFF will prevent the BRI from echoing command received from the host. Setting the ECHO attribute to ON will allow the BRI to echo back all commands sent to the BRI.

Example:
Set the ECHO attribute message to ON.

OK>ATTR ECHO = ON
OK>

7.2.3 BAUD attribute

BAUD=1200/2400/4800/9600/19200/38400/57600/115200

Setting the BAUD attribute to a specified value will change the baud rate of the serial port in a serial port enabled reader. This can be used to match up the reader baud rate to the host baud rate.

Example:
Set the reader baud rate to 57600.

OK>ATTR BAUD = 57600
OK>

NOTE: the OK> prompt will be sent at the old baud rate. The next command must be sent in at the new baud rate.

7.2.4 XONXOFF attribute

XONXOFF=ON/OFF

Setting the XONXOFF attribute to ON or OFF will enable or disable software handshaking with the host device.

Example:
Set the XONXOFF attribute to ON to enable software handshaking.

OK>ATTR XONXOFF = ON
OK>

7.2.5 CHKSUM attribute

CHKSUM=ON/OFF

Setting the CHKSUM attribute to ON or OFF will enable or disable the sending and receiving of message checksums to/from the BRI enabled reader.

Example:
Set the CHKSUM attribute to ON to enable the sending and receiving of message checksums.

OK>ATTR CHKSUM = ON
OK>

7.2.6 Checksum description and operation

The checksum value is sent or returned as 2 ASCII readable characters and represents a modulo 256 summing of the command or response, up to but not including the <CR> or <CRLF> characters. The checksum shall be calculated, then returned as 2 ASCII characters representing the HEX value of the modulo 256 sum. The field separator character will be inserted just before the checksum and will be included in the checksum calculation.

READ TAGIDA5<CRLF>
h0123456789ABCDEF 0A<CRLF>

In the above example, the checksum value for the response is 0x40A. The response checksum is then returned as the two least significant characters of the modulo 256 calculated checksum. In this case the characters ‘0 ‘and ‘A’ will be returned. Note that

the command and response message checksums will only be enabled upon setting the BRI attribute CHKSUM. If a checksum error is detected on an incoming message the BRI will return an error message **CKERR** to the host. For commands, the checksum is done on all characters up to the actual checksum itself. If the user decides to put an ASCII space character before the checksum value, that space character must be included in the checksum calculation.

7.2.7 IDREPORT attribute

IDREPORT=ON/OFF

Setting the IDREPORT attribute to ON or OFF will enable or disable the reporting of tag ids for READ, WRITE or LOCK commands. By default, tag ids are always returned in response to READ, WRITE and LOCK commands.

Example:
Set the IDREPORT attribute to OFF to disable the automatic sending of tag ids in response to READ, WRITE and LOCK commands.

OK>ATTR IDREPORT = OFF
OK>

7.2.8 NOTAGRPT attribute

NOTAGRPT=ON/OFF

Setting the NOTAGRPT attribute to ON or OFF will enable or disable the sending of a message “NOTAGS” if no tags can be found after a READ, WRITE or LOCK command is executes and no tags can be found in the RF field.

Example:
Set the NOTAGRPT attribute to OFF to disable the sending of tag ids in response to READ, WRITE and LOCK commands.

OK>ATTR IDREPORT = OFF
OK>

7.2.9 INITTRIES attribute

INITTRIES = n

This attribute is used to set the maximum number of tag initialization attempts the reader will use. Its argument is a numeric value in the range of 1 – 255.

Example:

OK>ATTR INITTRIES = 1
OK>

7.2.10 LOCKTRIES attribute

LOCKTRIES = n

This attribute is used to set the maximum number of LOCK attempts the reader will use. Its argument is a numeric value in the range of 1 – 255.

Example:

OK>ATTR LOCKTRIES = 1
OK>

7.2.11 BRI attribute

BRI = ON/OFF

Setting the BRI attribute to OFF will disable the BRI and return to operation in ANSI T6 command mode. This change will not take effect until the reader is reset by either turning power on or off or sending the BRI RESET command.

Example:
Set the BRI attribute to OFF to disable the BRI interface.

OK>ATTR BRI = OFF
OK>

7.2.12 TAGTYPE attribute

TAGTYPE = G1/G2/MIXED

The TAGTYPE attribute determines the population of Intellitag™ tags the reader will operate on. The default value is MIXED. When it is known that G2 Intellitag™ tags are the only tags that will be in the RF field of the reader, some performance improvements will be realized. It is recommended that this value remain at the default of MIXED.

Example:
Set the TAGTYPE attribute to G2.

OK>ATTR TAGTYPE = G2
OK>

7.2.13 POWER attribute

POWER = n

The POWER attribute determines the percent of full power the reader will emit. The value can be set from 10% to 100%. Setting this value to less than 100% will reduce the range at which tags can be read. A value of 50% will reduce the read range of tags in the RF field by approximately one-half.
Example:
Set the POWER to 50%.

```
OK>ATTR POWER = 50
OK>
```

8.0 Keywords

The keywords used in the BRI are listed here. Some keywords have alternate shortened forms that may be more acceptable for some users. In the alternate forms column, { } braces are used to indicate optional characters. The | vertical bar are used to indicate either/or sequence of characters. The ( ) brackets are used, usually with the | bar to clarify.

The first keyword READ, as an example, can by typed as R, RD, or READ. Following the alternate specification for the READ keyword we find an R followed by the optional sequence of {D|EAD}.
Since {D|EAD} is optional (enclosed in braces), an R by itself suffices for the READ keyword. However, you may also specify the optional sequence. An optional sequence has two or more parts separated by a vertical bar, so either/or applies to D or EAD. The resulting possibilities equate to R, RD or READ. Write is possible as W, WR, or WRITE.

Note: The BRI is case insensitive. Write is the same as WRITE or write.

READ	R{D|EAD}
WRITE	W{R{ITE}}
SET	SET
PRINT	PR{INT}
ATTRIBUTE	ATTRIB{UTE}
WHERE	WH(R|ERE)
TRUE	TRUE
FALSE	FALSE
ON	ON
OFF	OFF
IDTRIES	IDTRIES
RDTRIES	RDTRIES
WRTRIES	WRTRIES
ANTTRIES	ANTTRIES
ANTENNAS	ANT{ENNA}S
INTEGER	INT
HEXADECIMAL	HEX
STRING	STR{ING}
TAGID	TAG{ID}
=	=
!=	!=|=!|<>
<	<
>	>
NOT	NOT|!|<>
=!	!=|=1|<>
!	NOT|!
REPEAT	REPEAT|RPT