TagSense ZR-x API v3.8
Reader Functions

TABLE OF CONTENTS

1 Introduction

1.1 Scope of this Document

The serial API between the host and the ZR-x reader is based on a command-response model. For commands sent to tags, a given

This document describes the API for all TagSense Active Tag Readers, including the ZR-USB, ZR-232, ZR-PCMCIA, and ZR-SD. In this document, the term "ZR-x" will refer to any TagSense active tag reader. This document describes the communication protocol between the host device and the ZR-x. This includes commands that pertain to reader functions, as well as commands that pertain to tag functions. Since the ZR-x acts as a "pass through" for tag function commands, a separate document describes the API for host-active tag communication. This document only covers communication between the host and the reader.

1.2 Simple Description of Active RFID Tag Protocol

The TagSense Active RFID tags employ the IEEE802.15.4 protocol as the physical layer that has been optimized and customized for RFID applications. The TagSense air interface protocol has been built on top of the 802.15.4 layer. The IEEE 802.15.4 protocol is most well-known as the physical layer for Zigbee. However, the Zigbee protocol is not well suited for most RFID applications since it contains a great deal of overhead and requires a larger program memory on the tag to support routing tables and multi-hop/mesh capability.

The TagSense active RFID tags use a star topology, where all tags transmit their information to the reader, and all command to control the tags are sent directly from the reader. The TagSense Active RFID tag protocol is also "tag-talks-first" or TTF, which designed for ad-hoc networks, where tags are continuously entering or leaving the network. Since the reader needs to quickly process all the packets that are being received from multiple tags, the reader does a minimum amount of processing on the tag data and passes it on to the host.

For sending commands to the tags, a 2-byte CRC is used in addition to the embedded CRC used by 802.15.4 in order to improve robustness and greatly reduce the probability of spurious commands generated by RF noise.

1.3 Low-Level Behavior

The low-level protocol behavior of the tags depends on the specific type of tag (e.g. ZT-10 or ZT-50), but in general, the low-level behavior is similar for all tags.

For receiving data from tags the behavior is as follows:

- When a tag appears within radio range of the reader, and the tag transmits a packet, the reader will receive that packet within a millisecond. The tag will then remain awake for approximately 10 milliseconds to wait for any response from the reader.
- Upon receiving the tag packet, and confirming that the packet is a valid packet, the reader will immediately transmit an acknowledgement back to the tag.
- Upon receiving an acknowledgement from the reader, the tag will immediately go to sleep and stay asleep until its next transmission cycle.

The tag anti-collision protocol is described in the tag protocol description document.

For sending commands to tags, the behavior is as follows:

- When the host sends a command to the reader that is intended for a specific tag, the command will be stored in the reader command queue until a data packet from the specific tag is received.
- Upon receiving a data packet from a specific tag, the reader will immediately check its command queue and transmit the first command in its queue that is addressed to the specific tag. The reader sends this command as part of its acknowledgement.
- The tag then received this command as part of the reader acknowledgment and executes the command. With a few exceptions, the tag at this point will go back to sleep. Note that any changes in its state will not appear until the next time the tag wakes up and transmits.
- For certain types of tags (e.g. ZT-50) the tag may also transmit an acknowledgement response back to the reader, which may also contain error codes. The tag then goes to sleep.

Important note: when there are multiple readers being used, it is important to configure the settings on the readers properly to ensure that the tags respond properly to any commands that are being sent.

2 Hardware Interface and Settings

Each reader connects to a host by different means. In the case of the ZR-USB, the reader connects to host computers over USB. The drivers and instructions for installing them are available at the FTDI site (http://www.ftdichip.com/Drivers/FT232-FT245Drivers.htm). The ZR-232 can be connected through a normal serial COM port. The ZR-PCMCIA and ZR-SD can be accessed through drivers provided by TagSense.

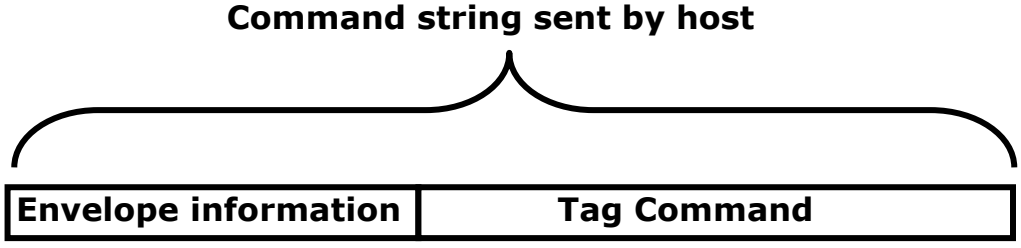
If the drivers are properly installed, the readers can be accessed over standard COM ports using standard terminal emulator programs such as Secure CRT and Hyperterm. While connecting using these applications, the following connection options must be used:

- 8 bits
- No parity
- 1 stop bit
- Baud rate – 57600 bps
- Flow Control - None

3 API Overview

3.1 Host to Reader Communications

The serial API between the host and the ZR-x reader is based on a command-response model. For commands sent to tags, a given command string sent from the host to the reader is comprised of two parts: the tag command packet and the reader envelope information.



The *envelope information* contains instructions for the reader that specify: 1) how a given command is to be delivered to the tag, 2) the operation of the reader command queue, and 3) special functions such as synchronizing the time with the tag or setting the real-time clock on the reader itself. This document describes the envelope information, which includes commands that are directed to the reader itself.

The second part of the command string is the *tag command*. Since the TagSense RFID platform is designed to support a variety of tag devices and different versions of the protocol, the reader does not parse the tag command but rather transmits the command string verbatim to the tag. Thus, it is possible to use the same reader to communicate with different versions of the tag protocol. The tag command protocol is described in separate documents. For example, the ZT-10 and ZT-50 tags use different protocols.

In addition to the commands sent to tags, it is also possible for the host to send a command packet that is used to control or query a reader parameter. This type of packet is completely independent of the tag functions and is called a *reader command*.

In summary, tag commands and reader commands are sent as separate packets and are described in Section 4 of the data sheet below.

3.2 Reader to Host Communication

There are two types of reader to host transmissions:

- *Tag Response Packet* – Upon receiving a data packet from a tag, the reader will immediately relay this packet to the host (with the addition of other information)
- *Command Response Packet* – Upon receiving a command from the host, the reader will send an acknowledgement or response to the host command. Note that both tag commands and reader commands will generate an acknowledgement from the reader.

These two basic types of packets are described in the following sections.

4 Host-to-Reader Communications

As mentioned previously, there are two categories of communication between the host and the reader. Each of these is described below.

4.1 Sending Commands to a Tag

4.1.1 Tag Command Sequence

Command strings used for controlling the ZT-x tags are sent from the host via the ZR-x reader. However, since the tagging system is designed to operate as a "tag talk first" system, the commands sent by the host will be stored in the reader until the corresponding tag wakes up and transmits its ID. All pending tag commands are stored in the reader *queue*.

When a reader receives a data packet from a tag, the reader will check to see if there are any commands waiting for the specific tag. If any commands do exist, then the reader will immediately transmit this command. *Note: only one command can be sent to the tag per tag data packet transmission. Thus, if more than one command is pending in the reader queue, then only the first command will be sent to the tag; the remaining commands will be sent in response to subsequent tag data packet transmissions. In order to save time, the tag command protocol is sufficiently flexible that is possible to combine two or more functions into the same command packet; however doing this is risky and may result in unpredictable behavior, so it should be tested first.*

For the purpose of illustration, we can describe the following simple example: The host sends a command to turn on the red LED of a tag with ID 0x1234. Upon receiving such a packet, the ZR-x will send an acknowledgement packet to the host, and then store the command in the command queue. Then, when a data packet is successfully received from tag ID 0x1234, the command will then be broadcast immediately from the reader to the tag.

4.1.2 Queue Functions and Special Commands

TagSense provides a variety of powerful commands that enable the host to control the handling of tag commands and acknowledgements.

Acknowledgement options: the host can choose how the reader handles deleting the command from the queue upon sending it with the Queue Control Bits.

There are four possible settings for every command that is sent:

- If these bits are set to "*Delete command from queue immediately upon sending,*" the reader will immediately delete the command from its queue when it transmits the command, regardless of whether the tag acknowledges receipt of the command.
- If these bits are set to "*Delete command from queue upon receiving ACK,*" the reader will keep the command in its queue even after transmitting the command, and will only delete it if receives an acknowledgement from the tag.
- If these bits are set to "*Don't delete command from queue*," the command will remain in the queue regardless of whether it is sent or acknowledged. The only way to remove a command with these settings is for the host to use a reader function command to delete it.

The host can indicate to the reader whether or not it should forward tag command acknowledgements to the host. This is done by setting/clearing the "Enable Tag Ack Forwarding" Flag.

WildCard function: The host can also specify whether the tag function command is a *wildcard* command or not. A wildcard command is one that does not have a specific tag ID destination. If the reader receives a wildcard command, it will issue the command to the first tag that is detected *regardless* of that tag's ID. If the wildcard bit is set in the descriptor field, the Tag ID field should still be included, but the reader will ignore it.

Time Synchronization: Since the tag contains a real-time-clock (RTC), it is necessary to provide some means of setting the time on the tag. However, this task is not trivial because the reader and host do not know when the tag will appear next. To solve this problem, TagSense created an RTC on the reader itself. Thus, to set the time on a tag, the host can send a "synch time" command which will be

held in the reader queue until the tag appears. At this time, the tag will then synchronize its time wit the RTC on the reader.

TagSense provides the ability to synchronize the time on the tag as part of any command sent to the tag. In the command protocol structure, there is a time synchronization bit which indicates if the host wishes to synchronize the RTC time on the tag when sending the command. If this bit is clear, the reader will not update the tag's RTC. If this bit is set, the reader will insert its own RTC into the tag command packet *at the time of transmitting the tag function command packet.*

4.1.3 Tag Command Syntax

Table 1 shows the general form of a typical tag function command packet. Fields with fixed values are indicated.

In the Descriptor field, the three Packet Type bits are used by the host to indicate what type of packet it is sending. Currently there are only 2 types of packets – reader function command packets, and tag function command packets. For tag function command packets, these bits must be set to 0b000.

Field	Value	Remarks
Start Byte	0x55	
Packet Size		This 1 byte field is the size of the packet starting from (and including) the third byte, and ending at (but not including) the end byte.
Descriptor Field		This 1 or multiple byte field indicates several properties specific to this command packet.
	0bX	Bit 7 – Synchronize Bit
	0bX	Bit 6 – Wildcard Bit
	0bXX	Bit 5:4 – Queue Control 0b00 – Delete command from queue immediately upon sending 0b01 – Delete command from queue upon receiving ACK 0b10 – Don't delete command from queue 0b11 - RFU
	0b001	Bit 3:1 – Packet Type 0b000 – Reader Function Command Packet 0b001 – Tag Function Command Packet
	0b0	Bit 0 – Field Extension Bit
Command handle		This 1 byte field, provided by the host, is a handle for this command while it is queued in the ZR-x.
Tag ID		Two byte field with the tag ID.
Tag Command Packet Data		Multiple bytes which contain the commands/settings for the tag. The format of this field is described in the document "ZT-x Protocol v3.0"
End Byte	<CR>	

Table 1 – Tag Function Command Packet Structure

4.2 Sending Commands to the Reader

In addition to sending commands to a tag, it is often necessary to send commands to the reader itself. These commands can be used to control the reader or set various parameters on the reader.

The syntax for the reader commands and the list of reader commands is described in the following sections.

4.2.1 Reader Command Syntax

Reader function packets are used to send commands to the ZR-x readers (not to the tags). These command are used to modify or query reader parameters and also to manage the tag command queue. The ZR-x will process these commands immediately. Depending on the type of command, the ZR-x will either respond with a general acknowledgement, or it will respond with information requested by the host. An example of this type of command would be a request for the current tag function command queue.

Table 2 shows the general form of a reader function command packet the host might send. Fields with fixed values are indicated.

0x denotes a hexadecimal number, 0b denotes a binary number.

<table>
<tr><th>Field</th><th>Value</th><th>Remarks</th></tr>
<tr><td>Start Byte</td><td>0x55</td><td></td></tr>
<tr><td>Packet Size</td><td>1 byte</td><td>This 1 byte field is the size of the packet starting from (and including) the third byte, and ending at (but not including) the end byte.</td></tr>
<tr><td>Descriptor Field</td><td>0b0000
0b000

0b0</td><td>This 1 or multi-byte field indicates several properties specific to this command packet.

<table>
<tr><td>Bit 7:4</td><td>RFU (can be set to zeroes)</td></tr>
<tr><td>Bit 3:1</td><td>Packet Type
0b000 – Reader Function Command Packet
0b001 – Tag Function Command Packet
0b010 – RFU
0b011 - RFU
0b100 - RFU
0b101 - RFU
0b110 - RFU
0b111 - RFU</td></tr>
<tr><td>Bit 0</td><td>Field Extension Bit</td></tr>
</table>
</td></tr>
<tr><td>Reader Command</td><td></td><td>Single byte reader command.</td></tr>
<tr><td>Reader Command Arguments</td><td>Optional</td><td>Optional field containing arguments to the reader command.</td></tr>
<tr><td>End Byte</td><td><CR></td><td></td></tr>
</table>

Table 2 – Reader Function Command Packet Structure

In order to manage all the commands that are pending in the reader command queue, several reader function commands are implemented which allow the host to query and manage the command queue. Up to 32 (SUBJECT TO CHANGE) tag function commands total can be queued at any one time. In order to distinguish one command form another in the command queue, the host is allowed to give each tag function command is given a unique, 1 byte *handle.* (see table 1). These handles are essentially unique labels that are used to identify which tag function commands are currently in the queue.

4.2.2 List of Reader Commands

Table 1 shows the general form of a typical tag function command
The actual command and any necessary arguments are contained in the Reader Command Packet Data field of the Reader Command Packet (see Table 2). Table 3 lists the command/argument bytes that can be inserted in this field. Only one command can be included in a given reader packet.

Command	Value	Argument	Remarks
Read Firmware Version	0x00	<none>	Reader will return its firmware version. Older firmware versions of the ZR-x will return an error to this request.
Read Queue Entries	0x01	<none>	Reader will return the command handles for all queued tag commands
Read Queue Length	0x02	<none>	Reader will return the size of the queue (i.e. number of commands)
Delete Queue Entry	0x03	1 byte command handle	Reader will delete tag command with handle specified.
Read Queue Entry Command	0x04	1 byte command handle	Reader will return the destination tag ID and payload for the handle specified.
Set Reader ID	0x05	2 byte reader ID	Reader will set its ID to the argument of this command.
Read Reader ID	0x06	<none>	Reader will return its 2 byte ID.
Flush Queue	0x07	<none>	Reader will remove all tag commands from its queue
Set Reader Time	0x08	4 byte UTC	Reader will synchronize its internal clock to the argument of this command
Query Reader Time	0x09	<none>	Reader will return its 4 byte UTC.
Append Reader Time	0x10	1 Byte: 01= ON 00 = OFF	This controls whether or not the reader will append the value of its real-time clock (time stamp) to end of every tag packet that is received and forwarded to the host.

Table 3 – Reader Commands

Command	Value	Argument	Remarks
TX-test	0xF1	<none>	Reader will transmit continuously for approx 1 minute. This is used for radio testing only.
RX-test	0xF2	<none>	Reader will receive continuously for approx 1 minute. This is used for radio testing only.
Enable/Disable PassiveMode	0xF3	1 Byte 00=OFF 01=ON	If passive mode is set to ON, this puts the reader in "listen only" mode. In this mode of operation, the reader will not send any acknowledgments or commands to the tags. This is often used to minimize interference and collision when multiple readers are present.

Table 4 – Reader Commands

5 Reader-to-Host Communications

There are 3 types of communications between the Reader and the Host:

- **Reader command packet response** – this is the reader's response to a reader command packet sent by the host
- **Tag Command Packet Response** – this is the readers response to a tag command packet that is sent by the host
- **Tag Transmission Packet** – this is the packet that is sent by a tag and received by the reader and forwarded to the host

5.1 Reader Command Packet Response

The ZR-x sends a response packet immediately after receiving any type of command packet. The response packet will either contain a general acknowledgement, value or values requested by the host, or an error code.

Table 4 shows the general form of a packet that is sent by the reader to the host in response to a *reader command*. Fields with fixed values as well as optional fields are indicated.

<table>
<tr><th>Field</th><th>Value</th><th>Remarks</th></tr>
<tr><td>Start Byte</td><td>0xAA</td><td></td></tr>
<tr><td>Packet Size</td><td></td><td>This 1 byte field is the size of the packet starting from (and including) the third byte, and ending at (but not including) the end byte.</td></tr>
<tr><td>Descriptor Field</td><td>0x00</td><td>This field indicates the type of response

<table><tr><td>Bit 7:4</td><td>RFU</td></tr><tr><td>Bit 3:1</td><td>Packet Type
0b000 – Reader Function Command Response Packet</td></tr><tr><td>Bit 0</td><td>Field Extension Bit</td></tr></table></td></tr>
<tr><td>Reader Command Echo</td><td></td><td>This single byte is an echo of the reader command.</td></tr>
<tr><td>Response Code</td><td></td><td>This 1 byte field indicates the type of response packet.

0x00 – General Acknowledgement
0x01 – Acknowledgement with returned values
0x02 – Error: Invalid Character
0x03 – Error: Invalid Start Byte
0x04 – Error: Packet Too Short
0x05 – Error: Packet Too Long
0x06 – Error: Queue Full
0x07 – Error: Invalid Descriptor Field
0x08 – Error: Incorrect Length Byte
0x09 – Error: Unimplemented Command
0x0A – Error: Invalid Handle
0x0B – Error: Invalid Argument
0x0C – Error: Internal Error</td></tr>
<tr><td>Response Values</td><td>Opt.</td><td>This single or multi-byte field is only present if the value of the response code is 0x01.</td></tr>
<tr><td>End Byte</td><td><CR></td><td></td></tr>
</table>

Table 4 – ZR-x Response Packet Structure

5.2 Tag Command Packet Response

When the host sends a *tag command* packet to the reader, the reader will respond with a packet having the structure in the table below. Fields with fixed values as well as optional fields are indicated.

Field	Value	Remarks
Start Byte	0xAA	
Packet Size		This 1 byte field is the size of the packet starting from (and including) the third byte, and ending at (but not including) the end byte.
Descriptor Field	0x02	This field indicates the type of reponse Bit 7:4 – RFU Bit 3:1 – Packet Type 0b001 – Tag Function Command Response Packet Bit 0 – Field Extension Bit
Tag Command Handle		This single byte is the command handle.
Response Code		This 1 byte field indicates the type of response packet. 0x00 – General Acknowledgement 0x01 – Acknowledgement with returned values 0x02 – Error: Invalid Character 0x03 – Error: Invalid Start Byte 0x04 – Error: Packet Too Short 0x05 – Error: Packet Too Long 0x06 – Error: Queue Full 0x07 – Error: Invalid Descriptor Field 0x08 – Error: Incorrect Length Byte 0x09 – Error: Unimplemented Command 0x0A – Error: Invalid Handle 0x0B – Error: Invalid Argument 0x0C – Error: Internal Error
Response Values	*Opt.*	This single or multi-byte field is only present if the value of the response code is 0x01.
End Byte	<CR>	

Table 5 – ZR-x Response Packet Structure

5.3 Tag Transmission Packet Structure

This is the most common type of data packet.
When the reader receives a data packet from a tag, the ZR-x reader will immediately relay this data packet to the host. The reader wraps the tag's transmission within a start and end byte, and also adds several other bytes including the packet length byte, packet type, and RSSI value.

Table 6 shows the general form of a ZR-x response packet. Fields with fixed values as well as optional fields are indicated.

<table>
<tr><th>Field</th><th>Value</th><th>Remarks</th></tr>
<tr><td>Start Byte</td><td>0xAA</td><td></td></tr>
<tr><td>Packet Size</td><td></td><td>This 1 byte field is the size of the packet starting from (and including) the third byte, and ending at (but not including) the end byte.</td></tr>
<tr><td>Descriptor Field</td><td>0x04</td><td>This field indicates the type of response
<table>
<tr><td>Bit 7:4</td><td>RFU</td></tr>
<tr><td>Bit 3:1</td><td>Packet Type
0b010 – Tag Transmission Packet</td></tr>
<tr><td>Bit 0</td><td>Field Extension Bit</td></tr>
</table></td></tr>
<tr><td>Transmitted Tag Packet</td><td></td><td>This is a multi-byte field containing the contents of the packet transmitted by a ZT-x tag. This fieldis variable in length and is described in the ZT-10 and ZT-50 protocol documents.</td></tr>
<tr><td>RSSI Field</td><td></td><td>This 1 byte field contains the received signal strength of the tag as measured by the reader.</td></tr>
<tr><td>Time Stamp (optional)</td><td>4 Bytes</td><td>When the time stamp feature is enabled by the host, the reader will append a 4 byte time stamp to the data packet. These 4 bytes will not be included when the time stamp feature is disabled.</td></tr>
<tr><td>End Byte</td><td><CR></td><td></td></tr>
</table>

Table 6 – Transmitted ZT-x Packet Structure

5.4 Communication Examples

5.4.1 Detailed Example #1

The following example illustrates an exchange between the host and the ZR-x. In this example, the host requests the current queue list from the reader, and the reader returns a response:

Command string sent to reader: 55020081
Response from reader: AA06008101010304

In order to understand the meaning of these strings, the strings can be parsed and labeled as follows:

Field	Value	Remarks
Start Byte	0x55	
Packet Size	0x02	
Descriptor Field	0x00	Indicates a Reader Function Command Packet
Reader Command Packet Data	0x81	Read Queue Entries Command
End Byte	<CR>	

Table 7 – Example Queue Query Command

Field	Value	Remarks
Start Byte	0xAA	
Packet Size	0x06	
Descriptor Field	0x00	Indicates Reader Response Packet
Reader Command Echo	0x81	Read Queue Entries Command
Response Code	0x01	Acknowledgement with returned values
Response Value 1	0x01	Tag Function Command 1 is in queue
Response Value 2	0x03	Tag Function Command 3 is in queue
Response Value 3	0x04	Tag Function Command 4 is in queue
End Byte	<CR>	

Table 8 – Example Queue Query Response

5.4.2 Detailed Example #2

The following example illustrates how the reader responds when a command is sent to the tag. The following tag command string happens to be the command to exit data dump mode:

Command string sent to reader: 550812C2000D80100200
Response from reader: AA0302C200

In order to understand the meaning of these strings, the strings can be parsed and labeled as follows:

Field	Value	Remarks
Start Byte	0x55	
Packet Size	0x02	
Descriptor Field	0x00	Reader Function Command Packet
Reader Command Packet Data	0x81	Read Queue Entries Command
End Byte	<CR>	

Table 9 – Example of Tag command

Field	Value	Remarks
Start Byte	0xAA	
Packet Size	0x03	There are 3 payload bytes in this packet
Descriptor Field	0x02	Indicates Tag Command Response Packet
Reader Command Echo	0xC2	Tag Command handle
Response type	0x00	General acknowledgement (if an error occurred, this would be nonzero)
End Byte	<CR>	

Table 9 – Response to Tag Command

5.4.3 Detailed Example #3

The following example illustrates how the reader relays a packets that is received from a tag. The tag transmits asynchronously (autonomously) so this communication is not initiated by the host an there is no host command string.

Data from reader: AA0E04000DFFFF0430843007C0800FF7

In order to understand the meaning of these strings, the strings can be parsed and labeled as follows:

Field	Value	Remarks
Start Byte	0xAA	
Packet Size	0x0E	Number of payload bytes
Descriptor field	0x04	Indicates a Received Tag Packet
Descriptor Field	0x000D	ID of Tag
Reader Command Packet Data	0xFFFF	ID of Reader
Tag DATA		Multiple bytes of data
End Byte	<CR>	

Table 9 – Tag Command example

5.4.4 Other Command String Examples

The following is a list of sample command strings that can be sent by the host to the reader to control reader functions.

Query the Firmware Version

55020000

Query the Reader queue

55020001

Delete command with handle 0x7A from the queue

550300037A

Query the reader ID

55020006

Set reader ID to 1234

550400051234

Read command having handle 0x84

5503000484

Flush (erase) the reader command queue

55020007

Set reader time on the reader with
the Universal time value = 0EA26731

550600080EA26731

Query the reader time

55020009

6 Revision History

Version 3.0 –

Initial document

Version 3.1 -

Corrections:

Update: Added Descriptor Field, as well as functionality to handle tag acknowledgements.

Version 3.2 -

Corrections:

Update: Extensive changes.

Version 3.3 –

Corrections:

Update: Added wildcard functionality to tag function commands. Added "Flush Queue" reader function command.

Version 3.4 –

Corrections: Removed the "delete upon ack" bit from the tag function command frame, and replaced it with two bits to handle queue management of the command.

Version 3.5 – Corresponds to ZR-10 v3.3 firmware

Update: Added new reader function commands "Set Reader Time" and "Read Reader Time".

Version 3.6 – corresponds to v3.4 firmware
Update: Updated and organized the entire document

Version 3.7 – 06-03-08- corresponds to v3.7-3.10 firmware
Update: Updated format and added radio-testing commands

Version 3.8 – 11-15-08- corresponds to v3.11? firmware
Update: added "Listen-only" mode for reader, added time stamp output, added CRC description.