TagSense ZT-x API v3.21
Tag Functions (DRAFT)

TABLE OF CONTENTS

1 Introduction

This document describes the communication API between a host device and a TagSense active tag. The host is able to communicate with one or more active RFID tags by sending commands strings through the TagSense Active RFID reader (e.g. ZR-10, ZR-USB).

In order to preserve battery life, the Version 3 protocol and API has been designed to minimize the number of air transmissions required to communicate information from the host to the tag. In addition, in order to minimize the likelihood of bit errors and increase reading range, a great deal of effort has been made to create a protocol that minimizes the average length of data packets transmitted by both the tags and the readers.

TagSense provides various active RFID protocols design for different applications. Unlike most passive RFID system, most of the active RFID tags created by TagSense employ an **asynchronous** protocol, which is optimized for battery life. However, other **synchronous** protocols are also available for applications where a high tag density and high data throughput is necessary (at the expense of battery life).

In the protocol used by the ZT-50, all communication is initiated by the ZT-x tag and is not centrally controlled by the RFID reader. The tag spends the majority of time in sleep, waking up only periodically to transmit beacon packets. This allows battery power to be conserved, since the tag does not spend 100% of its time listening for reader packets. Immediately following the transmission of a beacon packet, the tag listens for a brief period of time for a response from the reader. In the general case, this response is simply an acknowledgement from the reader that it received the tag's packet. At this point, the transaction is complete, and the tag will return back to sleep mode.

However, if a host device connected to the reader has sent a tag function command packet directed at that tag's ID, the reader response will contain that command packet. Upon receipt of this tag command packet, the tag will set its variables accordingly, and depending on certain fields in the tag command packet, it may or may not transmit a tag response packet back to the reader. At this point, the transaction is complete, and the tag will return back to sleep mode.

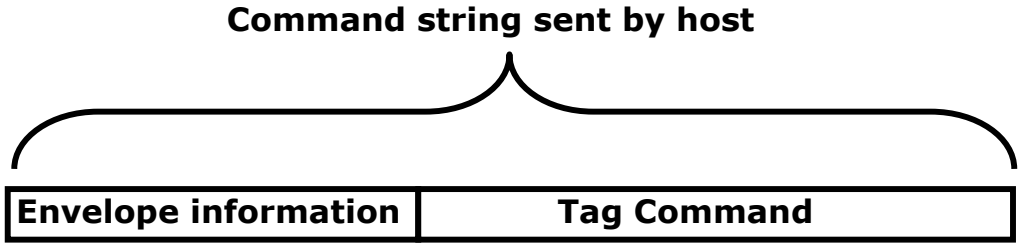
In this v3.x document, we refer to the tag as the ZT-x, which at the moment includes the ZT-50 active tags. While the command set is identical for all 3.x tags, the ZT-50 contains an additional EEPROM memory chip which enables greater data logger memory capacity.

Version 3 of the ZT-x protocol has been designed to make it as user customizable as possible, making it a powerful platform that can be used for a wide variety of applications. To this end, the API allows the user in depth control over all ZT-x functions, from data logging to configurable I/O pins to user definable state transitions. The next section describes several unique features of the TagSense ZT-x.

2 Protocol Overview and Command Structure

The TagSense *physical-layer* air interface protocol used between an active tag and the reader conforms to the standard IEEE 802.15.4. TagSense has also preserved to a large extent the Media Access Control (*MAC layer*) in the IEEE 802.15.4 protocol.

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* is described in a separate document which describes the reader commands. 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.

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 version of the tag protocol. This document describes Version 3 (v3.x) of the TagSense Active RFID protocol. Previous versions of the tag protocol are described in a separate documents.

Upon receiving a command packet from a reader, the tag will respond with an *acknowledgement packet*. (note this feature can be disabled if desired).

2.1 GPIO (General Purpose I/O)

The ZT-x has 11 general purpose I/O pins. These pins can be configured to be digital inputs, digital outputs, analog inputs. Four of these pins have a special function: TTL level RS-232 RX, TTT-level RS-232 TX, and 2 external interrupts. Not every pin is capable of being set in every possible configuration, but with up to 7 analog inputs or up to 11 digital input/outputs (or some combination of the two), the ZT-x can be a powerful tool for either monitoring sensors, actuating external electrical devices, or interfacing to an external microcontroller.

2.2 Tag Parameter Reporting

In addition to being able to remotely set and configure the various state variables on the tag, it also possible to control which parameters are to be "displayed" or reported by the tag in each bacon data packet. This topic is known as reporting.

The most common application for reporting is to view the value of sensors that are connected to the tag analog or digital inputs.

All of the major parameters of the tag (such as the transit interval and RSSI) and the value of the analog or digital inputs to the tag can be reported in the tag beacon packet. By default, most of the paramter fields in the tag beacon packet are disabled (turned OFF) in order to save battery life and increase reading distance. However each of these fields can be enabled (turned ON) by means of enabling the reporting of the specific parameter that is needed.

If the user enables the reporting of a particular parameter, then this parameter will be permanently reported in every subsequent data packet until the reporting is disabled (turned OFF).

2.3 Tag Parameter Queries

[THIS FUNCTION Is UNDER DEVELPOPMENTAND DOES NOT YET EXIST]

In some cases, it is not necessary to enable reporting in every beacon packet, but rather the user simply wants to query a particular parameter to find out it current setting. This function is also supported y the TagSense version 3 protocol and is known as a parameter *query* or *reading*.

In the case of a query, the query request is made via the command packet that is sent from the reader to the tag. This command will contain a request for the tag to reply with the parameter value.

Upon receiving the command, the tag will reply with the parameter value in the acknowledgement packet that it sends back to the reader. Note that for this to work properly, the user must make sure that the tag acknowledgement function is enabled.

2.4 Real Time Clock

The ZT-x contains a real time clock that has been implemented in firmware and is accurate to better than 15 minutes over a week. This performance is quite good, given the fact that the tag is put to sleep in between tag transmissions. An integrated calibration algorithm enables the ZT-x to maintain its clock accuracy despite changes in battery voltage or ambient temperature.

The time on the tag is represented by a 4-byte time integer which is equal to the number of seconds elapsed since January 1, 2000.

2.5 User Programmed Modes

One of the most powerful features of the ZT-x is the ability for the user to set up to 8 programmable states or *modes* of operation. A *mode* is defined by a set of internal state variables, which includes the configuration of the GPIO, the transmission interval, the sampling rate for data logging, etc.

The ability to create preset modes enables the tag to automatically and quickly switch from one behavior to another. For example, to conserve battery power, a user might want the tag to transmit only once an hour most of the time; but if an external sensor (such as a motion sensor) is triggered, then it is desirable for the tag to transmit more often. In this case, one mode can be set which transmit once per hour, and a second mode can be set which transmits once per second (for example). Similarly, it is possible to configure the tag to sample a sensor input at one rate, but then change to another sampling interval in another mode.

There are several ways that a tag can be prompted to switch from one mode to another. The transition between modes is done through user-definable *triggers*. A trigger can be in the form of an analog input signal, a digital input signal, or an external sensor. These are described briefly below:

1) **Alarms**: An alarm is a time-based trigger. It is possible to set a time alarm such that a particular mode will be entered at a specific time. In addition to setting a specific start time to enter the mode, the user must also specify the duration for a particular mode. This is very analogous to programming a VCR for television. For example, the tag can be programmed to enter mode #4 at 12:45 pm and continue in this mode for 2 hours. At the expiration of this time, the tag will return to the previous mode or to the default mode (mode 0).

2) **External Digital Trigger** (interrupt): A tag can be programmed to enter a specific mode when triggered by an external digital input (0 or 1, Active High or Active Low). Examples of this include a pushbutton, a contact switch, a motion sensor, or a magnetic reed switch.

3) **Analog Sensor Trigger:** It is also possible to program a tag to enter a specific mode through the use on an analog sensor. In this case, the user must define an *analog trigger threshold*, and an *analog trigger direction*. The particular mode will be activated when the analog sensor value (10-bit integer value) crosses the threshold. The trigger direction defines whether the mode is activated upon crossing the threshold from above or from below.

Special Cases:

There are a couple special modes that are also defined:

Data dump mode: This mode is used to put the tag in a special state for the purpose of transferring and erasing its data memory. The process of downloading data from a tag is described in detail in a separate section.

Power Reset mode: When the battery is removed and replaced, the tag will briefly enter a special power reset mode. This allows a reader to send a command to the tag when a new battery is inserted and also allows the tag to generate a special memory record that to annotate this event.

2.6 Custom Program Sequence

[THIS FUNCTION Is UNDER DEVELPOPMENTAND DOES NOT YET EXIST]

For certain applications, there may be a need for the tag to perform a sequence of operations every time it wakes up. In this case, a simple programmable mode is not sufficient to meet this need. Examples of such applications include the following:

- If the tag is connected to an external microcontroller device, every time the tag wakes up, it might need to send a command sequence to the external device and wait for a response.

- In some sensor applications, the tag needs to perform some measurement sequence every time it wakes up, such as flash an LED 10 times and then sample the A/D analog input.

- Certain applications require that the tag perform some processing on the sampled data (such as averaging multiple sensor readings or frequency counting) and then write the result into data logger memory.

For such applications, TagSense has provided a means for the user to create a custom command sequence. This sequence is treated as one of the programmable modes and is activated by setting the appropriate bits in the Trigger Field of the command packet.

2.7 Real-Time Location (RTLS) Functions

[THIS FUNCTION IS UNDER DEVELPOPMENTAND DOES NOT YET EXIST]

For many manufacturing or asset tracking applications, it is necessary to provide some means for monitoring the physical location of a tag. Although an approximate location algorithm can be derived using the Received Signal Strength Indicator (RSSI) signal, the RSSI signal can be affected by environmental parameters and is not very reliable.

As an alternative to using the RSSI signal, the ZT-x also has the ability to record the ID of any reader which communicates with the tag. Using this function, it is possible to create a method for identifying geographic zones by using multiple readers, each with its own unique ID. TagSense provides a simplified reader device called a *locator beacon* which serves this purpose. A locator beacon will continuously transmit its *location ID code*. The tag records this location ID code and includes it in its normal beacon packet transmissions.

In addition to recording the location ID code, the ZT-x is also capable of measuring and recording the RSSI of the locator beacon transmission. By knowing the location ID and the location RSSI, it is possible to create a reasonably robust system that tracks the location of active tags.

The protocol syntax for these functions is described in the RTLS Field section.

3 Host to Tag Communication

There are several types of actions the host might want the tag to undertake. These include setting a state variable, requesting the current of a state variable, enabling or disabling some feature or reported value in the tag's beacon packet, or putting the tag into one of 8 modes. The protocol is designed to be flexible enough to allow the host to accomplish any number of these tasks all within a single command. This is accomplished through a hierarchical, bit-masked, command structure.

At the top level of the hierarchy is the *Global Packet Control* field. This field can be one or more bytes, where each bit represents the presence of a specific group of related tag functions. For example, if the user memory control bit in the *Global Packet Control* field is set, this indicates the presence of the *User Memory Control* field. In turn, the individual bits in the *User Memory Control* field indicate the presence of specific tag functions related to the user memory (e.g. writing a value, retrieving a value, etc.) If the user memory control bit in the *Global Packet Control* field is not set, then the *User Memory Control* field is not present in this command packet, and the packet contains no commands that control any feature of the user memory.

All control fields (e.g. *Global Packet Control, User Memory Control,* etc.) can be either single or multiple bytes in length. This is indicated through a control field extension bit, which is always the LSB of every control field byte. For example, if the LSB of the first byte of the *Global Packet Control* field is clear, then the second byte of the *Global Packet Control* is not present; conversely, if the LSB is set, the second byte is present. This allows the protocol to be extendable for new functions to be added in the future.

Every command packet also contains a special *command field* which resides outside the command hierarchy. This field contains information relevant to the command, but does not itself set or change any feature on the tag.

Table 1 below shows the high level structure of a ZT-x command.

Field	Remarks
Command Header	Single or multiple byte value that contains special information specific to this command.
Packet Control Header	Single or multiple byte value that indicates which fields (e.g. Network Field, GPIO Field, etc.) are contained in this packet.
Network Field	Multi-byte field that handles network related commands (e.g. setting transmission interval). This field is only present if so indicated by the *Packet Control Header.*
GPIO Field	Multi-byte field that handles GPIO related commands (e.g. setting the Pin 1 as a digital output). This field is only present if so indicated by the *Packet Control Header.*
Alarm Field	Multi-byte field that handles commands related to alarms and time-triggered events (e.g. setting the start time for Mode 1). This field is only present if so indicated by the *Packet Control Header.*
Trigger Field	Multi-byte field that handles trigger related commands (e.g. setting Mode 3 to be entered if sensor 1 readings exceed a certain value). This field is only present if so indicated by the *Packet Control Header.*
User Memory Field	Multi-byte field that handles functions related to the user memory (e.g. writing a 16 byte value to bank 5). This field is only present if so indicated by the *Packet Control Header.*
Data Logging Field	Multi-byte field that handles data logging related commands (e.g. setting the tag to dump all data records stored since previous dump). This field is only present if so indicated by the *Packet Control Header.*
RTLS Field	Multi-byte field that handles data relating to the Real-Time Location System functions. This field is only present if so indicated by the *Packet Control Header.*

Table 1 – Tag Command Packet Structure

The following sections describe the format for each of these fields in greater detail.

3.1 Command Header

This header contains special information relevant to all commands in the packet. For example, it indicates whether the tag should provide an acknowledgement upon receipt of this command. Also, since the tag can have several programmed "modes," this field indicates to which mode the parameters in this command apply. Table 2 shows the structure of this field.

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Tag ACK	Mode2	Mode1	Mode0	RFU	RFU	RFU	Command Field Extension Bit

Bit 7 Tag ACK

If this bit is set, the tag is required to send an acknowledgement packet back to the reader. If clear, the tag does not return an acknowledgment

Bit 6:4 Mode

These three bits indicate which mode the mode-dependent parameters in this packet apply to. For example, if this value is set to 0, and sensor 1 and sensor 2 reporting is enabled in the GPIO section of the command, then the tag will report sensor 1 and sensor 2 information when it is in mode 0; otherwise, it will behave according to the parameters set for that mode.

Bit 3:1 RFU

For the version 3 protocol, these bits MUST be set to 0.

Bit 0 Command Field Extension bit

If this bit is set, there is a second command field byte that follows immediately after this one. This allows for future expansion of the protocol. In this version, this bit should be set to 0.

3.2 Packet Control Header

This field indicates which fields are present in this packet. If more than one field is present, their order in the overall command packet will be the same as the order of their respective control bits from MSB to LSB. Table 3 shows the structure of this field.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Network Field Mask Bit	GPIO Field Mask Bit	Alarm Field Mask Bit	Trigger Field Mask Bit	User Memory Field Mask Bit	Data Logging Field Mask Bit	RTLS Field Mask Bit	Packet Control Header Extension Bit

Bit 7 Network Field Mask Bit

If this bit is set, the Network Field is present; otherwise, it is not present.

Bit 6 GPIO Field Mask Bit

If this bit is set, the GPIO Field is present; otherwise, it is not present.

Bit 5 Alarm Field Mask Bit

If this bit is set, the Alarm Field is present; otherwise, it is not present.

Bit 4 Trigger Field Mask Bit

If this bit is set, the Trigger Field is present; otherwise, it is not present.

Bit 3 User Memory Field Mask Bit

If this bit is set, the User Memory Field is present; otherwise, it is not present.

Bit 2 Data Logging Field Mask Bit

If this bit is set, the Data Logging Field is present; otherwise, it is not present.

Bit 1 RTLS Field Mask Bit

If this bit is set, the Real-Time Location System (RTLS) Field is present; otherwise, it is not present.

Bit 0 Protocol Control Header Extension Bit

If this bit is set, there is a second protocol control header byte that follows immediately after this one. This allows for future expansion of the protocol. In this version, this bit should be set to 0.

3.3 Network Field

This data field includes parameters that manage the radio communications between one tag and another. For historical reasons, and the OSI data layer convention for IEEE802.15.4, this field is denoted as the *network control field*. Examples of parameters included in this field are the transmission interval, transmission power, tag ID, firmware version (read-only), most recently seen beacon locator device, and mechanisms for spreading out in time beacon packets from multiple tags.

The network control header is nominally 2 bytes in length. The first byte is a mask byte while the second byte is an action byte. In a command sent by the reader to the tag, the mask field is used to indicate which parameters are being modified and which parameters are being left alone. For those parameters which are being modified, the action byte indicates how those parameters are to be modified. Bits in the action byte are processed only if the bit in the corresponding position in the mask byte is set to 1; otherwise the action bit is ignored and the corresponding parameters are left unmodified.

As with all control headers, the LSB position is a field extension bit. If it is set to 1 on both the mask and action bytes, a second pair of mask/action control bytes immediately follows the first pair. If either of the LSB bits are set to 0, then the second pair is not included in the packet. If any arguments are present, they will follow the final pair of mask/action control bytes, in the same order as their corresponding bits in the control bytes, from MSB to LSB.

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Halt Sleep Mask Bit 0	Time Mask Bit 0	TX Interval Setting Mask Bit	TX Interval Report Mask Bit	TX Power Setting Mask Bit	TX Power Reporting Mask Bit	Tag ID Mask Bit 0	Network Control Field Extension Mask Bit

Table 4 shows the structure of the Network Control Mask Header 1

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Tag Halt Bit	Enable/ Disable Transmit on Interrupt	Set/Read TX Interval Bit	Enable/ Disable TX Interval Report Bit	Set/ Read TX Power Bit	Enable/ Disable TX Power Report Bit	Set/ Read Tag ID Bit	Network Control Field Extension Bit

Table 5 shows the structure of the Network Control Action Header 1.

3.3.1 Halting or Hibernating a Tag

Bit 7 Tag Halt Bit

There are many situations in which it is desirable to set the tag to a deep sleep for an extended period of time. Since the term "sleep" is used ambiguously and informally by many RFID companies, we prefer to use the term *"Halt"* or *"hibernate"* to distinguish this state from the sleeping or idling state that the tag enters in between transmissions. Unlike normal sleep which the tag performs periodically, a tag that has been halted will remain asleep *indefinitely* until it is woken up manually through the use of an external sensor or switch (such as a push button, a motion switch, or a magnetic reed switch).

The Tag Halt state is similar in principle to using the Trigger field to wake up the tag when activated by an external sensor. However, there are important differences between these 2 cases:

- In the tag halt sate, all tag functions except the external interrupt is turned OFF. All real-time clock and data logging is disabled. *As a result, the tag halt state has the minimum power consumption of any tag state*.
- Using the normal external trigger function, when the tag goes to sleep, all the data logging functions and real-time clock remain active.

The primary use of the tag halt state is to provide a means of turning off the tag without removing the battery. Once halted, the tag can be shipped or stored for long periods of time (many months or years).

When the tag is woken up, the tag will perform a power reset, similarly to the case of replacing the battery.

The use of the Tag Halt Bit is as follows:

- If Sleep Mask Bit 1 is 0, this bit is ignored. Otherwise:
- If this bit is set, the tag will go to sleep.
- If this bit is clear, the tag will ignore it.

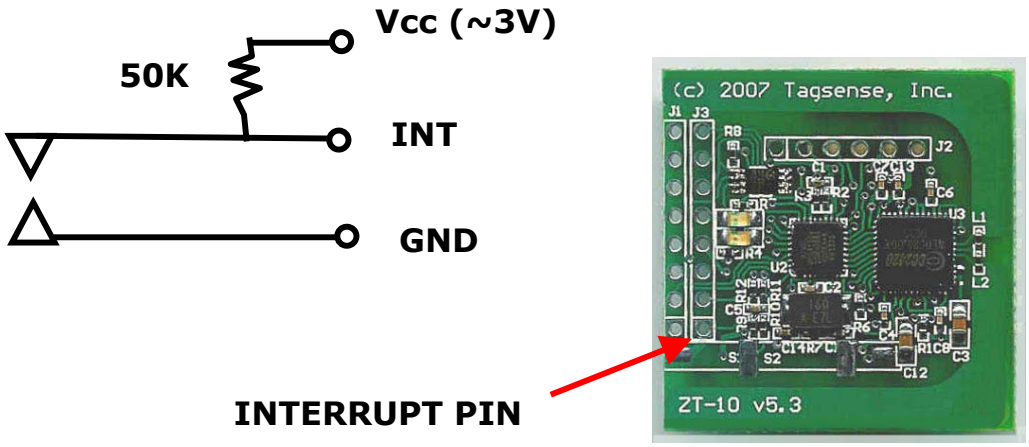
3.3.2 Enabling/Disabling Transmit on Interrupt

Bit 6 Enable/Disable Transmit on Interrupt

If Bit 6 is set, this will enable the transmit on interrupt. In this case, the tag will transmit a packet whenever the INT1 pin is pulled LOW. When Transmit on Interrupt is enabled, the tag will still transmit at periodic intervals as set by the user.

The Transmit on Interrupt function can be used for many applications, and also helps a great deal to save battery life. One example would be to attach a motion sensor to the tag and enable the tag to transmit only when the tag is moved. In order to save battery life, the tag can be programmed to transmit once every 60 minutes, for example. However, if transmit on interrupt is enabled, the tag will transmit a data packet immediately when the tag is moved or shaken.

The external interrupt pin can also be attached to a "wake up circuit". This would provide a means to trigger a transmission using some sort of external stimulus, such as an electromagnetic field or an optical (infrared) signal, or even a microphone. A common application for military sensor network is to trigger a transmission when a sound or vibration is detected.



The interrupt pin is normally high (Vcc) and will trigger a transmission when there is a falling edge (pulled down momentarily to GND). If using a pushbutton or magnetic reed switch, these should be connected between the Interrupt pin and GND. Waking up from deep sleep or Halted state is triggered by detecting a low logic level instead of an edge.

3.3.3 Setting Tag Transmit Interval

Bit 5 Set/Read TX Interval Bit

The transmit interval of a tag is configured by setting bit 5 of the control header in the Network Field. This bit is called the TX Interval bit.

- If the TX Interval Mask bit is 0, the TX interval remains unchanged.
- If the TX Interval Mask bit is 1 and the TX Interval action bit is 0, the tag will return its one byte TX Interval value in its acknowledgement packet.
- If TX Interval Mask Bit is 1 and the TX Interval action bit is also 1, then a 1 byte argument is included in this packet following the network control field bytes. This value represents the transmission interval the tag should use between beacon packets. The table below maps this value to the number of seconds. This variable corresponds to the specific user mode given in the Command Header.

Value	Interval
0x01	.5s
0x02	1s
0x03	2s
0x04	5s
0x05	10s
0x06	20s
0x07	30s
0x08	1m
0x09	2m
0x0A	5m
0x0B	10m
0x0C	15m
0x0D	30m
0x0E	45m
0x0F to 0x15	For all parameter values (X) from 0x0F to 0x15, the corresponding time interval (T) in minutes

	can be calculated as: T = 30*(X-13) Maximum value is 0x15=240min=4 hours
0x15-FE	undefined
0xFF	Activate "Spreading function" (see text description)

3.3.4 Description of Tag "Spreading Function"

Setting the interval value of 0xFE will activate the *spreading function*. This is a special function that was added to improve performance when using multiple tags. In some cases where there are many tags operating simultaneously, it is possible that some number of tags will become synchronized with each other such that they will wake up and transmit at nearly the same time and thus cause increased number of data collisions. In this case, it is desirable to have some means of spreading out the tag transmissions to that they will not wake up at the same time. The spreading function included in the TagSense version 3 protocol provides a means for addressing this problem and improve performance when many tags are present.

When the interval value of 0xFE is used, the tag will not change its current transmit interval but will instead *offset* the current transmit interval by performing a delay equal to the current transmission interval divided by 2 (except for the lowest transmission interval). Another way of describing this function is that the tag will shift its phase by half a period (180 degrees).

3.3.5 Enabling/Disabling Reporting of Tag TX Interval

Bit 4 Enable/Disable TX Interval Report Bit

- If TX Interval Reporting Mask Bit is 0, this bit is ignored.

Otherwise:

- If this bit is set, the tag will enable reporting of its 1 byte TX Interval in every beacon packet transmission.
- If this bit is clear, the tag will disable further reporting of its 1 byte TX Interval in every beacon packet transmission.

The reporting will be enabled or disabled for the specific user mode given in the Command Header.

3.3.6 Setting the Tag Transmit Power

Bit 3 Set/Read TX Power Bit

- If TX Power Setting Mask Bit is 0, this bit is ignored.

Otherwise:

- If this bit is set, a 1 byte argument is included in this packet following the network control field bytes. This value represents the transmission power level the tag should use for all its broadcasts. This value is mapped to a power level according to the table below.

Value	TX Power
0x07	0 dBm
0x06	-1 dBm
0x05	-3 dBm
0x04	-5 dBm
0x03	-7 dBm
0x02	-10 dBm
0x01	-15 dBm
0x00	-25 dBm

If this bit is clear, the tag will return its current 1 byte TX Power level in its acknowledgement. This variable will be read from or set for the specific user mode given in the Command Header.

3.3.7 Enabling/Disabling the Reporting of Tag TX Power

Bit 2 Enable/Disable TX Power Report Bit

- If TX Power Reporting Mask Bit is 0, this bit is ignored.

Otherwise:

- If this bit is set, the tag will enable reporting of its 1 byte TX Power in every beacon packet transmission.
- If this bit is clear, the tag will disable further reporting of its 1 byte TX Power in every beacon packet transmission.

The reporting will be enabled or disabled for the specific user mode given in the Command Header.

3.3.8 Setting the Tag ID

Bit 1 Set Tag ID Action Bit

The Tag ID is defined as 2 bytes in order to conform to the IEEE 802.15.4 MAC protocol. However, it is recognized that in some applications, a longer ID field is needed. In such cases the tag User Memory field can be used.

Since the ID is transmitted in every packet, the length of this field is limited to 2 bytes in order to minimize the packet length. However, for those applications which require a longer ID (for example a 96-bit EPC ID), the TagSense Version 3 protocol has also provided a user memory field that can be enabled to extend this ID field. Please see the User Memory Field section of this document for further information.

The Tag ID bit is used as follows:

- If Tag ID Mask Bit 0 is 0, this bit is ignored. Otherwise:

- If this bit is set, a 2 byte argument is included in this packet following the network control field bytes. The tag will set its ID to this value.

- If this bit is clear, the tag will return a general acknowledgement, which will contain its ID.

Note that there is no corresponding Enable/Disable Tag ID Report Bit. This is because the tag ID is always included in every packet transmission.

Bit 0 Network Control Field Extension Bit

If this bit is set, there is a second pair of network control field bytes that follows immediately after this one. This is provided to allow for future expansion of the Tagsense protocol.

3.4 GPIO Field

The (General Purpose I/O) GPIO field indicates which GPIO commands are present in this packet and is used to configure the I/O pins of the tag and also to specify which i/o data is to be included (reported) in the tag beacon packet and which i/o pins are to be sampled and included in the data logging memory. The presence of this field is indicated by setting the appropriate bit in the Packet Control Header.

3.4.1 Types of i/o Pins

The input and output pins provided on the ZT-50 are not all identical. This is a consequence of the specific microcontroller chip employed in the ZT-50, and is not due to the protocol design. While some pins can be both analog or digital, some pins can only be analog, and some can only be digital. There are also other special pins that support frequency counting.

The i/o configuration of pins 1-4 can be described as follows:

- A pin can be set to be ANALOG or DIGITAL
- If a pin is set to DIGITAL, then the pin can be an OUTPUT or INPUT
- If the pin is set to be an OUTPUT, then the value of the pin is set (to 1 or 0)
- If the pin is set to be ANALOG, then analog reference voltage must be specified for the pin (1.1V or battery voltage)
- A pin can also be disabled, thus internally disconnecting it from other internal modules on the microcontroller.

Since each pin can have many configurations, the GPIO field makes use of several bytes to specify all the possible combinations. The possible configurations are shown in the figure below:

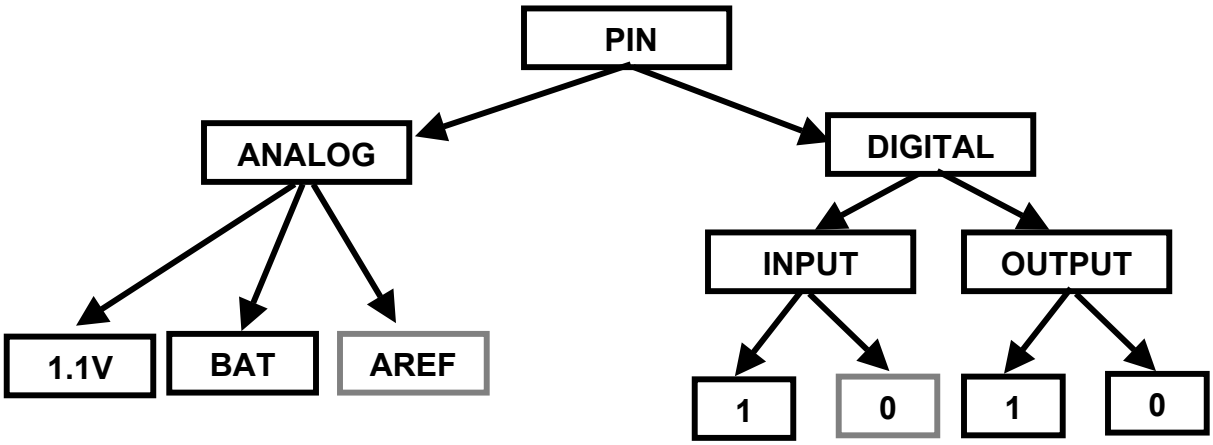


Fig 6. Possible state for a GPIO pin. The 0 state under the INPUT state is used to disable the pin.

Pins 5-8 do not support Analog input, but instead function as frequency counting inputs. More specifically, pins 5 & 6 count higher frequencies, and pin 7 counts lower frequencies (measures time between edges). At present, pin 8 is digital only and does not support any analog or frequency counting functions.

The i/o configuration of pins 5-7 can be described as follows:

- A pin can be set to be FREQUENCY COUNT or DIGITAL
- If a pin is set to DIGITAL, then the pin can be an OUTPUT or INPUT
- If the pin is set to be an OUTPUT, then the value of the pin is set (to 1 or 0)
- If the pin is set to be FREQ CONT, then frequency range (timer prescaler) must be specified for the pin (low frequency or high frequency range)
- A pin can also be disabled, thus internally disconnecting it from other internal modules on the microcontroller.

Since each pin can have many configurations, the GPIO field makes use of several bytes to specify all the possible combinations. The possible configurations are shown in the figure below:

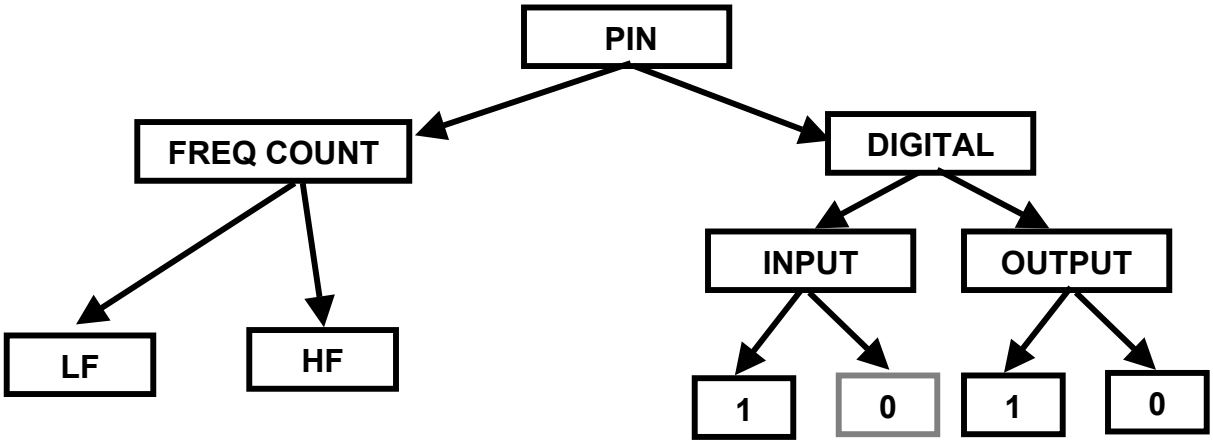


Fig 7. Possible state for a GPIO pin. The 0 state under the INPUT state is used to disable the pin.

3.4.2 Hardware Dependencies

While TagSense has made a strong effort to create a communication protocol that is independent of the hardware implementation, of the ZT-50 tag, this hardware abstraction layer comes a price of increased programming complexity and firmware code space on the tag hardware. Since there is always a strong desire to minimize the cost of the tag, we have made what we think is a reasonable trade-off between complexity and cost.

Over the years, TagSense has made a variety of tag versions with the number or sensor input ranging from 1 to 8, and handling both analog and digital data.

The current version of the ZT-50 hardware has the following Pin restrictions:

GPIO PINS		
PIN	FUNCTIONS SUPPORTED	NOTES
1	Analog, Digital	
2	Analog, Digital	
3	Analog only	
4	Analog only	
5	Frequency (T0), Digital	HF, 8-bit counter
6	Frequency (T1), Digital	HF, 16-bit counter
7	Frequency, Digital	LF
8	Digital only	

Table 8. Possible state for current hardware configuration (v6.x).

3.4.3 Procedure for Setting the Configuration of each I/O pin

In order to set the configuration of one or more I/O pins, TagSense has adopted the "mask and action" approach for sending a command. This approach is commonly employed in other RFID protocols.

In order to configure one or more I/O pins, the specific pins must be selected through the use of a "Mask Byte." Then the configuration of these pins is then specified using several subsequent "Action Bytes."

Using this approach, it is possible to efficiently configure multiple I/O pins at the same time within the same command packet.

3.4.4 Structure of GPIO Field for Command Packets

The structure of the GPIO field is summarized in the table below. As can be seen from the table, the first byte in the field selects which pins are to be configured and the following bytes specify the configuration of the pin(s).

Field	Remarks
GPIO Mask Byte	1-byte value that selects which pins are to be configured in this command packet. All other pins are left unchanged.
A/D Config Byte	This byte specifies which pins should be set to ANALOG/FREQ CONT and which pins set to DIGITAL
I/O Config Byte	This byte specifies which pins should be OUTPUTS and which pins should be INPUTS
GPIO Parameter Configuration Byte	• For digital pins, this byte specifies the value of the digital OUTPUT pins • For ANALOG pins, this byte specifies the analog reference (1.1V or 3V) to be used. • For FREQ COUNTING pins, this byte specifies the frequency range (LF or HF) to be used. • This byte can also be used to DISABLE a pin by setting it to be a digital INPUT and setting this bit field to 0.
GPIO Reporting Byte	This byte specifies which GPIO data is reported in a standard tag beacon packet.
GPIO Data Logging Byte	This byte specifies which GPIO data is recorded when data logging is enabled.

Table 4. Structure of GPIO Field in a command packet

GPIO Mask Byte

This byte specifies which pins are to be configured in this command packet. For every pin that is to be configured, the corresponding bit in the Mask Byte should be set (=1). The structure of this byte is given below. Note that the pin numbering starts with 0.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pin7	Pin6	Pin5	Pin4	Pin3	Pin2	Pin1	Pin0

Table 4. Structure of GPIO Mask Byte

GPIO A/D Configuration Byte

Each bit in this byte specifies whether the corresponding i/o pin is set to be digital or analog. In this case, a bit =1 represents an analog (or frequency counting) input, and a bit=0 represents a digital input. Any pins that are set to be an output will also be represented as bit=0.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
A7	A6	A5	A4	A3	A2	A1	A0

Table 5. Format of A/D Config Byte

GPIO I/O Configuration Byte

This byte specifies the configuration of the I/O pins. If a given I/O pin is to be an OUTPUT, the value of the corresponding bit should be set to 1; if the pin is to be an INPUT, then the corresponding bit should be cleared (set to 0). Pins that are set to be analog are reported as inputs by default. The pin designators are assigned as shown in the table below.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
A7	A6	A5	A4	A3	A2	A1	A0

Table 6. Format of I/O Config Byte

GPIO Digital Configuration/Pin Disable Byte

This byte has two functions and its interpretation is dependent on the previous two bytes.

- For each GPIO pin, there are 3 possible configurations: ANALOG INPUT, DIGITAL INPUT, and DIGITAL OUTPUT. These are specified by the previous bytes in the GPIO field. There is no such thing as an ANALOG OUTPUT.
- For ANALOG INPUT pins, this byte is used to specify the ANALOG REFERENCE to be used for the pins that are set to be ANALOG (inputs). The individual bits set here are only

meaningful if these pins are also set to ANALOG in the A/D Configuration Byte. If the pin is set to use the BATTERY (3V nominal) as a reference, then the corresponding bit should be set to 1; if the specific pin is to use the 1.1V internal reference, then the value for the corresponding bit should be set to 0.

- For pins that have been set to DIGITAL OUTPUT pins, the corresponding bits in this byte specify the value of the digital output (HIGH or LOW).
- For pins that are set to DIGITAL INPUT pins, this byte is also used to provide a way to disable one or more GPIO pins. To disable a specific pin, it should be set to DIGITAL INPUT and the corresponding bit in the GPIO Digital Configuration Byte should be set to 0. If the bit is set to 1, then the pin will function normally (as either an analog or digital input). The purpose of disabling a pin is to internally disconnect the pin from any digital or analog circuitry; this may reduce (improve) power consumption of the tag and also provide some protection and isolation from the unused header pins on the interface connector of the tag.

The format of this byte is given in the table below.

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
D7	D6	D5	D4	D3	D2	D1	D0

Table 10. Format of Digital Report Byte

Important Note for Analog Reference: The 3V reference is actually the battery voltage and is subject to change over time as the battery discharges. It should also be noted that TagSense uses a voltage divider and the fixed 1.1V reference to measure battery level. Even though the 3V reference will droop as the battery discharges, any analog reading that use this reference can be corrected by using the battery value reading as well.

GPIO Reporting Configuration Byte

This byte is used to specify which data is to be reported in the tag Beacon Packet. Simply activating a particular analog input channel or digital input will not cause the data from these pins to be automatically reported in the tags' Beacon Packet. The user must explicitly turn on

reporting for the GPIO pins. Therefore, this Reporting Configuration Byte is used to specify which GPIO pin data (if any) is to be reported. The reporting format is discussed in the GPIO field description of the Tag to Host communication section of this datasheet.

Note: Analog and Digital inputs are treated differently:

- To enable the reporting of any of the ANALOG input pins, the corresponding bits in the GPIO Reporting configuration byte must be set (=1). Since the analog readings are 10-bit, 2 bytes are used to report each analog reading in the tag Beacon Packet.
- In order to enable the reporting of any DIGITAL input pins, the corresponding bits in the GPIO Reporting configuration byte must be set (=1). However, since all digital input pins can be reported by a single byte, there will be only one byte reported in the tag Beacon Packet for any digital inputs that are active. In other words, if the reporting is turned on for 1 or more DIGITAL inputs, then the tag will report a single byte which includes the value of all of the digital input pins. If there are no DIGITAL INPUTS or if the reporting is not turned on for any of the DIGITAL inputs, then the digital reporting byte will not be included in the tag Beacon Packet. The format of the Beacon Packet is described elsewhere in this data sheet.

The pin designators are assigned as show in the table below.

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
A7	A6	A5	A4	A3	A2	A1	A0

Table 11. Format of GPIO Reporting Config Byte

GPIO Data Logging Configuration Byte

This byte is used to specify which data is to be recorded in the tag's internal memory when data logging is enabled. Regardless of which GPIO pins are active, the user must explicitly turn on the data logging for these GPIO pins. Therefore, this Data Logging Configuration Byte is used to specify which GPIO pin data (if any) is to be sampled and recorded. The data logging format is discussed in the Data Logging field description of the Tag to Host communication section of this datasheet.

Note: Analog and Digital inputs are treated differently:

- To enable the recording of any of the ANALOG input data, the corresponding bits in the GPIO Data Logging configuration byte must be set (=1). Since the analog readings are 10-bit, 2 bytes are used to record each analog reading in the tag Beacon Packet.
- In order to enable the recording of any DIGITAL input pins, the corresponding bits in the GPIO Data Logging byte must be set (=1). However, since all digital input pins can be represented by a single byte, there will be only one byte recorded in the tag Beacon Packet for any digital inputs that are active. In other words, if the reporting is turned on for 1 or more DIGITAL inputs, then the tag will record a single byte which includes the value of all of the digital input pins. If there are no DIGITAL INPUTS or if the reporting is not turned on for any of the DIGITAL inputs, then the digital reporting byte will not be included in the tag Beacon Packet. If both digital inputs and analog inputs are active, then the tag will record both a digital data byte plus several analog data bytes, as needed. The format of the Data Logging records is described elsewhere in this data sheet.

The pin designators are assigned as show in the table below.

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
A7	A6	A5	A4	A3	A2	A1	A0

Table 11. Format of GPIO Data Logging Configuration Byte

3.4.5 Example of GPIO Field Command Packet

For the purpose of illustration, the following example is given:

Assume the following:

- We want to set I/O Pin 0 and I/O Pin 1 to be an analog inputs
- We want to set Analog channel 0 to use 1.1 volt reference
- We want to set Analog channel 1 to use the 3V reference

- We want to set I/O Pin 2 to be a digital output and we want to set it output HIGH (=1).
- We want to leave all other I/O pins unchanged.
- We want to turn on the reporting for i/o Pin 0, Pin 1, and Pin 3.
- However, for data logging, we only want to record data from Pin 0.

The following is a sample GPIO field that would be included in the command packet to set this configuration:

GPIO FIELD for Example #1			
Sub-Field	**Length**	**Value**	**Remarks**
GPIO Field Mask Byte	1 Byte	0x03	This indicates that we want to change pins 0, 1, and 2. All other pins will be left unchanged.
Analog/Digital Config Byte	1 Byte	0x03	This indicates that Pin 0 and Pin 1 are set to Analog, and Pin 3 is set to Digital. The value of the other bits is irrelevant.
I/O Config Byte	1 Byte	0x04	This indicates that only pin 3 is set to be an OUTPUT. Pins 0 and 1 are set to INPUTS. All other bits are irrelevant.
GPIO Parameter Config Byte	1 Byte	0x06	This indicates that PIN 2 should be set to HIGH (=1), and that PIN 1 is set to use 3V reference and PIN 0 is set to use the 1.1V reference.
GPIO Reporting Byte	1 Byte	0x07	This indicates that data from Pin 0 (analog), Pin 1 (analog) and Pin 2 (digital) is to be reported in the beacon packet.
Analog Reference Byte	1 Byte	0x01	This indicates that only the data from Pin 1 (analog) is to be included for data logging.

Thus the full command packet would appear as:
550C0284CCCC0040030304060701

This can be parsed as follows:

55 = start byte
0C = packet length (the number of bytes after this byte)
02 = packet identifier
84 = command handle
CCCC = Tag ID
00 = Mode number (this GPIO setting is applied to mode 00)
40 = packet control header (indicating that GPIO field is present)
03 = GPIO Mask Byte
03 = GPIO A/D Config Byte
04 = GPIO I/O Config Byte
06 = GPIO Parameter Config Byte
07 = GPIO Reporting Byte
01 = GPIO Data Logging Byte

3.5 Alarm and RTC Field

This field provides a means to set/query the alarms on a tag and also to set or query the real-time clock on the tag. An alarm is defined as a preprogrammed mode that is triggered by the internal clock. Thus the tag will enter the preprogrammed state at the specified time.

Each tag has up to 7 programmable alarms; each alarm has a corresponding state, as well as timing parameters.

By default, the ZT-50 tag contains 2 pre-programmed modes: Mode 0, which is the default mode, and Mode 8 which is Data Dump mode. Up to seven additional modes can be specified and programmed by the user (Modes 1-7).

In order to set a particular mode to be activated at a specific time, then the Alarm Field must be set. If the Alarm Field is not set, then the specified mode will not be triggered by time but will instead be triggered by another means.

Examples of parameters in this group are setting start times and duration times for various alarms that can ultimately trigger a change in mode, as well as setting the current time.

For example, Alarm 1 can be set to have a start time of 8 A.M., and a time duration of 20 minutes. Additionally, the state corresponding to alarm 1 could have its transmission interval to 1 second, while the default state has its transmission interval set to 1 hour. When 8 A.M. arrives, alarm 1 will fire, and the transmission interval will change from once an hour to once a second. At 8:20, the alarm will expire, and the default state will be entered again, with the transmission interval returning to 1 hour.

It should be noted that the Alarm Field is designed around a 24 hour clock. The alarm is intended not to be a one-time event, but rather to be activated every 24 hours.

Alarms can be individually enabled and disabled, and operate on a 24 hour cycle (e.g. the alarm from the previous example will fire at 8 A.M. every day as long as it is enabled). The Alarm field can be used to add new alarms, change an existing alarm, or delete alarms already created.

For future development, we might create way to enable the use to choose between a relative time setting (relative to midnight and repeats daily) or an absolute time stamp that occurs only once.

3.5.1 Alarm Field Format

The Alarm Field consists of a Start Time which is a 4-byte value *relative to midnight*, and a Duration, which is also 4-bytes. The 4-byte time values are integers and are in units of seconds.

Note: Since there are only 86400 seconds in a day, it is possible to specify a relative time (relative to midnight, for example) using only 3 bytes. However, because we also want to support absolute time records for the purpose of setting alarms, we have decided to allocate 4-bytes for representing time stamps in all cases.

Upon expiration of the duration time, the tag will return to the mode 0, which is the default mode, or if other modes have been activated, will return to the mode that is next on the priority list [see next sub-section titled, "Alarm Priority"].

If we wish for the tag to remain in the new mode indefinitely, then the duration can be set to infinite by entering FFFF as the duration. In this case, the only way to then switch out of this mode would be if another alarm goes off to cause it to switch to another mode, or if the user puts the tag in a manual control mode and manually forces the tag to enter another mode.

The bit format for the alarm field is given in the table below.

Alarm Field Control Byte 1	Bit	Significance
	7	If set (bit = 1), then the alarm time and duration will be included (reported) in the next beacon packet that is transmitted by the tag. The alarm time and duration to be reported should correspond to the mode specified by bits 0:2 of this byte. If cleared (bit =0) then this bit is ignored and no alarm information will be included in the tag beacon packet.
	6	If set, the four-byte alarm start time is included in this packet. This start time shall be assigned to the mode given by bits 3:0 of this byte. If bits 3:0 are set to

	the default state (0), this is used to set the current real-time clock value of the tag. 5 If set, the four-byte alarm duration is included in this packet. This duration value shall be assigned to the mode given by bits 3:0 of this byte. If bits 3:0 are set to the default state (0), the tag will ignore this bit field. 4 If set, the alarm specified by bits 3:0 of this byte is enabled (this turns ON the alarm); if this bit is cleared (set to zero), then the corresponding alarm is disabled (turns OFF). 3 This bit is = 0 by default. If this bit is set to 1, then the alarm will activate the custom command sequence (described in Section 2.5 of this document). If this bit is set, then bits 2:0 are ignored. 2:0 These three bits specify to which state the various data fields apply. If these bits are all set to 0, then this specifies the default state.
Alarm Start Time Byte 1	If Bit 5 of the Alarm Field Control Byte 1 is set, this four-byte value represents the start time of the alarm specified by bits 2:0 of Alarm Field Control Byte 1. Alarms will fire every 24 hours. This start time value is specified as an offset, in seconds, from midnight, and the value must be in the range of 0 to 86399, the number of seconds in a day.
Alarm Start Time Byte 2	
Alarm Start Time Byte 3	
Alarm Start Time Byte 4	
Alarm Duration Byte 1	If Bit 4 of the Alarm Field Control Byte 1 is set, this four-byte value represents the number of seconds that the tag should remain in the state associated

	with the alarm specified by bits 2:0 of Alarm Field Control Byte 1. After this time expires, the tag will return to a lesser priority alarm state (or if no alarms are active, the default state, mode zero).
Alarm Duration Byte 2	
Alarm Duration Byte 3	
Alarm Duration Byte 4	

Bit 6 Enable/Disable Time Report Bit

If Time Mask Bit 0 is 0, this bit is ignored. Otherwise:

If this bit is set, the tag will enable reporting of its 4 byte UTC in every beacon packet transmission. (UTC = universal time code and is the number of seconds elapsed since Jan 1, 2000)

If this bit is clear, the tag will disable further reporting of its 4 byte UTC in every beacon packet transmission.

The reporting will be enabled or disabled for the specific user mode given in the Command Header.

3.5.2 Alarm Priority

Alarms operate on a priority basis. The higher numbered alarms have a higher priority than lower numbered alarms.

In the case that two or more alarms overlap in time, then the alarm with the higher mode number will have priority. For example, if we set Mode #1 to be triggered by an alarm with Start Time = 3pm and Duration = 6 hours; but then we also set Mode #2 to be triggered by an alarm with Start Time = 4pm and Duration = 1 hour, then the following will occur:

- At 3pm, the tag will switch to Mode #1 and annotate this in its memory if data logging is turned ON.

- At 4pm, Mode #2 will be activated because it has a higher Mode number and thus higher priority than Mode #1.
- At 5pm, Mode #2 will expire and thus will return to Mode #1.
- At 6pm, Mode #1 will expire and the tag will return to the default mode (Mode = 0).

As another example, if the tag is in Program Mode 4, and then Alarm 3 fires, the tag will ignore this trigger event and remain in Mode 4. Once the duration for Alarm 4 has expired, if Alarm 3's duration has not expired, the tag will enter Mode 3; otherwise, it will return to the default state (mode 0). Correspondingly, if Alarm 6 fires while in Mode 4, the tag would immediately enter Mode 6.

3.6 Trigger Field

The trigger field provides a way to change the tags state or dynamically modify its behavior. This field provides the user with a means of managing how external events will trigger the changes in the tag's state.

3.6.1 Autonomous operation vs manual operation

The trigger field enables a ser to change tag's behavior based on the value of the sensor inputs. This is known as *autonomous operation*, because the tag will automatically change its behavior based on preprogrammed rules. For example, if temperature sensor is attached and the temperature rises above a certain point, then the tag can be programmed to activate an alarm or sample data more frequently.

If autonomous control is disabled, the trigger field can also be used as a way to force the tag into a different state. This is known as *manual control*. Manual control is commonly used in the situations where the tag is constantly within communications range of the reader; therefore, autonomous triggering is not necessary. In order to force the tag to switch to a different state, the reader sends a command packet which includes the new mode number in the packet.

Autonomous control is enabled by default.

3.6.2 Tag Select Mode

The trigger field is also used to execute the *tag select* command. The design of the tag select command was loosely based on the tag select function that is commonly used with passive RFID tags: the tag

select command provides a means for a reader to put the tag in a special state that will allow it to receive commands more easily. This is particularly important in the case of an asynchronous "tag talks first" protocol which is what is used for the TagSense active RFID tags.

The Select command forces the tag to transmit once per second regardless of its current transmit interval. This enables the reader to send multiple commands in a relatively short time without needing to wait many transmit intervals. This command is also useful for sending commands to a tag among a group of many tags.

3.6.3 Trigger field Parameters

The following is a list of the parameters that can be changed or modified in the trigger field:

- Upper threshold of a specific input pin
- Lower threshold of a specific input pin
- Specify a specific state/mode to enter when a specific sensor goes *above* the upper threshold
- Specify a specific state/mode to enter when a specific sensor goes *below* the lower threshold
- Force the tag to enter a specific state
- Force the tag to enter the "Tag Select" state
- Force the tag to enter the "Data Dump" state

3.6.4 Format of the Trigger Field

When a command is sent to a tag, the trigger field in the command packet begins with 2 bytes that are used to select which parameters are to be set or changed.

The Trigger Field follows the "Mask and Action" command model that is used in other data fields. The basic structure consists of two initial bytes – a *mask byte* and an *action byte* – followed by arguments if needed. In the mask byte, if a particular bit is clear, then the corresponding bit in the action byte is ignored; conversely, if a particular mask bit is set, then the corresponding action bit will be interpreted and the appropriate action taken.

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Upper threshold Mask bit	Lower threshold Mask bit	Upper threshold Target State Mask bit	Lower threshold Target State Mask bit	RFU Currently set to 0	Tag Select Mask Bit	Control Mode Mask Bit	Trigger Control Field Extension Mask Bit

Table 4 shows the structure of the Trigger Control Mask Header

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0=return value 1=set value	0=return value 1=set value	0=return value 1=set value	0=return value 1=set value	RFU - Currently set to 0	Tag Select Bit	Enable/ Disable Manual Control Bit	Trigger Control Field Extension Bit

Table 5 shows the structure of the Trigger Control Action Byte.

The following sections describe how the bits in the Action byte are interpreted for this field:

Bit 7 Upper Threshold Bit

If the Upper Threshold Mask bit is clear (=0), then the corresponding bit in the Action Byte is ignored.

If the Upper Threshold Mask bit is set (=1), then if the corresponding bit is set (=1) in the Action Byte, then the Mask Byte and Action bytes will be followed by a 3-byte argument:

- A single byte specifying the sensor input (0-7) to be monitored.
- Two-bytes specifying the Upper Threshold value (0-1023)

For example, if the 3-byte argument is = &H030130, this means that the tag will monitor sensor input #3, and the upper threshold will be set to &H130 (=304 decimal).

Note: the sensor number must be a value from 0-7. If any large number is entered, the tag will ignore the higher level bits.

Note: the threshold value must be a value from 0-1023. If a number greater than 1023 is entered, the value will be set to 1023.

Note: the factory default value for the lower threshold is 1023.

If the Upper Threshold Mask bit is set (=1), then if the corresponding bit is clear (=0) in the Action Byte, then this will prompt the tag to include a 3-byte value in its acknowledgement packet which indicates the specific sensor value and upper threshold. Essentially, this provides a means for the user to query the tag upper threshold parameters.

Bit 6 Lower Threshold Bit

If the Upper Threshold Mask bit is clear (=0), then the corresponding bit in the Action Byte is ignored.

If the Lower Threshold Mask bit is set (=1), then if the corresponding bit is set (=1) in the Action Byte, then the Mask Byte and Action bytes will be followed by a 3-byte argument:

- A single byte specifying the sensor input (0-7) to be monitored.
- Two-bytes specifying the Lower Threshold value (0-1023)

For example, if the 3-byte argument is = &H030130, this means that the tag will monitor sensor input #3, and the lower threshold will be set to &H130 (=304 decimal).

Note: the sensor number must be a value from 0-7. If any large number is entered, the tag will ignore the higher level bits.

Note: the threshold value must be a value from 0-1023. If a number greater than 1023 is entered, the value will be set to 1023.

Note: the factory default value for the lower threshold is 0.

If the Upper Threshold Mask bit is set (=1), then if the corresponding bit is clear (=0) in the Action Byte, then this will prompt the tag to include a 3-byte value in its acknowledgement packet which indicates the specific sensor value and lower threshold. Essentially, this provides a means for the user to query the tag lower threshold parameters.

Bit 5 Upper Threshold Target State Bit

This bit is used to program which state the tag will switch to when the sensor input exceeds the upper threshold value. This state is known as the *target state*.

If the Upper Threshold Target State Mask bit is clear (=0), then the corresponding bit in the Action Byte is ignored.

If the Upper Threshold Target State Mask bit is set (=1), then if the corresponding bit is set (=1) in the Action Byte, then the Mask Byte and Action bytes will be followed by a 1-byte argument (0-7) specifying which state the tag will enter when the sensor input exceeds the Upper Threshold value.

Note: the tag will revert back to its previous state if the sensor input returns below the Upper Threshold value.

Note: the sensor number must be a value from 0-7. If any large number is entered, the tag will ignore the higher level bits.

If the Upper Threshold Target State Mask bit is set (=1), then if the corresponding bit is clear (=0) in the Action Byte, then this will prompt

the tag to include a 1-byte value in its acknowledgment packet which indicates the upper threshold target state. Essentially, this provides a means for the user to query the tag upper threshold target state.

Bit 4 Lower Threshold Target State Bit

This bit is used to program which state the tag will switch to when the sensor input goes below the lower threshold value. This state is known as the *target state*. **Note:** the target state for the lower threshold can be different from the target state of the upper threshold.

If the Upper Threshold Target State Mask bit is clear (=0), then the corresponding bit in the Action Byte is ignored.

If the Upper Threshold Target State Mask bit is set (=1), then if the corresponding bit is set (=1) in the Action Byte, then the Mask Byte and Action bytes will be followed by a 1-byte argument (0-7) specifying which state the tag will enter when the sensor input exceeds the Upper Threshold value.

Note: the tag will revert back to its previous state if the sensor input goes above the Lower Threshold value.

Note: the sensor number must be a value from 0-7. If any large number is entered, the tag will ignore the higher level bits.

If the Lower Threshold Target State Mask bit is set (=1), then if the corresponding bit is clear (=0) in the Action Byte, then this will prompt the tag to include a 1-byte value in its acknowledgement packet which indicates the target state for the lower threshold. Essentially, this provides a means for the user to query the tag lower threshold target state.

Bit 3 Reserved for Future Use (RFU)

Bit 3 is currently not used.

Bit 2 Tag Select Bit

This bit is used to implement the Tag Select command. This function is used to temporarily change the transmit interval of a tag in order to facilitate the sending of commands.

If the Tag Select Mask Bit is 0, this bit is ignored.

If the Tag Select Mask Bit is set (=1), then this will trigger the tag to enter the *Tag Select Mode*. The Tag Select Action bit is not used, and can be set to 1 or 0.

If the Tag Select Mask bit is set, and the tag enters Select Mode, the tag will immediately but temporarily change its transmit interval to 1 second. However, all other state variables will remain the same. After a specified number of transmissions (determined by the user) the tag will automatically exit Select Mode and revert to its previous state.

Upon entering Select Mode, the amount of time that the tag remains in Select Mode is determined by the transmission interval and the *Select Duration* which is a parameter specified by the user. The Select Duration is defined as the number of packet transmissions that the tag will send before reverting back to its original state. Additionally, the tag will exit the select mode if it receives any new command from the reader.

The Select Duration is specified by one byte argument following the trigger control field bytes that specifies how many transmissions the tag will send before exiting Select Mode.

Bit 1 Enable/Disable Manual Control Bit

This bit enables the user to force the tag to switch into one of the preprogrammed modes. By default, when the ZT-50 tag is reset (power cycled), the ZT-50 is programmed to have Autonomous control.

To *enable* manual control (disable autonomous control) and manually force the tag to switch to a specific state, then this bit must be <u>set</u> in the mask and action bit of the command packet. In this case, mode transitions will only occur if the reader triggers them.

If this bit is set, the mask and header bytes are followed by a 1-byte argument which is used to specify the number of the mode (0-7) which the tag should enter. Valid values for this argument are shown in the table below. Following the execution of this command, the tag will not transition to any other mode automatically – it will remain in the specified mode indefinitely and only transition to a new mode if this command is sent again.

By default, the ZT-50 tag contains 2 pre-programmed modes: Mode 0, which is the default mode, and Mode 8 which is Data Dump mode. Up to seven additional modes can be specified and programmed by the user (Modes 1-7).

This 1 byte argument also provides a means to add and trigger custom command sequences. The list of possible tag modes is listed in the table below. Mode numbers greater than 7 are not accessible in the standard ZT-50 and require custom firmware.

If the Manual Control bit is clear, then autonomous control is *enabled*, and the tag will transition from mode to mode automatically, based on the configurations in the Trigger registers. If the tag already has autonomous control enabled, the tag will ignore this bit.

Value	Mode
0x00	User Programmed Mode 0
0x01	User Programmed Mode 1
0x02	User Programmed Mode 2
0x03	User Programmed Mode 3
0x04	User Programmed Mode 4
0x05	User Programmed Mode 5
0x06	User Programmed Mode 6
0x07	User Programmed Mode 7
0x08	Data Dump Mode (factory programmed)
0x09-0xFE	RFU
0xFF	Execute Custom Sequence

Table 6 shows the possible arguments for the manual control bit.

Bit 0 Protocol Extension Bit

This bit is reserved for future expansion of the protocol. If this bit is set to 1, a separate Mask and Action byte follow this byte. At present, this bit is set to zero in all packets.

3.6.5 Trigger Priority

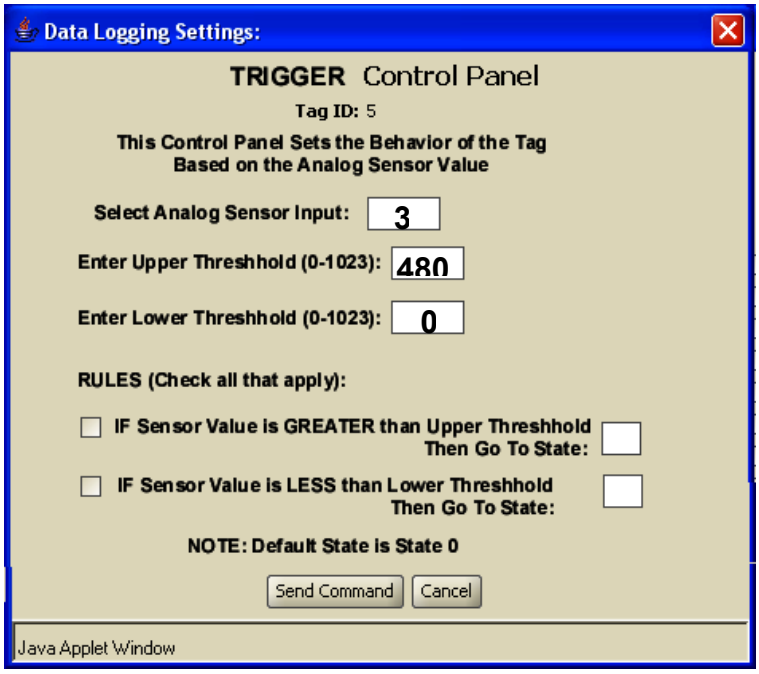
If multiple triggers are defined, it is possible that two or more triggers may be activated simultaneously. In this case, as with multiple alarms, the mode with the higher mode number will have priority.

For example, if there are multiple temperature sensors connected to a tag and we define mode #2 to be activated when Sensor A is above X degrees, and we define mode #3 to be activated when Sensor B is above X degrees, then if both Sensor A and Sensor B are above X degrees, then Mode #3 will be chosen over Mode #2 because Mode #3 has the lower mode number.

3.6.6 Implementation Example

The following is a sample user interface window for setting the Trigger Field.

Example #1:



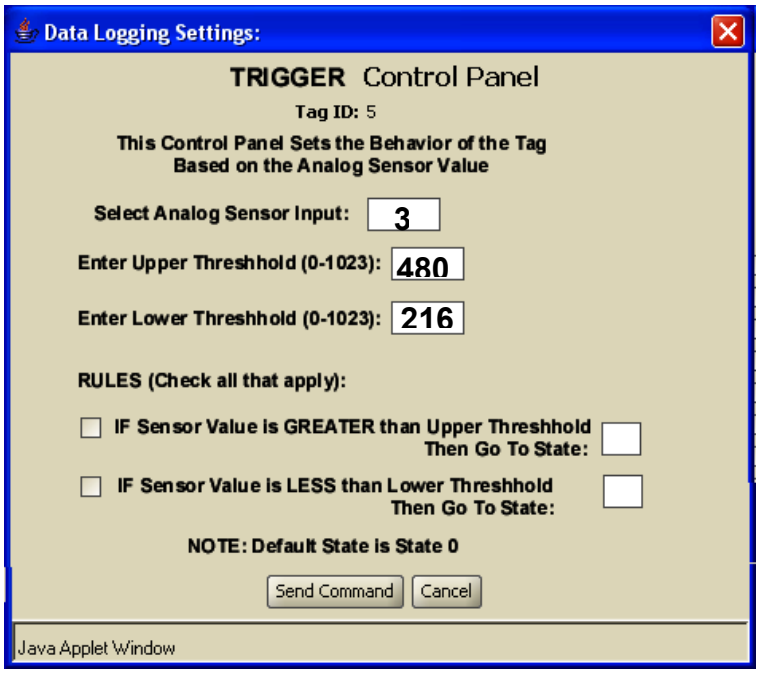
- Analog sensor input = 3
- Upper threshold = 480
- Lower threshold = 0 (default)
- No rules checked

Trigger field:

Mask Byte = &b10000000 = &H80
Action Byte = &b10000000 = &H80
Argument = &H0301E0

➔ Trigger field = &H80800301E0

Example #2:



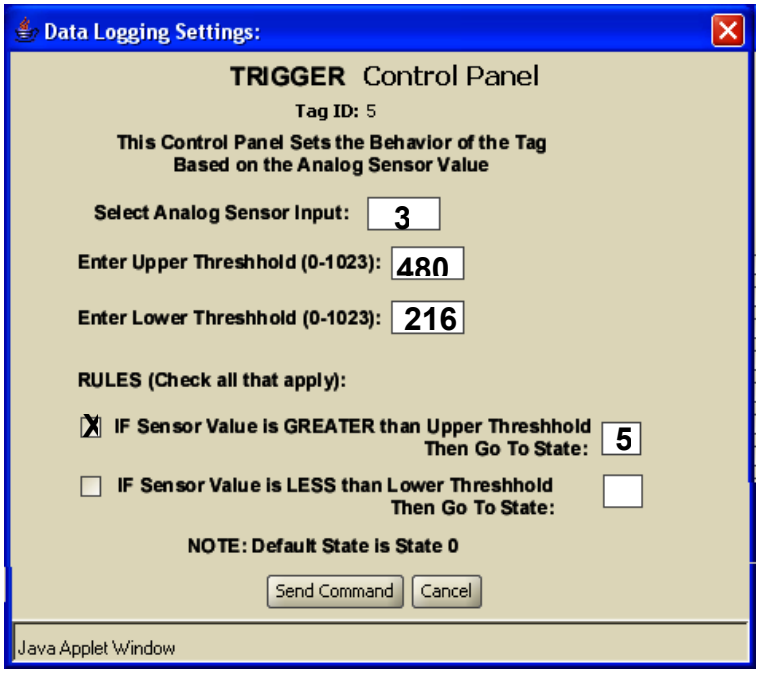
- Analog sensor input = 3
- Upper threshold = 480 (1023 is default)
- Lower threshold = 216
- No rules checked

Trigger field:

Mask Byte = &b11000000 = &HC0
Action Byte = &b11000000 = &HC0
Argument = &H0301E00300D8

➔ Trigger field = &HC0C00301E00300D8

Example #3:



- Analog sensor input = 3
- Upper threshold = 480
- Lower threshold = 216
- Rules: Upper Threshold Target State = 5

Trigger field:

Mask Byte = &b11000000 = &HE0
Action Byte = &b11000000 = &HE0
Argument = &H0301E00300D805

➔ Trigger field = &HC0C00301E00300D805

3.7 User Memory Field

[this section under construction]

In addition to the Data Logger memory (which is large), the ZT-x tags contain a small amount of user memory (256 bytes) which can be used to store industry standard identifier data, such as an EPC (Electronic Product Code), Ubiquitous ID, or EAN barcode number, etc.

The TagSense Version 3 protocol allows the user to read or write data to user memory in units of 2 bytes (16 bits) called memory blocks. This memory block size was chosen to conform to the user memory memory block definitions used in the EPC Global Class 1 Gen 2 tag protocol specification.

Unlike Data Logger memory, the data in user memory can be included in the normal beacon packet that is transmitted every time the tag wakes up. Thus the user memory field can be used as means of extending the ID space of the ZT-x tags. For example, since the Tag ID field is only 2 bytes, for certain applications, it may be desirable to increase the ID field to conform to other numbering standards used in industry such as the EPC standard (which is typically 96 bits = 12 bytes). By enabling the reporting of the appropriate user memory fields, it is possible to configure the tag to report the 96 bit EPC ID in every data packet. 1 Kbyte of user memory. [ideally, we would want 8

The User Memory field indicates which user memory commands are present in this packet. Examples of parameters in this group are writing 16 byte values to various user memory banks.

(see version 1.x document for sample usage of user memory field)

The User Memory Field format is as follows:

Field	Remarks
User Memory Control Field (2 Bytes)	Bit Significance 7 This bit determines whether we are writing to or reading from user memory Bit = 1 memory WRITE Bit = 0 memory READ 6 If bit 7 is set (=1) then this bit is ignored. If bit 7 is clear (=0) then this bit determines whether the user memory is to be included (reported) in the standard beacon packet as follows: Bit 6 = 1 include this user memory field in all subsequent packets Bit 6 = 0 include the user memory in the next packet only. 5 ZT-Link Enable bit: If this bit is set, then the ZT-10 tag is instructed to behave as a ZT-LINK radio module to relay data to an externally connected client device (see ZT-Link Data sheet for more info) 4:0 RFU
User Memory Start Address (2 Bytes)	These bits specify the Starting Address to be used for writing or reading to internal tag user memory. User memory block = 16 bytes (?) Note that this provides for 32767 blocks of 16 bytes each = 4096 bytes (4 KB) total. If the ZT-Link Enable bit is set in the previous byte, then this 2-Byte field is interpreted as the device address. Address x0000 is reserved for the client device and is the default. But if the client device is a controller that is connected to other addressable devices, then the address of these other devices can be specified here.
Data Size	This byte specifies the size (length) of the data to

(1 Byte)	be read or written. The Data Size is in units of memory blocks (1 block = 16 bytes). It is not recommended to read or write more than one or two blocks at a time. So this parameter should be kept very small. In the case of the ZT-Link, this byte specifies the length of the data payload to be sent to the client device that is physically connected to the ZT-Link.
User Data	If bit 7 of the User Memory Control Byte is clear (=0) this data field is not present. If bit 7 of the User Memory Control Byte is set, this field contains the data to be written to the specified memory bank in User Memory. If the ZT-Link function is enabled, then this field contains all the data bytes that are to be transmitted to the client device.

3.8 Data Logging and RTC Field

The Data Logging Field controls all the functions related to data logging and accessing, transferring, and erasing data logger memory. This includes setting the data sampling interval, turning on the memory reporting, or retrieving the data that was stored in the tag data memory.

It should also be noted that the Data Field is used not only to record data from sensors, but it is also used to maintain a history, time record, or log of all the operations performed on the tag. If data logging is enabled, the tag will write a time stamp for every time that the tag entered a particular programmable mode and will also record every time that the tag was powered up (new battery replaced). As a result, the data logging function is thus useful as a debugging tool and as a means of recording tag history, in addition to simply storing sampled sensor data.

TagSense also provides a powerful command set to control the data dumping process to suite a variety of applications and operating conditions.

3.8.1 Data Logging Field Format

The ZT protocol allows the user to modify multiple parameters using a single command. In order to enable this capability, the Data Logging Field contains a Data Logging Control header that is 2 bytes in length. The first byte is a **Mask byte** while the second byte is an **Action byte**. In a command sent by the reader to the tag, the *mask byte* is used to indicate which parameters are being modified and which parameters are being left unmodified. If a parameter is to be modified, its corresponding bit is set to 1; otherwise, the bit should be set to zero (i.e. cleared). For those parameters that are being modified, the *action byte* then indicates how those parameters are to be modified. In other words, the bits in the action byte are processed only if the bit in the corresponding position in the mask byte is set to 1; otherwise the action bit is ignored and the corresponding parameters I left unmodified.

As with all control headers, the LSB position is a field extension bit. If it is set to 1 on both the mask and action bytes, a second pair of mask/action control bytes immediately follows the first pair. If either of the LSB bits are set to 0, then the second pair is not included in the packet. If any arguments are present, they will follow the final pair of mask/action control bytes, in the same order as their corresponding bits in the control bytes, from MSB to LSB.

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Data Sampling Interval Setting Mask Bit	Data Dump Group Size Mask Bit	Data Dump Group Index Mask Bit	Sample on Interrupt /Data Dump Mask Bit	Data Erase Mask Bit	Data Logging Reporting Mask	Data Logging Enable Mask	Data Logging Control Header Extension Mask Bit

Table 10 showing the structure of the Data Logging Control Mask Byte which is part of the Data Logging Header.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Data Sampling Interval Bit	Set/Read Data Dump Group Size	Set/Read Data Dump Group Index	Sample on Interrupt /Data Dump Bit	Erase Logged Data Bit	Enable/ Disable Total Packets Report Bit	Enable/ Disable Data Logging Bit	Data Logging Control Header Extension Bit

Table 11 showing the structure of the Data Logging Action Byte which is part of the Data Logging Header.

The following describes the meaning of the each of the bit fields and parameters in the Data Logging Header Action Byte. As mentioned previously, each of the parameters contained in the *Action Byte* can be either modified or left unchanged as determined by the corresponding bits in the *Mask Byte*.

3.8.2 Setting and Reading Tag Sampling Interval

Bit 7 Set/Read Data Sampling Interval Bit

The sampling interval of the tag can be set or queried by using bit 7 of the Data Logging field.

If the Data Sampling Interval Setting Mask Bit is 0, this bit is ignored, and the Data Sampling interval is left unchanged. (the default sampling interval is 1 second)

Otherwise:

If the Sampling Interval Mask bit is set but the corresponding bit in the Action byte is clear, then the tag will return its one byte Sampling Interval value in its acknowledgement. This is how to *query* the sampling interval.

If the sampling interval bits in both the Mask and the Action are set then this indicates a command to change the sampling interval. In this case a 1 byte argument is to be included in this packet immediately following the Data Logging header control field. The value of this argument represents the time interval that the tag should use for sampling data.

If Data Logging has been turned ON, then the tag will use this sampling interval to periodically sample the value of the input pins and record these values in memory. If Data Logging is OFF, then the sampling interval will be set, however the tag will not record any data until Data Logging is activated.

Also, please remember that the input data that is to be sampled is set by the GPIO field and not by the Data Logging Field. The Data Logging Field simply indicates how often to sample data and also whether or not data logging is enabled.

The sampling interval is represented by the 1 byte argument. The table below shows how this value is converted to an actual time interval in units of seconds.

Value	Interval
0x01	.5s
0x02	1s
0x03	2s
0x04	5s
0x05	10s
0x06	20s
0x07	30s
0x08	1m
0x09	2m
0x0A	5m
0x0B	10m
0x0C	15m
0x0D	30m
0x0E	45m
0x0F to 0x15 Note: 0x16 to FF is not defined	For all parameter values (X) from 0x0F to 0x15, the corresponding time interval (T) in minutes can be calculated as: T = 30*(X-13) Maximum value is 0x15=240min=4 hours

3.8.3 Setting or Reading the Data Dump Group Size

Bit 6 Set/Read Data Dump Group Size

The group size parameter determines how the data logger memory is divided and transmitted from the tag to the reader. The tag does not treat each data record as a separate unit, but rather treats the entire data logger memory as one large data block that can be divided arbitrarily. To optimize data dump speed, the group size can be set very large; however, transmitting a large number of bytes in a single packet has many adverse effects, such as reduced reading range, increased chance of bit errors and also increased chance of collision with other tag transmissions. On the other hand if the group size is set too small, then it will take a longer time to transfer a large data file and also require additional time due to communications overhead. Thus, the optimum group size should be determined empirically, depending on the particular application.

If the Data Dump Group Size Mask Bit is 0, this bit is ignored and the Group size parameter is left unchanged.
Otherwise:

If this bit is set, a one byte argument is included in this packet which indicates what group size the tag should use when a data dump is initiated. The group size is defined as the number of packets sent at once per data dump request. The group size used by the tag will be the value of this argument plus one. For example if the argument is zero, the tag will use a group size of one packet. If the argument is 255, the tag will use a group size of 256 packets.

If this bit is clear, the tag will return its current 1 byte data dump group size in its acknowledgement.

This parameter does not depend on the mode set in the Command Header.

3.8.4 Setting and Reading the Data Dump Group Index

Bit 5 Set/Read Data Dump Group Index

The Group Index provides a way to for the reader to request individual data packets that pertain to the data logger memory. This is useful for several reasons:

- The reader can request the tag to retransmit packets that were lost or not received correctly

- The reader can request packets out of sequence. (in some cases for example, it might be of interest to view only the last packets in the data logger memory)
- If the used data logger memory is very large, a small portion of the memory can be requested instead of downloading the entire memory contents.

The Group Index bit works as follows:
If the Data Dump Group Index Mask Bit is 0, this bit is ignored and the Group Index parameter is left unchanged. Otherwise:

If this bit is set, a 2-byte argument is included in this packet which indicates the group index the tag should use when a data dump is initiated. For example, if the tag has 22 total data dump packets, and the group size is set to 4, this implies that 6 groups of packets are needed to cover all the logged data. The group index can then be used to specify which of the 6 groups are to be transmitted by the tag when a data dump is initiated. The numbering of the groups begins with zero. Therefore, for this example, if the group index is set to zero, then tag the first group (data packets 0-3) will be transmitted. If the group index is set to 5, then tag packets 20-21 will be transmitted. If the user (host) accidentally send the tag an argument that is invalid (e.g. the index references a group packets beyond what the tag has stored), then the command will be ignored.

If this bit is clear, the tag will return its current 2-byte data dump group index in its acknowledgement.

This variable does not depend on the mode set in the Command Header.

3.8.5 Sample on Interrupt

Bit 4 Sample on Interrupt/Data Dump Bit

In some applications, it is desirable to have some way to force the tag to sample data not based on a time interval but based on an external event (e.g. pushbutton or electronic pulse).

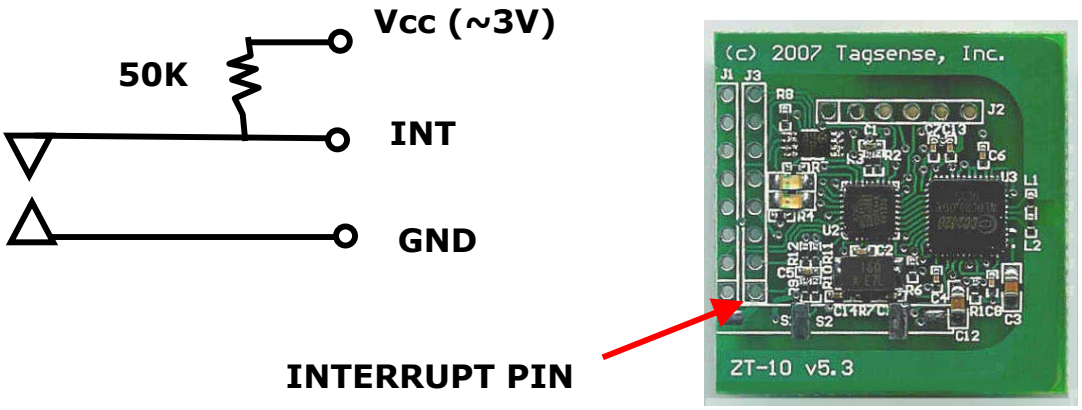
When the tag is not in Data Dump mode, Bit 4 is used to activate the *sample on interrupt* function. When sample on interrupt is activated, the tag will sample data whenever the external interrupt is triggered.

- If the Sample on Interrupt mask bit is 0, this bit is ignored.
- If the Sample on Interrupt mask bit is set (=1), then the Sample on Interrupt function can be enabled/disabled by setting/clearing the Sample on Interrupt action bit.

It should be noted that the sample on interrupt behavior is in addition to the regular sampling that is programmed by the sampling interval, and it provides an alternate way to cause the tag to sample the data. One example might be that we want the tag to sample data when the tag is moved. In this case, we can attach a motion sensor to the external interrupt pin; then when the tag is moved, this sensor will trigger the external interrupt and force the tag to sample data.

A second example might be that if the tag is in motion, we want to sample the data when the tag passes a certain point in space such as a doorway or a point along its path. In this case, we can attach a magnetic reed switch or an Infrared sensor to the tag; then when the tag passes a magnet or crosses the infrared beam (respectively) then the tag will sample data.

A third category of applications is simply using the tag as a means of interfacing to a external circuit. One example might be to create a medical sensor such as a blood oxygen sensor. This external sensor has its own separate circuitry that does the measurements and filters the raw signals and generates an output voltage. This external circuit can then trigger the tag to sample this voltage at the appropriate time in order to store the value in its data memory to be relayed to the remote reader at a later time.



The interrupt pin is normally high (Vcc) and will trigger a data sample when there is a falling edge (pulled down momentarily to GND). If using a pushbutton or magnetic reed switch, these should be connected between the Interrupt pin and GND.

3.8.6 Initiating Data Dump from Tag

Bit 4 Initiate Data Dump Bit

When the tag has been placed in Data Dump mode, bit 4 of the Data Logging field has a special meaning. Instead of controlling the Sample on Interrupt, this bit is used to initiate a "*data dump*" which transmits the contents of the tag data logger memory to the reader. *This command is only enabled if the tag is in Data Dump mode. Otherwise it is ignored.*

- If the Initiate Data Dump Mask Bit is 0, this bit is ignored.
- If the Initiate Data Dump bit is clear, it is also ignored.

Otherwise:

If both the Initiate Data Dump Mask bit and the Initiate Data Dump bit are both set, a data dump will be initiated. The tag will then proceed to transmit either all, or some subset of its records, depending on how many record bytes are in memory, the group size, and the group index. Each packet will contain at most 16 bytes of record information. Packets are sent in groups. The number of packets per group is determined by the group size. For example, if there are 22 packets total, and the group size is 4, the reader will need to send 6 separate commands to initiate a data dump to get all the records. In each of these commands, it should increment the group index. If it misses a data dump packet for any reason, it can retransmit the data dump command using the same group index.

Every group of packets transmitted by the tag thus has a unique number that identifies its location in the overall data logger memory. The host is then responsible for reassembling the entire data logger memory file by concatenating all the transmitted packet groups and stripping off the group packet header information and check sum bytes.

3.8.7 Erasing Tag Data Memory

Bit 3 Erase Logged Data Bit

This bit is used to erase the stored data logger memory data in the tag. However, this bit is will only take effect under the proper conditions. Otherwise, it will be ignored.

The following conditions apply:

- *This command is only enabled if the tag is in Data Dump mode. Otherwise it is ignored.*
- *The erase data command will also be ignored if the user (host) tries to request data in the same command (by setting bit 4).*

If the Data Dump Mask Bit is 0, this bit is ignored. Otherwise:

If this bit is clear, it is ignored.

If both the Erase Data Mask Bit and the Erase data Bit are both set (=1) then, all logged data is erased.

3.8.8 Enabling/Disabling Reporting of Memory Count

Bit 2 Enable/Disable Total Packets Report Bit

It is often useful to know how many data packets are currently stored in the data logger memory. This is useful for knowing when the data memory is almost full and also useful for knowing that the data memory is working properly. Since the reporting of the data memory adds more bytes to the transmitted data packet, it is preferable to turn OFF this reporting when it is not needed, in order to preserve battery life and extend the tag reading range.

- If the Data Logging Reporting Mask Bit is 0, this bit is ignored.

Otherwise:

- If this bit is set, the tag will enable reporting of the total number of packets worth of records the tag is currently holding. The number of packets will be represented by the 2-byte number immediately following the data logger control bytes. Each packet holds 16 bytes of records.
- If this bit is clear, the tag will disable further reporting of the total number of packets worth of records the tag is currently holding.

The reporting will be enabled or disabled for the specific user mode given in the Command Header. Note that this reporting is permanently enabled in the Data Dump mode. [Q: is there a way to query or report the memory count when in data dump mode?]

3.8.9 Enabling/Disabling Data Logging

Bit 1 Enable/Disable Data Logging Bit

If the Data Logging Enable Mask Bit 0 is 0, this bit is ignored. Otherwise:

If this bit is set, data logging is turned on for the mode specified in the Command Header. If this bit is clear, data logging is turned off for the mode specified in the Command Header. Note that data logging is not possible when the tag is in Data Dump mode. The tag must be taken out of data dump mode in order to resume data logging.

Bit 0 Data Logging Control Field Extension Bit

If this bit is set, there is a second pair of data logging control field bytes that follows immediately after this one.

3.8.10 The Use of User Modes/States

Remember that is possible for a tag to have multiple states or modes, each with its own set of state variables.

When a command is sent to change the sampling interval or change a particular parameter, this setting corresponds to the specific State (or mode) that is indicated in the Command Header.

For example, the tag may be in State 0, for example, but we may want to program the sampling interval for State 1. By specifying the state or mode number in the Command Header, it is possible to define a specific sampling interval for State 0 and a different sampling interval for State 1, for example.

3.9 RTLS Field

The RTLS field is used to control the RTLS functionality included in the ZT-50 tag. The RTLS function is based on the use of additional devices known as "Locator Beacons" which are small radio devices that are used to mark specific locations, such as a doorway or gate.

The RTLS functions provide a variety of ways that the ZT-50 tag can make use of the locator beacon signals and include the location ID into its standard datapacket.

RTLS Header control Byte	Bit	Significance
	7:3	RFU
	2	If this bit is set (=1) then the RSSI Variability of the Location Beacon ID (1 byte value) is to be permanently included (reported) in the beacon packet of the tag. If this bit is cleared (=0) then the RSSI Variability of the Location Beacon ID is not included (not reported)
	1	If this bit is set (=1) then the RSSI of the Location Beacon ID (1 byte value) is to be permanently included (reported) in the beacon packet of the tag. If this bit is cleared (=0) then the RSSI of the Location Beacon ID is not included
	0	If this bit is set (=1) then the Location Beacon ID (2 byte value) is to be permanently included (reported) in the beacon packet of the tag.

	If this bit is cleared (=0) then the Location Beacon ID is not included
Location Beacon ID Byte 1	This is the ID of the most recently detected Locator Beacon.
Location Beacon ID Byte 2	
Location Beacon RSSI (1 Byte)	This is the RSSI of the most recently detected Locator Beacon
Location Beacon RSSI Variability (1 Byte)	This is the difference between the last 2 RSSI values from the most recently detected locator beacon

4 Tag to Host Communication

There are various types of data packets that can be transmitted by a tag. This section describes the different types of tag packets that are possible.

At present, the tag will transmit 3 general types of data packets:

- **Beacon packet** – this is the general purpose data and IP packet
- **Acknowledgement packet** – this is a special packet used to acknowledge commands sent by the reader to the tag. In some contexts, this is known as a *response packet*.
- **Data Dump packet** – this is a special data packet used to transfer data-logger memory records from the tag to the reader.

These 3 types of packets are described in the following sections.

4.1 The Beacon Packet

The Beacon packet is the most common type of packet transmitted by the tag. The Beacon packet is a very flexible general-purpose data packet that communicates all the relevant information from the tag to a reader device.

4.1.1 General Packet Structure for Beacon packet

The table below gives the general structure of the tag beacon packet.

Field	Value	Remarks
Tag ID		Two byte source Tag ID
Reader ID		Two byte destination Reader ID
Packet Number		Single byte value that increments with each transmitted beacon packet
Protocol Version & Packet Type	0x30	Single byte value whose upper nibble indicates the version of the protocol, and lower nibble indicates the type of packet. This value is 0x30 for beacon packets in this version of the protocol.
Packet Control Header		Single or multiple byte value that indicates which fields (e.g. Network Field, GPIO Field, etc.) are present in this packet.

Network Field		Multi-byte field that contains network related parameters. This field is only present if so indicated by the *Packet Control Header.*
GPIO Field		Multi-byte field that contains GPIO related parameters. This field is only present if so indicated by the *Packet Control Header.*
Alarm Field		Multi-byte field that contains Alarm related parameters. This field is only present if so indicated by the *Packet Control Header.*
Trigger Field		Multi-byte field that contains Trigger related parameters. This field is only present if so indicated by the *Packet Control Header.*
User Memory Field		Multi-byte field that contains User Memory related parameters. This field is only present if so indicated by the *Packet Control Header.*
Data Logging Field		Multi-byte field that contains Data Logging related parameters. This field is only present if so indicated by the *Packet Control Header.*
RTLS Field		Multi-byte field that contains data related to the Real-Time Location System functions. This field is only present if so indicated by the *Packet Control Header.*

4.1.2 Packet Control Header

The structure of the Packet Control header byte for the tag is the same as the Packet Control header byte used for reader-to-tag communications.

The Packet Control field is a meta header which indicates which fields are present in this packet. If more than one field is present, their order in the overall tag beacon packet will be the same as the order of their respective control bits from MSB to LSB. Table 3 shows the structure of this field.

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Network Field Mask Bit	GPIO Field Mask Bit	Alarm Field Mask Bit	Trigger Field Mask Bit	User Memory Field Mask Bit	Data Logging Field Mask Bit	RTLS Field Mask Bit	Packet Control Header Extension Bit

Bit 7 Network Field Mask Bit

If this bit is set, the Network Field is present; otherwise, it is not present.

Bit 6 GPIO Field Mask Bit

If this bit is set (=1), the GPIO Field is present; otherwise, it is not present.

Bit 5 Alarm Field Mask Bit

If this bit is set, the Alarm Field is present; otherwise, it is not present.

Bit 4 Trigger Field Mask Bit

If this bit is set, the Trigger Field is present; otherwise, it is not present.

Bit 3 User Memory Field Mask Bit

If this bit is set, the User Memory Field is present; otherwise, it is not present.

Bit 2 Data Logging Field Mask Bit

If this bit is set, the Data Logging Field is present; otherwise, it is not present.

4.1.3 Network Field for Beacon Packet

The figure below gives the meaning of the Network Field Control Header.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Current Time Report Bit	Transmit Interval Report Bit	Transmit Power Report Bit	Battery Voltage Report Bit	Current Mode Report Bit	RFU	RFU	Network Field Extension Mask Bit

Bit 7 Current Time Report Bit

If this bit is set to 1, a 4-byte value representing the current time follows this field. If this bit is set to 0, the current time is not reported this beacon packet.

Bit 6 Transmit Interval Report Bit

If this bit is set to 1, a 1-byte value representing the transmit interval *for the mode the tag is currently operating in* follows this field. If this bit is set to 0, the transmit interval is not reported this beacon packet.

Bit 5 Transmit Power Report Bit

If this bit is set to 1, a 1-byte value representing the transmit power *for the mode the tag is currently operating in* follows this field. If this bit is set to 0, the transmit power is not reported this beacon packet.

Bit 4 Battery Voltage Report Bit

If this bit is set to 1, a 1-byte value representing the battery voltage follows this field. If this bit is set to 0, the battery voltage is not reported this beacon packet.

Bit 3 Current Mode Report Bit

Sometimes it is useful to know what mode the tag is in.
If this bit is set to 1, a 1-byte value indicating the current mode number follows this field. If this bit is set to 0, the mode number is not reported this beacon packet.

4.1.4 GPIO Field for Beacon Packet

If the GPIO field bit is set in the Packet Control Byte, then this field will be included in the tag data packet. This field contains information about the I/O pins, including the value of any sensors attached to these pins.

In general, an i/o pin can in one of many configurations. It is useful to provide a means of querying or reporting the specific configuration of the i/o pins on the tag.

The current specification allows a user to query the following information regarding the i/o pins:

- To know if a given pin is set to be ANALOG or DIGITAL
- If the pin is set to DIGITAL, then to know if the pin is set to be an OUTPUT or INPUT
- If the pin is set to be an INPUT, then to know what is the value of that input (1 or 0)
- If the pin is set to be ANALOG, then the pin is assumed to be an input, but it is necessary to know the value of the reading (10-bit digitized value of the input voltage on the pin).
- In addition, it may also be necessary to know which analog reference voltage is being used for the pin (1.1V or battery voltage).

The current specification allows the user to query the value of these settings or simply to report these settings in the beacon packet of the tag.

The figure below gives the meaning of the *GPIO Field Control Header*.

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Analog/Digital Config Report	I/O Config Report	Digital Report	Analog Reference	Analog Report	RFU	RFU	RFU

Table 7. Format of GPIO Field Control Byte

The meaning of each of the bits in the GPIO Field Control Header Byte is described below.

Bit 7 A/D Config Report Bit

If this bit is set to 1, then this indicates that the current tag packet includes an additional byte, immediately following the GPIO Control Header Byte, which reports the which GPIO interface pins are set to be digital and which are set to be analog. In this case, a bit =1 represents an analog input, and a bit=0 represents a digital input. Any pins that are set to be an output will also be represented as bit=0.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
A7	A6	A5	A4	A3	A2	A1	A0

Table 8. Format of A/D Config Byte

Bit 6 I/O Config Report Bit

If this bit is set to 1, then this indicates that the current tag packet also contains an additional 1-byte value representing the configuration of the I/O pins. This byte will be reported after the byte related to Bit 7 (if present). If a given I/O pin is set to be an output, the value of the corresponding bit will be 1; if set to an input, then the value will be 0. Pins that are set to be analog are reported as inputs by default. The pin designators are assigned as shown in the table below.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
A7	A6	A5	A4	A3	A2	A1	A0

Table 9. Format of I/O Config Byte

Bit 5 Digital Report Bit

If this bit is set to 1, then the current tag packet also contains an additional 1 byte value representing the value of the I/O pins. Any pins that are set to be ANALOG shall be reported as bit=0. All pins that are set to be DIGITAL shall be reported, whether or not they are set to be INPUTS or OUTPUTS. These bytes will be reported after any of the other bytes related to Bit 6 described above.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

Table 10. Format of Digital Report Byte

Bit 4 Analog Reference Report Bit

If this bit is set to 1, then this indicates that the current tag packet also contains an additional 1 byte representing the configuration of the Analog Reference used for the Analog pins. This byte will be reported after any of the byte related to Bit 5. The value of this byte is only meaningful if any of the pins are set to ANALOG. If the pin is set to use the BATTERY (3V nominal) as a reference, then the corresponding bit will be 1; if the specific pin is set to use the 1.1V internal reference, then the value for the corresponding bit will be 0. Pins that are set to digital or set to be outputs are reported as 0 by default. The pin designators are assigned as show in the table below.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
A7	A6	A5	A4	A3	A2	A1	A0

Table 11. Format of Analog Reference Config Byte

Bit 3 Analog Data Report Bit

If this bit is set to 1, then the present tag packet also contain one additional configuration byte, call the Analog Data Report Byte, that indicates which analog data channels are being reported in this tag packet. This Byte occurs after any bytes associated with the previous higher order bits (bit7-4). The Analog Data Report Byte contains 8-bit – one for each analog channel. If a given analog data channels is turned ON, then its corresponding bit will be set (=1); otherwise, the corresponding bit will be clear (=0). The actual analog data will be included in the current packet immediately following the Analog Data Control Byte. Each analog input is 10-bits, so two bytes are allotted to each analog value. Therefore, if any analog channels are active (bits set to 1), then the current packet will contain a series of *additional* bytes immediately following this Analog Data Report Byte. The readings for all analog channels will be reported in order from highest to lowest (channel 7 to channel 0). Each reading consists of 2 bytes with the higher-order byte being reported first.

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
A7	A6	A5	A4	A3	A2	A1	A0

Table 12. Format of Analog Data Report Byte

For example, if analog inputs A4 and A5 are turned ON, then the Analog Report Byte will be 0x48, indicating that there will be four additional bytes which contain the 10-bit analog value of inputs A4 and A5, each represented by a 2-byte value. These bytes will be reported after any of the other bytes related to Bits 7, 6, 5 and 4, as described above.

For the purpose of illustration, the following two examples are given:

Example 1:
Assume the following:

- I/O Pin 0 and I/O Pin 1 are set to be an analog inputs
- Analog channel 0 is using 1.1 volt reference
- Analog channel 1 is using a 3V reference
- I/O Pin 2 is set to be a digital output and has been set HIGH (=1).
- All other I/O pins are set to digital inputs, and all of the inputs are grounded (set to zero).

If we turned on all the reporting bytes so that the tag will display all of the fields (which is wasteful but useful for figuring out the i/o configuration of the tag for the first time) the following is a sample series of bytes that would be generated by the tag (in the GPIO sub field):

GPIO FIELD for Example #1			
Sub-Field	**Length**	**Value**	**Remarks**
GPIO Field Control Header	1 Byte	0xF8	This means that this data packet includes the following: Analog/Digital Report, I/O Config Report, Digital Report, Analog Reference Report, and Analog Report.
Analog/Digital Report	1 Byte	0x03	This means that Pin 0 and Pin 1 are set to Analog
I/O Config Report	1 Byte	0x04	This means that only Pin 2 is set to be an OUTPUT. All other pins are INPUTS.
Digital Report	1 Byte	0x04	This means that the value of PIN 2 is set to HIGH (=1). All ANALOG pins get counted as 0, and all other pins are LOW.

Analog Reference	1 Byte	0x02	Bit 0 = 1 because it is using the 1.1 V reference and Bit 2 =1 because its reference is set to 3 V. The other bits are irrelevant, but in this case they are also set to zero because they are not analog inputs
Analog Report	1 Byte	0x03	This means that data for Pin 0 and Pin 1 will be reported
Analog Data	4 Bytes	0x0108 0x03A2	These bytes represent the value of the voltages at Pin 0 and Pin 1. The Analog inputs are 10 bit values. The voltage reference for Pin 0 is 1.1V and the reference for Pin 1 = 3V (battery) Therefore, these value translates to the following voltages: Pin 0: 264/1024 *1.1V= 0.284V Pin1: 930/1024 *3V= 2.724V Note: the 3V reference is the battery voltage, so this will change over time as the battery discharges. The 1.1V reference, however, is fixed.

Example 2:

More commonly, for efficiency, all the reporting fields are turned OFF except the data packet fields which contain actual live data.

Using the same assumptions and same tag I/O pin configuration as in Example #1, it would be more common to encounter a data packet that is much shorter and looks like the following packet below. Because many of the sub fields have been turned OFF, the user must already know the current tag i/o pin configuration.

GPIO FIELD for Example #1			
Sub-Field	**Length**	**Value**	**Remarks**
GPIO Field Control Header	1 Byte	0x08	This means that this data packet includes *only* the Analog Report sub field.
Analog Report	1 Byte	0x03	This means that data for Pin 0 and Pin 1 will be reported
Analog Data	4 Bytes	0x0108 0x03A2	Same as above. These bytes represent the 10-bit value of the voltages at Pin 0 and Pin 1.

4.1.5 Alarm Field for Beacon Packet

If the alarm field bit is set in the Packet Control Byte, then this field will be included in the tag data packet. This field contains information about the alarms.

Alarm Field Control Byte 1	Bit Significance 7 If set (bit = 1), then the alarm time and duration will be included (reported) in this beacon packet. The alarm time and duration to be reported should correspond to the mode specified by bits 0:2 of this byte. If cleared (bit =0) then this bit is ignored and no alarm information will be included in the tag beacon packet. 6 Set to 0 5 Set to 0 4 This bit indicates whether or not a particular alarm is enabled. If this bit =1, then this means that the alarm specified by bits 2:0 of this byte is enabled; If this bit=0, then it is disabled. 3 Set to 0 2:0 These three bits specify to which state the various data fields apply. If these bits are all set to 0, then this specifies the default state. If the tag is currently in the default state then these settings will immediately take effect.
Alarm Start Time Byte 1	If Bit 7 of the Alarm Field Control Byte 1 is set, this four-byte value represents the start time of the alarm specified by bits 2:0 of Alarm Field Control Byte 1. Alarms will fire every 24 hours. This start time value is specified as an offset, in seconds, from midnight, and the value must be in

	the range of 0 to 86399, the number of seconds in a day.
Alarm Start Time Byte 2	
Alarm Start Time Byte 3	
Alarm Start Time Byte 4	
Alarm Duration Byte 1	If Bit 7 of the Alarm Field Control Byte 1 is set, this four-byte value represents the duration (in units of seconds) corresponding to the alarm specified by bits 2:0 of Alarm Field Control Byte 1.
Alarm Duration Byte 2	
Alarm Duration Byte 3	
Alarm Duration Byte 4	

4.1.6 Trigger Field for Beacon Packet

If the Trigger field bit is set in the Packet Control Byte, then this field will be included in the tag data packet. This field contains information about external signals that can be used to control the ZT-50 as well as frequency counting functionality.

At present, the trigger field of the Beacon Packet is generally used to read the frequency counting data. In the future, the Trigger Field may also be used to indicate the configuration of the external interrupt.

The figure below gives the meaning of the *Trigger Field Control Header*.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
External Interrupt Indicator	Frequency Count A	Frequency Count B	Timer A	Timer B	RFU	RFU	RFU

Table 7. Format of Trigger Field Control Byte

The meaning of each of the bits in the Trigger Field Control Header Byte is described below.

Bit 7 External Interrupt Indicator Bit

If this bit is set to 1, then this indicates that the current tag packet includes an additional byte, immediately following the GPIO Control Header Byte, which reports the which GPIO interface pins are set to be

Bit 6 Frequency Counter A Bit

If this bit is set to 1, then this indicates that the current tag packet includes a 2-byte frequency count value for frequency input A (pin 6)

Bit 5 Frequency Counter B Bit

If this bit is set to 1, then this indicates that the current tag packet includes a 2-byte frequency count value for frequency input B (pin 7)

Bit 4 Timer A Bit

If this bit is set to 1, then this indicates that the current tag packet includes a 2-byte frequency count value for frequency input A (pin 6)

Bit 3 Timer B Bit

If this bit is set to 1, then this indicates that the current tag packet includes a 2-byte frequency count value for frequency input B (pin 7)

4.1.7 User Memory Field for Beacon Packet

Field	Remarks
User Memory Control Byte	Bit Significance 7 Set to 0 6 If this bit = 0 then no user memory information is included in this packet. If this bit =1, then the Start Address, Data Length, and User Memory Data filed are included. 5:0 RFU
User Memory Bank Address (1 Byte)	If bit 6 of the memory control byte = 0 then this byte will not be present. Otherwise, this number can range from 0-255 and represents the starting address (user memory block number) for the user memory data that is reported in this packet. Note that there exist 256 blocks of 16 bytes each = 4096 bytes (4 KB) total.
Data Size (1 Byte)	If bit 6 of the memory control byte = 0 then this byte will not be present. Otherwise, this number can range from 0-255 and represents the number of memory blocks that are included in this packet. This number should be small, such as 1-3.
User Memory Data (multiple bytes)	If bit 6 of the memory control byte = 0 then this byte will not be present. Otherwise, this field contains multiple bytes. The number of bytes present is the Data Size * 16.

4.1.8 Data Logging Field for Beacon Packet

In any data packet transmitted from the tag, there is a small field called the "data logging field" which contains information relevant to the data-logging functions of the tag. The user can choose to include this field or not.

Perhaps the most useful information in this field is the reporting of how many data records the tag is currently holding in its memory. This information is used by the reader to monitor how much data the tag is holding and decide when to dump data from the tag or when to erase its memory.

The format of the data-logging field is described below.

Field	Remarks
Data-Logging Control Byte	Bit Significance 7 If this bit = 1, then this indicates that The memory reporting is turned ON. If this bit =0, then memory reporting is turned OFF. 6-1 RFU (Reserved for Future Use) 0 Protocol extension bit If this bit =1, then there will be another byte following this control byte with more parameters. (this is for future use) If this bit =0, then there are no further control bytes.
Data Memory Usage (1 Byte)	If bit 7 =1, then this field will immediately follow the Data-logging control Byte. This field represents the number of data records currently in the tag memory. Note that there exist 256 blocks of 16 bytes each = 4096 bytes (4 KB) total. This will probably be changed to a 2 byte field in order to allow for a larger memory size.

4.1.9 RTLS Field for Beacon Packet

RTLS Header control Byte	Bit Significance 7:3 RFU 2 If this bit is set (=1) then the RSSI Variability of the Location Beacon ID (1 byte value) is included (reported) in this beacon packet. If this bit is cleared (=0) then the RSSI Variability of the Location Beacon ID is not included (not reported). 1 If this bit is set (=1) then the RSSI of the Location Beacon ID (1 byte value) is included (reported) in this beacon packet. If this bit is cleared (=0) then the RSSI of the Location Beacon ID is not included. 0 If this bit is set (=1) then the Location Beacon ID (2 byte value) is included (reported) in this beacon packet. If this bit is cleared (=0) then the Location Beacon ID is not included.
Location Beacon ID Byte 1	This is the ID of the most recently detected Locator Beacon. These bytes are present only if Bit 0 of the RTLS Header Control Byte is set.
Location Beacon ID Byte 2	
Location Beacon RSSI (1 Byte)	This is the RSSI of the most recently detected Locator Beacon. This byte is present only if Bit 1 of the RTLS Header Control Byte is set.
Location Beacon RSSI Variability (1 Byte)	This is the difference between the last 2 RSSI values from the most recently detected locator beacon. This byte is present only if Bit 2 of the RTLS Header Control Byte is set.

4.2 The Response Packet

The version 3 protocol provides the ability for a tag to acknowledge when it has received a command from the reader through its *response packet*. The Response packet is also used to pass several messages or parameters back to the reader, such as error codes or response to reader queries.

It should be clarified that the Response packet is a separate transmission that is composed by the tag, and is different from the simple radio acknowledgement that is mentioned in the IEEE 802.15.4 specification. In the TagSense documentation, the word "acknowledgement" is sometimes used informally to refer to the response packet. However, the word "response" is more technically correct.

The table below gives the general structure of the *Response* packet.

Field	Value	Remarks
Tag ID		Two byte source Tag ID
Reader ID		Two byte destination Reader ID
Packet Number		Single byte value that increments with each transmitted beacon packet (or in the case of an ack packet, corresponds with the last transmitted beacon packet).
Protocol Version & Packet Type	0x31	Single byte value whose upper nibble indicates the version of the protocol, and lower nibble indicates the type of packet. This value is 0x31 for ACK packets in this version of the protocol.
Response Code		Single byte value that indicates whether the command was successfully executed or not. A 0x00 represents a success. A value 0x01-0xFF represents an error code.

4.3 The Data Dump Packet

Upon receiving a *Data Dump* command, the tag transmits its memory record data in the form of data dump packets.

The table below gives the general structure of a data dump packet.

Field	Value	Remarks
Tag ID		Two byte source Tag ID
Reader ID		Two byte destination Reader ID
Packet Number		Single byte value that increments with each transmitted beacon packet (or in the case of a data dump packet, corresponds with the last transmitted beacon packet).
Protocol Version & Packet Type	0x32	Single byte value whose upper nibble indicates the version of the protocol, and lower nibble indicates the type of packet. This value is 0x32 for datadump packets in this version of the protocol.
Data Dump Packet Number		Single byte value that indicates the sequence number of this data dump packet.
Total Number of Data Dump Packets		One byte value that indicates the total number of packets that are associated with this data dump.
Record Block Length		One byte value that indicates how many bytes the record block in this packet is.
Record Block		Multi-byte field that contains up to 16 bytes of record data.

5 Data Records

This section describes the format of the data records that are automatically generated by the ZT-x tag.

5.1 Basic Data Logger Function Overview

The version 3 tag protocol provides the tag the ability to automatically record data in its internal memory by writing data records n specified formats. In addition, the real-time clock of the ZT-x also provides the ability to annotate exactly what time a particular event occurred through the use of a *time stamp*.

In order to support sensor functions, the ZT-50 can generate various types of data records, depending on the type of data that is being sampled (digital or analog signals) and also depending on what triggered the sample to be taken (timer-based sampling or interrupts-based event). The basic sensor data types are the following:

- Analog
- Digital
- Hybrid data
- Interrupt-generated analog data
- Interrupt-generated digital data
- Interrupt-generated hybrid data

In addition, the Zt-50 also generates other special data records that are used to record important events, such as the removal of the battery or the setting of the time. These special records are the following:

- Set time
- Power reset
- Mode change
- Start of recording
- End of recording
- Tag presence

5.2 General Record Format and Time Stamps

The general format for a memory record consists of a header, which identifies the record, the data for the record, and the time stamp.

As described in other sections of this datasheet, the time on the tag is represented as a 4-byte integer which is the number of *half-seconds* elapsed since Jan 1, 2000. This is loosely based on the UTC (universal time) that is used for computers, which is based on the seconds elapsed since Jan 1, 1970. Note that the basic unit of time on the ZT-50 tag is a half second, which means that the fastest packet transmission rate and fastest sampling rate is 2 Hz.

In order to save data logger memory, all *data records* only contain a 2-byte time stamp, which represents the two least significant bytes of the 4 byte time stamp. In order to provide the correct time offset, the tag also periodically generates *hour records*, which are hourly memory markers that are used to reconstruct the actual time. Each hour record contains the full 4-byte UTC time stamp (in units of half-seconds).

The table below gives the general structure of a record:

Field	Value	Remarks
Record Header		A single byte that indicates the type of record and also generally includes the present Mode number of the tag as well.
Record Data	*optional*	Multi-byte value that whose length and meaning depends upon the information in the record header. If this is a mode transition record, this indicates the settings of the GPIO. If this is a data record, this is the actual analog and digital sensor information. If this is an hour record, this is a one byte value representing the battery voltage.
Time stamp		In the case of a data record, this is a 2-byte value representing the two Least Significant Bytes (LSB) of the 4-Byte time stamp, in units of half-seconds. In the case of every other type of record, this is a 4 -Byte value representing the absolute UTC in half-seconds.

Record Header

The Record Header bytes are detailed below.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RFU	Mode 2	Mode 1	Mode 0	Record Type Bit 3	Record Type Bit 2	Record Type Bit 1	Record Type Bit 0

Bits 6:4 **Mode Descriptor**
These three bits indicate in which of the 8 user configurable modes the record was taken.

Bits 3:0 **Record Type**

The table illustrates the meanings for these bits.

Value	Type
0b0000	Analog Data Record
0b0001	Digital Data Record
0b1000	Hybrid Data Record – this indicates that the data record contains both digital and analog data. In this case, the digital data would appear as a single byte, followed by a pairs of bytes for each analog data value.
0b0010	Hour Record
0b0011	Mode Transition Record
0b0100	Start of Recording
0b0101	End of Recording
0b0110	Tag Presence Record (not used by tag but used by ZR-HUB)
0b0111	Power Reset/Change of Battery
0b1001	Real-Time Clock value Reset
0b1010	Interrupt-generated Analog data
0b1011	Interrupt-generated Digital data
0b1100	Interrupt-generated Hybrid data
0b1101	RFU
0b1110	RFU
0b1111	Reserved for custom applications

Record Data

The format of the data records depends on the type of record. The different types of records and the format for each type of record are described in the following sections.

5.3 Start of Recording Record

A Start of Recording record is produced every time the data logging function is activated on the tag. The Start of Recording record contains the GPIO settings to be used for the logged data and also the full 4-byte UTC time stamp indicating when logging was started.

The format for the Start of Recording record is shown on the following page.

<table>
<tr><th>Field</th><th>Value</th><th>Remarks</th></tr>
<tr><td>Record Header Byte</td><td>0b0mmm0100</td><td>This indicates a START RECORDING record.
The mmm indicates the current tag Mode (0-7).</td></tr>
<tr><td>Record Data 1</td><td>A/D Config Byte</td><td>Any bit that is set indicates that the corresponding pin is set to be an analog input. Any bit that is clear (=0) indicates that the corresponding pin is set to be digital.

<table><tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr><tr><td>Pin 8</td><td>Pin 7</td><td>Pin 6</td><td>Pin 5</td><td>Pin 4</td><td>Pin 3</td><td>Pin 2</td><td>Pin 1</td></tr></table></td></tr>
<tr><td>Record Data 2</td><td>I/O Config Byte</td><td>Any bit that is set indicates that the corresponding pin is set to be an output. Any bit that is clear (=0) indicates that the corresponding pin is set to an input. All analog pins are assumed to be inputs, regardless of the bit value set here.

<table><tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr><tr><td>Pin 8</td><td>Pin 7</td><td>Pin 6</td><td>Pin 5</td><td>Pin 4</td><td>Pin 3</td><td>Pin 2</td><td>Pin 1</td></tr></table></td></tr>
<tr><td>Record Data 3</td><td>Parameter Byte</td><td>For each pin that is defined to be a digital OUTPUT, the bit set here defines the output value of the pin.
For each pin that are defined to be digital INPUT, the bit set here is used to indicate if the pin is enabled (bit set) or disabled (bit clear).
For each pin that is defined to be ANALOG, the corresponding bit indicates which voltage reference is used Bit clear = 1.1 Volt reference; bit set = Vcc reference

<table><tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr><tr><td>Pin 8</td><td>Pin 7</td><td>Pin 6</td><td>Pin 5</td><td>Pin 4</td><td>Pin 3</td><td>Pin 2</td><td>Pin 1</td></tr></table></td></tr>
<tr><td>Record Data 4</td><td>Log Indicator Byte</td><td>This byte indicates which data is being logged. Any bit that is set indicates that the corresponding pin is included in the logged data.

<table><tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr><tr><td>Pin 8</td><td>Pin 7</td><td>Pin 6</td><td>Pin 5</td><td>Pin 4</td><td>Pin 3</td><td>Pin 2</td><td>Pin 1</td></tr></table></td></tr>
<tr><td>Time Stamp</td><td></td><td>4-Byte UTC TimeStamp</td></tr>
</table>

5.4 End of Recording Record

Start of Recording records have the following format:

The End of Recording records have the following format:

Field	Value	Remarks
Record Header	0b0mmm0101	This indicates an END RECORDING record
Timestamp		4-Byte time stamp (in half-second units) elapsed since 0 hrs, January 1, 2000.

Note that End of Recording records have no Record Data field.

5.5 Mode Transition Record

As described in a separate section of this datasheet, the ZT-50 tag can support 8 separate states or "modes". Each mode has its own set of state variables consisting of: the GPIO settings, the sampling interval, the transmit interval, the start time, and the duration.

A Mode Transition record is generated whenever the tag changes modes. As mentioned previously, there are many conditions that can cause a tag to enter a new mode (see Alarm Field and Trigger Field sections of this datasheet). Upon entering a new mode, the tag will create a new Mode Transition record which contains the number of the mode (0-7) as well as the value of the state variables for the new mode (GPIO settings).

Note that if a user resets the GPIO settings on a tag while the tag is in that mode, then the tag will generate a Mode Transition record for the same Mode, in order to indicate that the GPIO settings were changed.

The format of the Mode Transition Record is the same as the Start of Recording Record. However, for completeness, the format is included on the following page.

<table>
<tr><th>Field</th><th>Value</th><th>Remarks</th></tr>
<tr><td>Record Header Byte</td><td>0b0mmm0111</td><td>This indicates a MODE TRANSITION record.
The mmm indicates the current tag Mode (0-7).</td></tr>
<tr><td>Record Data 1</td><td>A/D Config Byte</td><td>Any bit that is set indicates that the corresponding pin is set to be an analog input. Any bit that is clear (=0) indicates that the corresponding pin is set to be digital.

<table><tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr><tr><td>Pin 8</td><td>Pin 7</td><td>Pin 6</td><td>Pin 5</td><td>Pin 4</td><td>Pin 3</td><td>Pin 2</td><td>Pin 1</td></tr></table></td></tr>
<tr><td>Record Data 2</td><td>I/O Config Byte</td><td>Any bit that is set indicates that the corresponding pin is set to be an output. Any bit that is clear (=0) indicates that the corresponding pin is set to an input. All analog pins are assumed to be inputs, regardless of the bit value set here.

<table><tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr><tr><td>Pin 8</td><td>Pin 7</td><td>Pin 6</td><td>Pin 5</td><td>Pin 4</td><td>Pin 3</td><td>Pin 2</td><td>Pin 1</td></tr></table></td></tr>
<tr><td>Record Data 3</td><td>Parameter Byte</td><td>For each pin that is defined to be a digital OUTPUT, the bit set here defines the output value of the pin.
For each pin that are defined to be digital INPUT, the bit set here is used to indicate if the pin is enabled (bit set) or disabled (bit clear).
For each pin that is defined to be ANALOG, the corresponding bit indicates which voltage reference is used Bit clear = 1.1 Volt reference; bit set = Vcc reference

<table><tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr><tr><td>Pin 8</td><td>Pin 7</td><td>Pin 6</td><td>Pin 5</td><td>Pin 4</td><td>Pin 3</td><td>Pin 2</td><td>Pin 1</td></tr></table></td></tr>
<tr><td>Record Data 4</td><td>Log Indicator Byte</td><td>This byte indicates which data is being logged. Any bit that is set indicates that the corresponding pin is included in the logged data.

<table><tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr><tr><td>Pin 8</td><td>Pin 7</td><td>Pin 6</td><td>Pin 5</td><td>Pin 4</td><td>Pin 3</td><td>Pin 2</td><td>Pin 1</td></tr></table></td></tr>
<tr><td>Time Stamp</td><td></td><td>4-Byte UTC TimeStamp</td></tr>
</table>

5.6 Analog Data Records

This is generally the most common type of record. The Analog Data record is used when any of the GPIO pins are set to *analog* inputs. For each tag input pin configured to be an analog input, a 10-bit digitized value of the voltage is recorded (0-1023) followed by a 2-byte time-stamp.

There are two types of Analog Data records. The standard Analog Data record is created when the data is sampled according to the preprogrammed sampling interval set by the user. However, as described in the GPIO section of this data sheet, it is also possible to use an external interrupt event to trigger a sample to be taken. In order to distinguish between the 2 types of Analog Data records (timer-based or interrupt-based), a different record type is used. Aside from the record type, the 2 types of analog records are identical in every respect.

Note that only a 2-byte time stamp is used, in order to preserve data memory. These two bytes are the two Least Significant Bytes of the 4-Byte UTC time. However, the Hour record and the Start of Recording record include a full 4-byte time stamp in order to enable the absolute time to be reconstructed for each data sample.

Analog Data records have the following format:

<table>
<tr><th>Field</th><th>Value</th><th>Syntax</th></tr>
<tr><td>Record Header</td><td></td><td>The record header contains the 4-bit Record type as well as the 3-bit Mode number.
For a standard, time based sample, the Record Header is as follows:
<table>
<tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr>
<tr><td>RFU</td><td>Mode 2</td><td>Mode 1</td><td>Mode 0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>
For an interrupt-based sample, the following Record Header is used:
<table>
<tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr>
<tr><td>RFU</td><td>Mode 2</td><td>Mode 1</td><td>Mode 0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr>
</table></td></tr>
<tr><td>Record Data</td><td>Digitized analog values

(2 bytes per sensor input)</td><td>For Analog Data Records, the Record Data field contains a list of analog values. Each value is represented by 2 bytes (0b0000 followed by the 10-bit ADC value).
The sensor values are returned in order of most significant byte first.
<table>
<tr><td>A(i) Byte 1</td><td>A(i) Byte 0</td></tr>
<tr><td>A(j) Byte 1</td><td>A(j) Byte 0</td></tr>
<tr><td>A(k) Byte 1</td><td>A(k) Byte 0</td></tr>
</table>
Etc.
Where i>j>k

This means that the A/D value of higher numbered analog input precede the lower numbered analog input. (e.g. Analog Input 7 appears before Analog Input 4).</td></tr>
<tr><td>Time Stamp</td><td></td><td>2-Byte UTC TimeStamp</td></tr>
</table>

5.7 Digital Data Record

The Digital Data Record consists of the record header, data, and time stamp. The data field consists of a single byte, regardless of how many digital pins are active. As specified by the user, both the digital inputs and digital outputs are recorded.

There are two types of Digital Data records. The standard Digital Data record is created when the data is sampled according to the preprogrammed sampling interval set by the user. However, as described in the GPIO section of this data sheet, it is also possible to use an external interrupt event to trigger a sample to be taken. In order to distinguish between the 2 types of Digital Data records (timer-based or interrupt-based), a different record type is used. Aside from the record type, the 2 types of digital records are identical in every respect.

Digital Data records have the following format (next page):

<table>
<tr><th>Field</th><th>Value</th><th>Syntax</th></tr>
<tr><td>Record Header</td><td></td><td>The record header contains the 4-bit Record type as well as the 3-bit Mode number.
For a standard, time based sample, the Record Header is as follows:
<table><tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr><tr><td>RFU</td><td>Mode 2</td><td>Mode 1</td><td>Mode 0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr></table>
For an interrupt-based sample, the following Record Header is used:
<table><tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr><tr><td>RFU</td><td>Mode 2</td><td>Mode 1</td><td>Mode 0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr></table></td></tr>
<tr><td>Record Data</td><td>Bit Values</td><td>For Digital Data Records, the Record Data field contains a single byte which reflects the value of the digital I/O pins. For any pins that have been defined to be ANALOG, the value of the corresponding bit is indeterminate.
<table><tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr><tr><td>Pin 8</td><td>Pin 7</td><td>Pin 6</td><td>Pin 5</td><td>Pin 4</td><td>Pin 3</td><td>Pin 2</td><td>Pin 1</td></tr></table></td></tr>
<tr><td>Time Stamp</td><td></td><td>2-Byte UTC TimeStamp
This are the 2 Least-Significant Bytes of the 4-Byte UTC time.</td></tr>
</table>

5.8 Hybrid Data Record

As the name implies, the Hybrid Data Record consists of both digital data and analog data. This type of record is used when the tag is programmed to record both analog and digital data.

There are two types of Hybrid Data records. The standard Hybrid Data record is created when the data is sampled according to the preprogrammed sampling interval set by the user. However, as described in the GPIO section of this data sheet, it is also possible to use an external interrupt event to trigger samples to be taken. In order to distinguish between the 2 types of Analog Data records (timer-based or interrupt-based), a different record type is used. Aside from the record type, the 2 types of hybrid records are identical in every respect.

Hybrid Data records have the following format:

<table>
<tr><th>Field</th><th>Value</th><th>Syntax</th></tr>
<tr><td>Record Header</td><td></td><td>This indicates the Mode Transition record and the 3-bit
<table>
<tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr>
<tr><td>RFU</td><td>Mode 2</td><td>Mode 1</td><td>Mode 0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>
Mode number.</td></tr>
<tr><td>Record Data</td><td>Digital Data

Analog Data

(2 bytes per analog input)</td><td>The first data byte represents the digital data, using the same format as the Digital Data Record.
<table>
<tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr>
<tr><td>Pin 8</td><td>Pin 7</td><td>Pin 6</td><td>Pin 5</td><td>Pin 4</td><td>Pin 3</td><td>Pin 2</td><td>Pin 1</td></tr>
</table>
Following the digital data byte, the remaining bytes in the Record Data field contains a list of analog values in the same format as the Analog Data Record. The sensor values are returned in order of most significant byte first.
<table>
<tr><td>A(i) Byte 1</td><td>A(i) Byte 0</td></tr>
<tr><td>A(j) Byte 1</td><td>A(j) Byte 0</td></tr>
<tr><td>A(k) Byte 1</td><td>A(k) Byte 0</td></tr>
</table>
Etc.
Where i>j>k

This means that the A/D value of higher numbered analog input precede the lower numbered analog input. (e.g. Analog Input 7 appears before Analog Input 4).</td></tr>
<tr><td>Time Stamp</td><td></td><td>2-Byte UTC TimeStamp</td></tr>
</table>

5.9 Hour Marker Record

Every hour, on the hour, a record is generated that contains the complete 4-byte timestamp (in units of half-seconds). This record is used to measure and record the passage of time and also to reconstruct the *absolute* time of the data records using the *relative* time stamps.

Note that Hour Records also contain the battery voltage value, and provide a way to monitor the battery level over time.

Hour marker records have the following format:

Field	Value	Remarks
Record Header	0b0mmm0010	This indicates an HOUR record. The mmm bits indicate the present mode number of the tag.
Battery Value	1 Byte	Battery voltage
Timestamp	UTC	4-byte UTC time stamp (in half-seconds)

5.10 Power Reset Record

A power reset record is generated when the power on the tag is reset. An example of this is if the battery is removed and replaced. When the tag power is reset, the tag will immediately create a Power Reset Record. This record is used to annotate the data logger memory so that the removal and replacement of the battery can be detected and recorded.

Field	Value	Remarks
Record Header	0b0mmm0111	This indicates an POWER RESET record. The mmm bits indicate the present mode number of the tag.
Battery Value	1 Byte	Battery voltage
Timestamp		4-byte UTC time stamp (in units of half-seconds)

Note: At present, the current version of firmware does not save the time or the mode of the tag. As a result, when the tag is powered up, it will automatically default to state 0 (Mode 0) and the time will be reset to t=0 (Jan. 1, 2000). However, in future versions of the firmware, this might be added if needed. The reasoning for designing it this way is as follows: It is possible for the tag to save the current time in memory so it can restore its time automatically when the battery is replaced; however, this may not be practical to do since writing to the EEPROM memory consumes more battery power and the time will need to be adjusted anyway since the tag does not know how much time elapsed while the battery was removed. Saving the mode number is also possible but this gets complicated if the UTC time is reset and there are programmed alarms in the memory.

5.11 Tag Presence Record (used only by ZR-HUB)

In applications where the ZR-HUB reader is being used to upload data to a web server, a special record type has been defined in order to simply record the presence of a tag in the web server data log.

This record does not exist in the tag's data memory, but it is only used for the purpose of formatting the data record log on the web server.

This is commonly used in cases where a tag is present but its data memory is empty (nothing has been recorded yet). In this case, the user has the option of instructing the ZR-HUB reader to record the presence of the tag, even though no sensor data was recorded.

The Tag Presence records have the following format:

<table>
<tr><th>Field</th><th>Value</th><th>Remarks</th></tr>
<tr><td>Record Header</td><td></td><td>This indicates the Tag Presence Record and the 3-bit Mode number.
<table>
<tr><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr>
<tr><td>RFU</td><td>Mode 2</td><td>Mode 1</td><td>Mode 0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr>
</table></td></tr>
<tr><td>Tag ID</td><td></td><td>2-Byte Tag ID (bytes are listed in the same order as occurs in the beacon packet)</td></tr>
<tr><td>Time Stamp</td><td></td><td>4-Byte UTC TimeStamp, defined with T=0, set to 0 hrs, January 1st, 2000</td></tr>
</table>

5.12Sample Data Records

The following is a sample data log memory file:

This is a sample response from the tag with ID 0x000F after receiving a data dump command. In this case the data file is small, so the data dump consists of only a single group of packets.

AA0E04000FFFFF5A30843007C58005F1
AA1B04000FFFFF5A32000510040EA2673102C20EA26490030000030EF1
AA1B04000FFFFF5A32010510A26731050EA2673501000800FB010000F5
AA1B04000FFFFF5A3202051000FC010008021D010000021F01000802F1
AA1B04000FFFFF5A320305102001000C022401000802310100 0A023DF1
AA0904000FFFFF5A3100F1

(note that the last byte in each line is a check sum byte and is not part of the data memory)

Extracting the data payload form these packets, we get the following:

040EA2673102C20EA26490030000030EA26731050EA2673501000800
FB01000000FC010008021D010000021F010008022001000C02240100
08023101000A023D

This data file can then be parsed as follows:

040EA26731 -- this is the start of recording record
02C20EA26490 – this is the hour record
030000030EA26731 -- I believe this is the mode transition record
050EA26735 – this is the end of recording record
01000800FB – digital data record
01000000FC – digital data record
010008021D – digital data record
010000021F – digital data record
0100080220 – digital data record
01000C0224 – digital data record
0100080231 – digital data record
01000A023D – digital data record

To expand these records further, TagSense provides a PC program that will translate the UTC time record into Day Month Year and

Second, and will also translate add meaningful labels to the data for annotation.
After expanding these records, we get the following result:

"Friday, October 12, 2007, 5:11:13 PM","Start of Recording",194
"Friday, October 12, 2007, 5:11:17 PM","End of Recording",194
"Saturday, January 01, 2000, 12:04:11 AM","Magazine Opened",194
"Saturday, January 01, 2000, 12:04:12 AM","Magazine Closed",194
"Saturday, January 01, 2000, 12:09:01 AM","Magazine Opened",194
"Saturday, January 01, 2000, 12:09:03 AM","Magazine Closed",194
"Saturday, January 01, 2000, 12:09:04 AM","Magazine Opened",194
"Saturday, January 01, 2000, 12:09:08 AM","Ad 1 Opened",194
"Saturday, January 01, 2000, 12:09:21 AM","Ad 1 Closed",194
"Saturday, January 01, 2000, 12:09:33 AM","Ad 2 Opened",194

6 Data Packet Examples

If you are writing software to parse a raw data packet the following examples are provided for illustration.

The following are sample data packets that conform to the protocol specified in this document.
For the purpose of these examples, the hexadecimal tag ID is assumed to be xCCCC.

Note that these sample packets contain the envelope information that is added by the reader (such as "AA0C"). The general concept is described in Section 2 of this document. The reader protocol specification is described in detail in a separate document.

+DEBUG [>>55080222000C80040600]
+DEBUG [<<AA03022200]
+DEBUG [<<AA0E04000CFFFFDB30843007C58000E2]
+DEBUG [<<AA0904000CFFFFDB3100E0]
+DEBUG [<<AA0C04000CFFFFDC30803007C5E8]
+DEBUG [<<AA0C04000CFFFFDD30803007C5E8]
+DEBUG [<<AA0C04000CFFFFDE30803007C4E9]

+DEBUG [<<AA0C04000CFFFFE530803007C5E5]
+DEBUG [<<AA0C04000CFFFFE630803007C5E8]
+DEBUG [<<AA0C04000CFFFFE730803007C5E9]
+DEBUG [<<AA0C04000CFFFFE830803007C5E9]
+DEBUG [>>55080223000C80040606]
+DEBUG [<<AA03020108]
+DEBUG [>>55080223000C80040606]
+DEBUG [<<AA03022300]
+DEBUG [<<AA0C04000CFFFFEB30803007C5E9]
+DEBUG [<<AA0904000CFFFFEB3100E9]
+DEBUG [<<AA0E04000CFFFFEC30843007C58000E9]
+DEBUG [<<AA0E04000CFFFFED30843007C58000E9]
+DEBUG [<<AA0E04000CFFFFEE30843007C58000E9]
+DEBUG [<<AA0E04000CFFFFEF30843007C58000E9]

6.1 Beacon Packet Example#1

Example #1: this tag already has data-logging turned OFF:

AA0C04000CFFFFE830803007C51003010803A2E9

The information in this packet can be parsed as follows:

Field	Value	Remarks
Start Byte	0xAA	Generated by reader
Packet Length	0x0C	Generated by reader
Packet Type	0x04	Generated by reader (this value indicates this a tag data packet)
TagID	0x000C	
Reader ID	0xFFFF	
Packet Number	0x2B	The packet number is incremented with every transmission
Tag Packet Type	0x30	This indicates a beacon packet
Packet Control Header	0xC0	This value indicates that following fields are present in this packet: the network field, GPIO field.
Network Field Byte	0x03	This value indicates that the transmission power and battery voltage sub-fields are present
Transmit Power	0x07	
Battery Voltage	0xC5	

GPIO Field Control Header	0x10	Only the Analog Report sub field is present.
Analog Report	0x03	This means that data for Pin 1 and Pin 2 will be reported
Analog Data	0x0108 0x03A2	These bytes represent the 10-bit value of the voltages at Pin 1 and Pin 2.
RSSI	0xE9	

6.2 Beacon Packet Example#2

Example #2: memory reporting and data-logging turned are ON:

AA0E04000CFFFF2B30843007C58014E7

The packets fields can be parsed as follows:

Field	Value	Remarks
Start Byte	0xAA	Generated by reader
Packet Length	0x0E	Generated by reader
Packet Type	0x04	Generated by reader (this value indicates this a tag data packet)
TagID	0x000C	
Reader ID	0xFFFF	
Packet Number	0x2B	The packet number is incremented with every transmission
Tag Packet Type	0x30	This indicates a beacon packet
Packet Control Header	0x84	This value indicates that the network field and the trigger field is present
Network Field Byte	0x30	This value indicates that the transmission power and battery voltage are present
Transmit Power	0x07	
Battery Voltage	0xC5	
Trigger Field Byte	0x80	Indicates that the number of memory records is reported

Number of Records	0x14	
RSSI	0xE7	

6.3 Beacon Packet Example#3

Example #3: memory reporting and data-logging turned are ON:

+DEBUG [<<AA0F040BB8FFFF35308010AD100100A702]

AA = start byte
0F = packet length
04 = packet identifier
0BB8 = Tag ID
FFFF = Reader ID
35 = Packet number
30 =
80 = packet control header (supposed to be C0)
10 = network control header saying battery is voltage is reported
AD = Battery voltage
10 = GPIO header saying analog reporting is included
01 = analog 0 channel is turned on
00A7 = A/D value of channel A0
02 = RSSI

6.4 Beacon Packet Example#4

Example #4: memory reporting and data-logging turned are ON:

AA12040BB8FFFF9B308410AB10010098800040FC

AA = start byte
12 = length of packet

04 = packet identifier
0BB8 = Tag ID
FFFF = reader ID
9B = packet counter
30 = packet control header
84 = should be C4
10 = network control byte saying battery voltage in included
AB = battery voltage
10 = GPIO header header byte
01 = analog channel 0 is reported
0098 = A/D value of analog channel 0
80 = data logging field header indicating that datalogging is enabled and memory record count is being reported
0040= number of samples records in memory
FC = RSSI value

7 Command Packet Examples

The following are sample command strings that conform to the protocol specified in this document.

For the purpose of these examples, the hexadecimal tag ID is assumed to be xCCCC.

Note that the command strings given here also include the envelope information field that is used by the reader. This envelope field and reader commands are described in a separate document (the Reader Function API document). The definition of "handle" is also described in the Reader Function API document.

Halt tag
55080285CCCC00808080

Set tag ID command (0x008002021234) to tag id 0xCCCC using handle 0x84

550A0284CCCC008002024444

---

send datadump command (0x00042020) to tag id 0xCCCC using handle 0x84

Group Size?? Index?
Group Size?? Index?

55080285CCCC00041010

---

send enter datadump mode command (0x0010020208) to tag id 0xCCCC
using handle 0x84

55090284CCCC0010020208

---

exit datadump mode command (0x00100200) to tag id 0xCCCC using handle 0x84

55088285CCCC00100200

---

erase logged data command (0x00040808) to tag id 0xCCCC using handle 0x84

55080284CCCC00040808

---

transmit datadump packet using handle 0x84
(using group size = 4, index = 0)

550B0284CCCC00047070040000

---

set TX interval to XX command to tag id 0xCCCC using handle 0x49

55090249FFFF00802020XX

---

send enable TX interval reporting to tag id 0xCCCC using handle 0x48
55080249FFFF00801010

---

send disable TX interval reporting to tag id 0xCCCC using handle 0x48
55080249FFFF00801000

---

send enable TX power reporting to tag id 0xCCCC using handle 0x48
55080249CCCC00800404

---

send disable TX power reporting to tag id 0xCCCC using handle 0x48
55080249CCCC00800400

---

send enable time reporting to tag id 0xCCCC using handle 0x48
55080249CCCC00804040

---

send disable time reporting to tag id 0xCCCC using handle 0x48
55080249CCCC00804000

---

send enable GPIO reporting to tag id 0xCCCC using handle 0x48
55080248CCCC00400404

---

send disable GPIO reporting to tag id 0xCCCC using handle 0x48
55080248CCCC00400400

---

send enable datalogging reporting to tag id 0xCCCC using handle 0x48
55080248CCCC00040404

---

send disable datalogging reporting to tag id 0xCCCC using handle 0x48
55080248CCCC00040400

---

send enable data logging to tag id 0xCCCC using handle 0x20
55080220CCCC00040202

---

send disable data logging to tag id 0xCCCC using handle 0x21
55080284CCCC00040200

Sample Command Strings (continued)

Send tag select command to tag id $$$$ for YY transmissions using handle 0x84

55090284$$$$00100404YY

8 Revision History

03/2007 – 10/2007 v3.0 – 3.5 Redemske original document

11/03/2007 v3.6 Fletcher edited

11/08/2007 v3.8 Fletcher edited records documentation

12/03/2007 v3.9 Fletcher edited beacon data logging field section and added new section (section 6) to provide more examples of data packets and how they can be parsed. Fixed GPIO field in the beacon packet section. Also fixed some mistakes in the Alarm Field bit descriptions.

1/09/2008 v3.10 Fletcher edited GPIO field for beacon packets, added examples using GPIO, and also corrected command packet network field to allow for transmit intervals up to 240 minutes max.

1/12/2008 v3.11 Fletcher added the Tag Presence record

1/19/2008 v3.12 Fletcher added the Sampling Interval setting in Data Logging Field

1/19/2008 v3.13 Fletcher added the Sampling on Interrupt section.

1/28/2008 v3.14 Fletcher changed GPIO reporting section to match byte order convention; GPIO command section was also added.

1/31/2008 v3.15 Fletcher added the Transmit on Interrupt section.

2/09/2008 v3.16 Fletcher added the Reporting and Data Logging Config bytes into the GPIO command packet description.

?? v3.17 Fletcher ??

4/25/2008 v3.18 Fletcher updated the Data Log Record format to reflect the latest generation of firmware.

7/03/2008 v3.19 Fletcher updated the Trigger Field section.

12/27/2009 v3.20 Fletcher updated user memory field to integrate the ZT-Link functionality. Updated RTLS field.

11/15/2009 v3.21 Fletcher updated transmit on interrupt function and sample on interrupt function. Also expanded Table of Contents. Fixed some minor typos and mistakes.