



EZT-560_i

User Communication Reference Manual

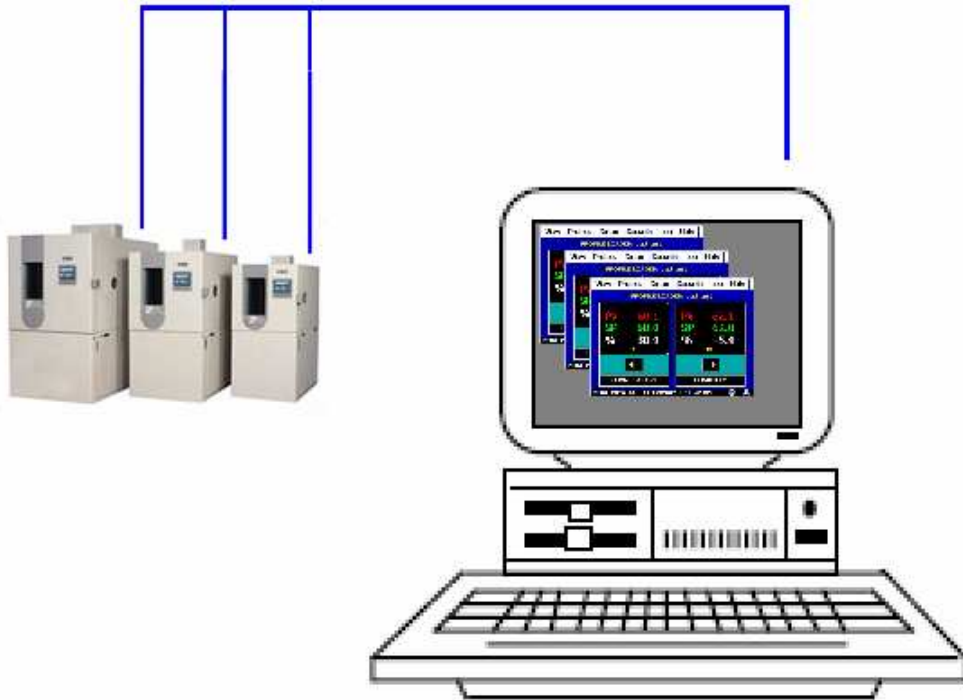


TABLE OF CONTENTS

1. Introduction..... 3
 1.1 Definition of Terms..... 3

2. Serial Communication 5
 2.1 Interface Standards 6
 2.1.1 Interface Converters 7
 2.2 Protocol..... 8
 2.3 Write your own Modbus Application 10
 2.3.1 Packet Syntax..... 12
 2.4 EZT-560i Control Registers 15
 2.5 EZT-560i Profile Registers 35
 2.5.1 Profile Download Algorithm 41
 2.5.2 Sending a Profile to the EZT-560i 43
 2.5.3 Starting a Profile in the EZT-560i 44

Appendix
 Common Terms and Definitions

1. Introduction

This document is targeted towards new users interested in using data communications with CSZ EZT-560i controllers. The purpose of this manual is to enable users to:

1. Understand the basics of data communications via standard definitions, interfaces and protocols.
2. Set up and use a simple network of one or more EZT-560i controller(s).

In this manual, numbers in the format 0x00 represent values in hexadecimal. Numbers in the format 0 represent values in decimal and finally, numbers in the format 00000000 represent values in binary unless otherwise stated.

1.1 Definition of Terms

Machine-to-Machine Communication

In order for machines to communicate with each other, they need a code called a character format or character set. They need rules called protocol to govern their conversation and prevent confusion and errors. Computers need a connecting interface over which to communicate. They may use one pair of wires to send information in one direction and another pair to send in the opposite direction (full duplex). Or they may use one pair to send in both directions (half duplex).

Character Format

The code or character format for the EZT-560i data communication is shared by virtually everyone in the electronics industry. This code defines a computer stream of 1's and 0's, that are created by varying a voltage signal in a regular manner. This code is the American Standard Code for Information Interchange, called ASCII.

Bits and Bytes

The word bit is simply the contraction of the words binary digit. A bit is the basic unit in ASCII. It is either a "1" or a "0". A byte is a string of eight bits that a computer treats as a single character. ASCII can use a single byte to represent each letter of the alphabet, each digit and each punctuation mark we use.

ASCII

The ASCII code defines 128 separate characters, one for each letter, digit and punctuation mark. ASCII also includes control characters similar to those we find on computer keys, such as backspace, shift and return. It also has nine communications control characters for identification, enquiry (inquiry), start of text, end of text, end of transmission, acknowledge, negative acknowledge and escape. The ASCII code is sometimes written in a base 16 number system that is called hexadecimal or "hex" for short. The numbers 0 through 9 represents the first ten digits of this system, and the letters A through F represents the final six digits. The 128 ASCII character codes with the decimal, binary and hexadecimal equivalents are listed in the following table.

ASCII Control Codes

ASCII Control Codes are used to give instructions to the remote device and result in specific actions, such as a line feed instruction on a printer. ASCII Control Codes, the first 33 ASCII characters (non printable), are important for the operation of communicating equipment. They give instruction to remote devices that result in specific actions such as a line feed on a printer. Holding down the keyboard control key while pressing the appropriate keyboard key is what sends these values.

ASCII Character Chart

Char	Code	Decimal	Binary	Hex	Char	Code	Decimal	Binary	Hex
NUL	Ctrl @	0	00000000	00	@	Shift 2	64	01000000	40
SOH	Ctrl A	1	00000001	01	A	Shift A	65	01000001	41
STX	Ctrl B	2	00000010	02	B	Shift B	66	01000010	42
ETX	Ctrl C	3	00000011	03	C	Shift C	67	01000011	43
EOT	Ctrl D	4	00000100	04	D	Shift D	68	01000100	44
ENQ	Ctrl E	5	00000101	05	E	Shift E	69	01000101	45
ACK	Ctrl F	6	00000110	06	F	Shift F	70	01000110	46
BEL	Ctrl G	7	00000111	07	G	Shift G	71	01000111	47
BS	Ctrl H	8	00001000	08	H	Shift H	72	01001000	48
TAB	Ctrl I	9	00001001	09	I	Shift I	73	01001001	49
LF	Ctrl J	10	00001010	0A	J	Shift J	74	01001010	4A
VT	Ctrl K	11	00001011	0B	K	Shift K	75	01001011	4B
FF	Ctrl L	12	00001100	0C	L	Shift L	76	01001100	4C
CR	Ctrl M	13	00001101	0D	M	Shift M	77	01001101	4D
SO	Ctrl N	14	00001110	0E	N	Shift N	78	01001110	4E
SI	Ctrl O	15	00001111	0F	O	Shift O	79	01001111	4F
DLE	Ctrl P	16	00010000	10	P	Shift P	80	01010000	50
DC1	Ctrl Q	17	00010001	11	Q	Shift Q	81	01010001	51
DC2	Ctrl R	18	00010010	12	R	Shift R	82	01010010	52
DC3	Ctrl S	19	00010011	13	S	Shift S	83	01010011	53
DC4	Ctrl T	20	00010100	14	T	Shift T	84	01010100	54
NAK	Ctrl U	21	00010101	15	U	Shift U	85	01010101	55
SYN	Ctrl V	22	00010110	16	V	Shift V	86	01010110	56
ETB	Ctrl W	23	00010111	17	W	Shift W	87	01010111	57
CAN	Ctrl X	24	00011000	18	X	Shift X	88	01011000	58
EM	Ctrl Y	25	00011001	19	Y	Shift Y	89	01011001	59
SUB	Ctrl Z	26	00011010	1A	Z	Shift Z	90	01011010	5A
ESC	Ctrl [27	00011011	1B	[[91	01011011	5B
FS	Ctrl \	28	00011100	1C	\	\	92	01011100	5C
GS	Ctrl]	29	00011101	1D]]	93	01011101	5D
RS	Ctrl ^	30	00011110	1E	^	Shift 6	94	01011110	5E
US	Ctrl _	31	00011111	1F	_	Shift -	95	01011111	5F
SP	SPACE	32	00100000	20	`	`	96	01100000	60
!	Shift 1	33	00100001	21	a	A	97	01100001	61
"	Shift 2	34	00100010	22	b	B	98	01100010	62
#	Shift 3	35	00100011	23	c	C	99	01100011	63
\$	Shift 4	36	00100100	24	d	D	100	01100100	64
%	Shift 5	37	00100101	25	e	E	101	01100101	65
&	Shift 7	38	00100110	26	f	F	102	01100110	66
'	'	39	00100111	27	g	G	103	01100111	67
(Shift 9	40	00101000	28	h	H	104	01101000	68
)	Shift 0	41	00101001	29	i	I	105	01101001	69
*	Shift 8	42	00101010	2A	j	J	106	01101010	6A
+	Shift =	43	00101011	2B	k	K	107	01101011	6B
,	,	44	00101100	2C	l	L	108	01101100	6C
-	-	45	00101101	2D	m	M	109	01101101	6D
.	.	46	00101110	2E	n	N	110	01101110	6E
/	/	47	00101111	2F	o	O	111	01101111	6F
0	0	48	00110000	30	p	P	112	01110000	70
1	1	49	00110001	31	q	Q	113	01110001	71
2	2	50	00110010	32	r	R	114	01110010	72
3	3	51	00110011	33	s	S	115	01110011	73
4	4	52	00110100	34	t	T	116	01110100	74
5	5	53	00110101	35	u	U	117	01110101	75
6	6	54	00110110	36	v	V	118	01110110	76
7	7	55	00110111	37	w	W	119	01110111	77
8	8	56	00111000	38	x	X	120	01111000	78
9	9	57	00111001	39	y	Y	121	01111001	79
:	Shift ;	58	00111010	3A	z	Z	122	01111010	7A
;	;	59	00111011	3B	{	Shift [123	01111011	7B
<	Shift ,	60	00111100	3C		Shift \	124	01111100	7C
=	=	61	00111101	3D	}	Shift]	125	01111101	7D
>	Shift .	62	00111110	3E	~	Shift `	126	01111110	7E
?	Shift /	63	00111111	3F	DEL	Delete	127	01111111	7F

2. Serial Communication

The primary interface CSZ has chosen for the EZT-560i employs serial communication, which is the exchange of data in a one-bit-at-a-time, sequential manner on a single data line or channel. Serial contrasts with parallel communication, which sends several bits of information simultaneously over multiple lines or channels. Not only is serial data communication simpler than parallel, it is also less costly.

Baud Rate

The baud unit is named after Jean Maurice Emile Baudot, who was an officer in the French Telegraph Service. He is credited with devising the first uniform-length 5-bit code for characters of the alphabet in the late 19th century. What baud really refers to is modulation rate or the number of times per second that a line changes state. This is not always the same as bits per second (BPS). However, if you connect two serial devices together using direct cables then baud and BPS are in fact the same. Thus, if you are running at 9600 BPS, then the line is also changing states 9600 times per second.

Typical baud rates for computers are 9600, 19200, 38400 and 57600 baud. As the baud rate increases, so does the transmission rate of data. Thus you get more information in a shorter period of time. However, the faster the transmission rate, the more susceptible it is to error due to the quality of the cable and sources of electrical “noise” in the environment. In order to balance throughput with reliability, CSZ has chosen to use 9600 baud as the data rate for the EZT-560i. *Thus a device used to communicate with the EZT-560i must have its serial port set for 9600 baud in order to for data communications to work properly.*

Start and Stop Bits

The start bit informs the receiving device that a character is coming, and a stop bit tells it that a character is complete. The start bit is always a 0. The stop bit is always a 1. The human speech equivalent of these bits could be a clearing of the throat to get someone’s attention (start bit); and a pause at the end of a phrase (stop bit). Both help the listener understand the message.

A stop bit has a value of 1 - or a mark state - and it can be detected correctly even if the previous data bit also had a value of 1. This is accomplished by the stop bit’s duration. Stop bits can be 1, 1.5, or 2 bit periods in length. CSZ has chosen to use the default – and most common – length of 1 period for the EZT-560i. *A device used to communicate with the EZT-560i must also have its serial port set to use a stop bit of 1 in order for data communications to work properly.*

Parity Bit

Besides the synchronization provided by the use of start and stop bits, an additional bit called a parity bit may optionally be transmitted along with the data. A parity bit affords a small amount of error checking, to help detect data corruption that might occur during transmission. You can choose either even parity, odd parity, mark parity, space parity or none at all. When even or odd parity is being used, the number of marks (logical 1 bits) in each data byte are counted, and a single bit is transmitted following the data bits to indicate whether the number of 1 bits just sent is even or odd.

For example, when even parity is chosen, the parity bit is transmitted with a value of 0 if the number of preceding marks is an even number. For the binary value of 0110 0011 the parity bit would be 0. If even parity were in effect and the binary number 1101 0110 were sent, then the parity bit would be 1. Odd parity is just the opposite, and the parity bit is 0 when the number of mark bits in the preceding word is an odd number. Mark parity means that the parity bit is always set to the mark signal condition and likewise space parity always sends the parity bit in the space signal condition. Since these two parity options serve no useful purpose whatsoever, they are almost never used. *The EZT-560i can be set for even, odd or no parity. A device used to communicate with the EZT-560i must also have its serial port set to use the same parity setting as the EZT-560i in order for data communications to work properly.*

2.1 Interface Standards

An interface is a means for electronic systems to interact. It's a specific kind of electrical wiring configuration. CSZ has selected to use two of the most common serial interfaces used today. This provides both a simple 1 to 1 connection to a PC or PLC using readily available cabling as well as a multi-drop connection where more than one EZT-560i can be placed on the line.

EIA-232 (Full Duplex)

An EIA-232 (formerly RS-232C) interface uses three wires: a single transmit wire; a single receive wire; and a common line. Only two devices can use an EIA-232 interface. A -3 to -24 volt signal indicates a 1 and a +3 to +24 volt signal indicates a 0. The EIA-232 signal is referenced to the common line rather than to a separate wire, as in EIA-485. Thus, an EIA-232 cable is limited to a maximum of 50 feet, due to noise susceptibility.

EIA-485 (Half Duplex)

An EIA-485 interface uses two wires: a T+/R+, a T-/R- line. A -5-volt signal is interpreted as a 1, a +5-volt signal as a 0. As many as 31 remote devices can be connected to a master on a multi-drop network up to 4000 feet long.

Wiring

Most PCs have a standard EIA-232 port (usually referred to as RS-232). In these instances, you must use an interface converter to connect to EIA-485. These interface standards are required to have a multi-drop system (more than one EZT-560i on the link). The following list references some vendors who sell these converters. Should your PC have the appropriate interface, just connect using the wiring shown in the Getting Started section.

For EIA-485, the terminal marked "A" usually connects to the T+/R+ while the "B" terminal connects to the T-/R- of the EZT-560i controller. The standards do not specify the wire size and type. Use of AWG 24 twisted pair provides excellent results. If shielded cable is used, terminate the shield at one end only. Always follow the manufacturer's instructions supplied with the interface converter. See Biasing of Buses next.

Biasing of Buses

The EIA-485 standard requires the bus to be biased for reliable communication. This requires termination resistors to be placed across the T+/R+ and T-/R- wires. One resistor is placed at the PC where it connects to the EIA-485 bus. The second resistor is placed at the last controller on the network. Do not place resistors at each controller. The impedance of the wires used for the bus determines the resistor value. For twisted pair, the value is typically 120 ohms. In addition, it may be necessary to have a pull-up and pull-down resistor between the power supply and ground of the interface adapter.

Check the documentation that came with your interface adapter. Biasing the bus reduces reflection of signals sent down the bus. These reflections are sometimes referred to as a standing wave. This condition is most notable when communicating at high baud rates over longer distances.

2.1.1 Interface Converters

The purpose of an interface converter is to allow two different buses to be connected together. Interface converters are required when connecting an EIA-232 port to an EIA-485 bus. The EIA-485 bus is a half duplex bus. This means that it can only send or receive data at any given time. Some interface converters on the market provide the ability to have full duplex with the EIA-485 bus. This is accomplished by using two receivers and transmitters tied in tandem. This type of converter will not work with the EZT-560i controller. Be sure that the model you purchase is designed for half duplex.

Another consideration when selecting an interface converter is how the converter handles switching between transmit and receive. Typically it is accomplished via a handshake line from the PC. When data flows into the converter from the PC, a handshake line is placed high. When data flows out of the converter to the PC, the handshake line is placed low. In this way, the handshake line controls the direction of information. Another method of achieving this is to use a built-in timer. The converter switches to transmit when a character is sent to it from the PC. After a period of time when the PC has not transmitted, the converter switches to a receive mode.

It is important that you understand how your converter accomplishes this task. You are required to wire this feature or make settings on the converter to enable this function. The PC will not talk to the controller correctly with out properly setting this. Your converter may also require settings through dip switches, to set up communications parameters like baud rate, data bits, start bits, stop bits and handshaking. The converter may also require a separate power supply. Some converters get their power from the handshake lines of the PC. If you rely on this method, you will need to wire these additional lines. In addition, your software must set these lines high. A more reliable method is to use the external power supply. This is especially necessary when using a laptop computer. See the documentation that is provided with your converter for more detail.

Not all converters are equal in performance. If your chamber operates in a harsh, electrically noisy environment, this can cause less robust converters to work intermittently or not at all. CSZ has only tested the converters listed below; however, CSZ makes no claims as to the performance or compatibility of these converters with your PC. These converters are equipped with automatic send data control circuits, driver control in the converter hardware, so you don't have to work with software at all. The circuit monitors data flow and enables the driver during transmission and automatically disables it when no data is being sent. There is no need to rework software or install new drivers.

B&B Electronics
707 Dayton Road
PO Box 1040
Ottawa, IL 61350
Phone 815-433-5100
<http://www.bb-elec.com>

Part # **485OI9TB** for EIA-232 to EIA-85
Part # **485PS2** (external power supply – required if handshake lines unavailable for power)

RESmith
4311 Smith Drive
Hamilton, OH 45011
Phone 513-874-4796
<http://www.RS485.com>

Part # **ASC24T-B9FPS** for EIA-232 to EIA-485 (provided with adapter cables and power supply)

2.2 Protocol

Protocol describes how to initiate an exchange. It also prevents two machines from attempting to send data at the same time. There are a number of different data communications protocols, just as there are different human cultural protocols that vary according to the situation.

The protocol portion of EZT-560i communications is very important, because it provides a quality of communication that others often don't have. Protocol-driven communications are more accurate, because they are less prone to both operator and noise errors. Protocol maintains system integrity by requiring a response to each message. It's like registered mail — you know that your letter has been received because the post office sends you a signed receipt.

In EZT-560i data communications, a dialog will continue successfully as long as the messages are in the correct form and responses are returned to the protocol leader. If the operator enters an incorrect message, or interference comes on to the data line, there will be no response. In that case the master must retransmit the message or go to a recovery procedure. If an operator continues to enter an incorrect message or interference continues on the data line, the system will halt until the problem is resolved. CSZ has selected Modbus RTU as the protocol of choice. Modbus RTU enables a PC to read and write directly to registers containing the EZT-560i's parameters. With it, you can read all 180 of the controller's parameters with three read commands.

Modbus Remote Terminal Unit (RTU)

Gould Modicon, now called AEG Schneider, created this protocol for process control systems called "Modbus". It has the advantage over other protocols of being extremely reliable in exchanging information. This protocol works on the principle of packet exchanges. The packet contains the address of the controller to receive the information, a command field that says what is to be done with the information and several fields of data. Reading from these registers retrieves all information in the controller. The last item sent in the packet is a field to ensure the data is received intact. This is called a cyclic redundancy check-sum. See the following example for information on how to generate this value. All information exchanged is in hex numbers. The EZT-560i only supports the binary version of Modbus, referenced as RTU. The ASCII version is less efficient and is not supported.

The CRC (Cyclical Redundancy Checksum) is calculated by the following steps:

1. Load a 16-bit register (called CRC register) with 0xFFFF
2. Exclusive OR the first 8-bit byte of the command message with the low order byte of the 16-bit CRC register, putting the result in the CRC register.
3. Shift the CRC register one bit to the right with MSB zero filling. Extract and examine the LSB.
4. If the LSB of the CRC register is zero, repeat step 3, else Exclusive OR the CRC register with the polynomial value 0xA001.
5. Repeat steps 3 and 4 until eight shifts have been performed. When this is done, a complete 8-bit byte will have been processed.
6. Repeat steps 2 through 5 for the next 8-bit byte of the command message. Continue doing this until all bytes of the command message have been processed. The final contents of the CRC register is the CRC value.

When transmitting the CRC value in the message, the upper and lower bytes of the CRC value must be swapped, i.e. the lower order byte will be transmitted first.

Cyclical Redundancy Checksum (CRC) Algorithm

```
unsigned int calc_crc(unsigned char *start_of_packet, unsigned char *end_of_packet)
{
    unsigned int crc;
    unsigned char bit_count;
    unsigned char *char_ptr;

    /* Start at the beginning of the packet */
    char_ptr = start_of_packet;
    /* Initialize CRC */
    crc = 0xFFFF;
    /* Loop through the entire packet */
    do{
        /* Exclusive-OR the byte with the CRC */
        crc ^= (unsigned int)*char_ptr;
        /* Loop through all 8 data bits */
        bit_count = 0;
        do{
            /* If the LSB is 1, shift the CRC and XOR the polynomial mask with the CRC */
            if(crc & 0x0001){
                crc >>= 1;
                crc ^= 0xA001;
            }
            /* If the LSB is 0, shift the CRC only */
            else{
                crc >>= 1;
            }
        } while(bit_count++ < 7);
    } while(char_ptr++ < end_of_packet);
    return(crc);
}
```

2.3 Write your own Modbus Application

Listed below are a few of the more common software packages that claim to support the Modbus protocol. CSZ does not recommend any one software package nor supports the implementation of any software package not sold by CSZ. This list is provided as informational only. CSZ makes no claims as to the performance or compatibility of any software package with your. Contact the software manufacturer for more information on applying their software.

LabView by National Instruments
6504 Bridge Point Parkway
Austin, TX 78730-5039
Phone 512-794-0100
<http://www.natinst.com>

OI-2000 by Software Horizons, Inc.
10 Tower Office Park
Suite 200
Woburn, MA 01801-2120
Phone 617-933-3747
<http://www.shorizons.com>

SpecView by SpecView, LLC
41 Canyon Green Court
San Ramon, CA 94583
Phone 510-275-0600
<http://www.specview.com>

Wonderware 2000 by Wonderware
Corp.
100 Technology Drive
Irvine, CA 92718
Phone 714-727-3200
<http://www.wonderware.com>

If you already have a software application that uses Modbus, you can simply skip to EZT-560i parameter table in the Getting Started section for the information your program requires. The rest of this section provides information on writing a software application that uses Modbus.

1. You need to code messages in eight-bit bytes, with event parity, one stop bit (8, even, 1). The EZT-560i has its parity set to even as default from the factory. If a different parity setting is desired, just set the EZT-560i to match the coded parity setting.
2. Negative parameter values must be written in twos' complement format. Parameters are stored in two-byte registers accessed with read and write commands to a relative address.
3. Messages are sent in packets that must be delimited by a pause at least as long as the time it takes to send 30 bits. To determine this time in seconds, divide 30 by the baud rate. In the case of EZT-560i communications at 9600 baud, this calculates to a minimum period of 3ms.

In addition, the EZT's timeout period must be added to that, in order to properly time the send and receive messages between the host computer and multiple EZT's on the serial link. With a default timeout period in the EZT-560i of 200ms, it makes a total pause of 203ms minimum.

4. Values containing decimal points such as process values and setpoints, have the decimal point assumed, i.e., the data exchange can only be performed using whole numbers. Thus, the value must be offset by a factor of 10 in order to exchange the data correctly. For example, a setpoint of 42.4 degrees must be sent as a value of 424 in order for the EZT-560i to be set correctly. Likewise, a process value read from the EZT-560i with a value of 967 is actually 96.7 degrees. Consult the parameter table for the proper format of each value.
5. When monitoring a process, try to keep the number of read and write commands to a minimum of 500ms between exchanges to a single controller. Continuously reading data at a faster rate consumes an excess amount of the controller's processor time and does not provide any additional benefits in process monitoring.

Handling Communication Errors

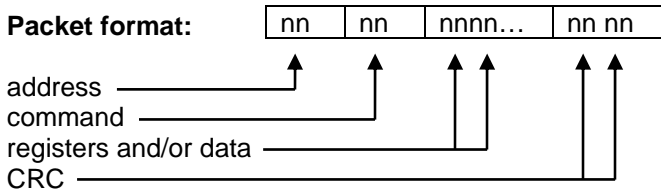
Reading or writing from/to a register that does not exist or is currently disabled will typically respond with an erroneous value or result in a time-out response, i.e., the EZT will not respond with a return message. Messages with the wrong format, timing or CRC are also ignored. A response will not be given to them. Only messages with the proper format, timing and CRC will be acknowledged. It is the user's responsibility to handle the error appropriately within their own software and determine whether to resend the message or halt for operator intervention.

User Responsibility

Refrain from altering prompts that do not appear on the EZT-560i's front panel or are not included on the specific model. Care must also be taken that the process can not cause damage to property or injury to personnel if the wrong commands are sent due to operator error or equipment malfunction. Be sure to use limit devices on any equipment placed inside the chamber that can generate heat to prevent system thermal runaway.

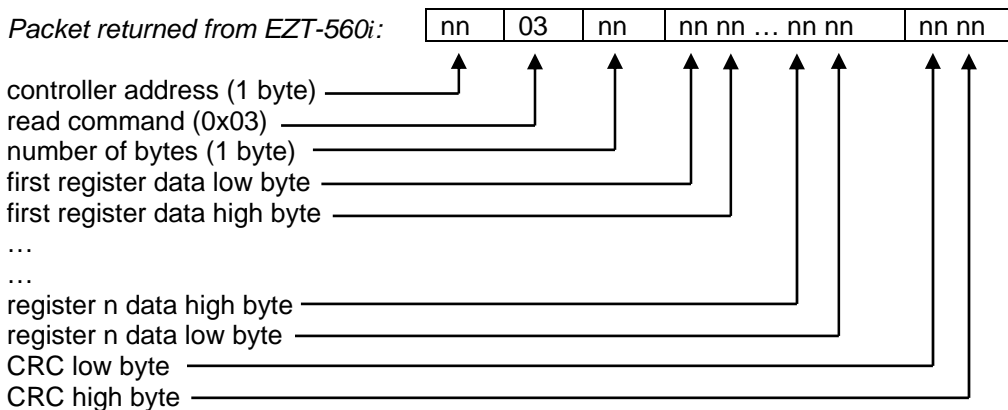
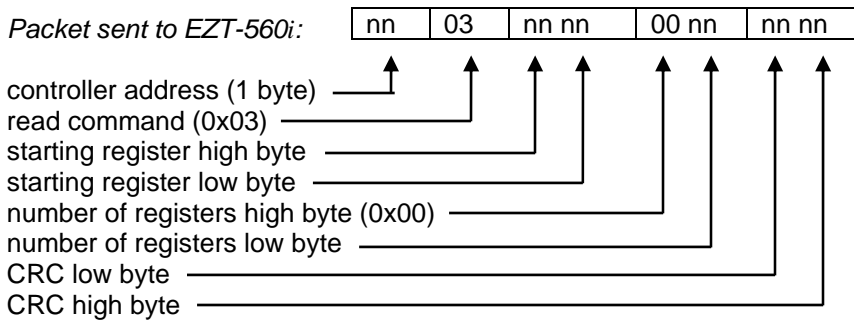
2.3.1 Packet Syntax

Each message packet begins with a one-byte controller address, from 0x01 to 0xF7. The second byte in the message packet identifies the message command: read (0x03); write (0x10); or loop back (0x08). The next n bytes of the message packet contain register addresses and/or data. The last two bytes in the message packet contain a two-byte Cyclical Redundancy Checksum (CRC) for error detection.



Read Register(s) Command (0x03)

This command returns from 1 to 60 registers. This command is to be used for reading one or more data locations from the EZT-560i.



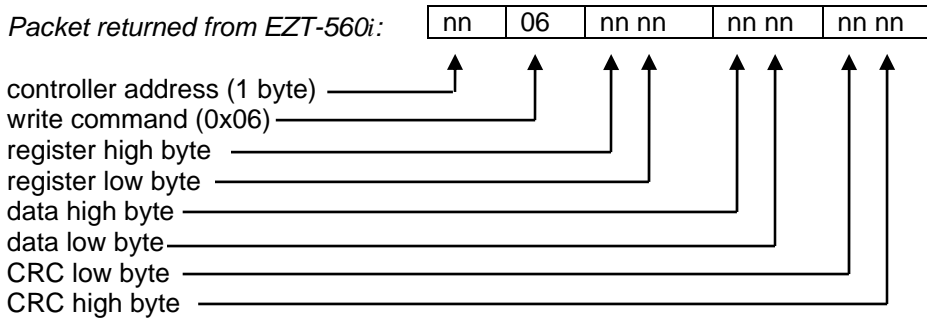
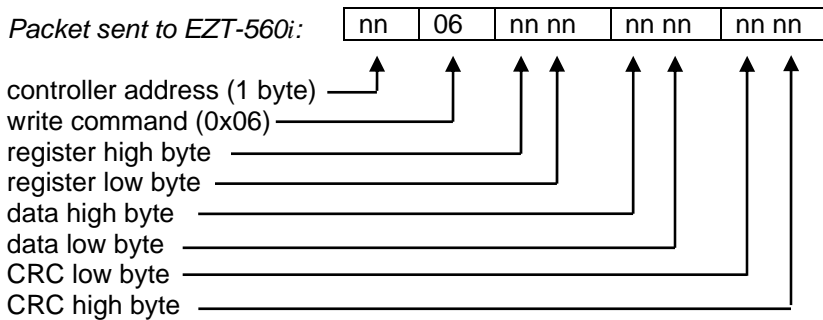
Example: Read register 61 (chamber temperature) of controller at address 1.
 Sent: 01 03 00 3D 00 01 15 C6
 Received: 01 03 02 00 EC B9 C9
 Message: 236 (0x00EC) – temperature is 23.6 degrees

EZT-560i User Communication Reference Manual

Example: Read registers 60 and 61 (chamber setpoint and temperature) of controller at address 1.
 Sent: 01 03 00 3C 00 02 04 07
 Received: 01 03 04 01 90 01 48 FA 44
 Message: 400 (0x0190) and 328 (0x0148) – setpoint is 40.0 and temperature is 32.8 degrees

Write Register Command (0x06)

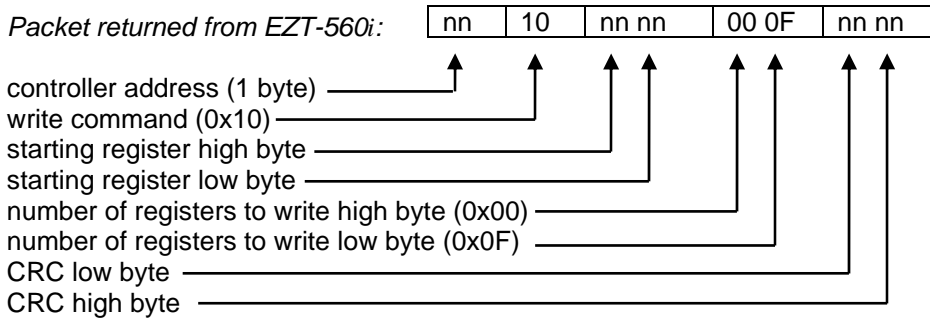
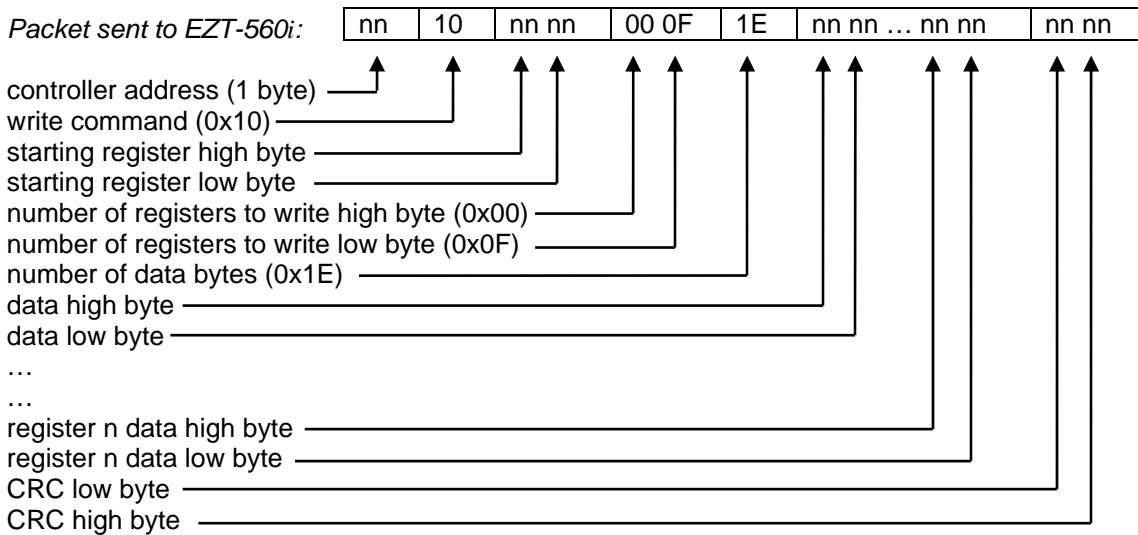
This command writes a value to a single register. This command is to be used for setting control values in the EZT-560i. To set multiple values, repeat the command for each data location.



Example: Write register 60 (temperature set point) of controller at address one to 20 degrees.
 Sent: 01 06 00 3C 00 C8 48 50
 Received: 01 06 00 3C 00 C8 48 50

Write Registers Command (0x10)

! This command is for use with profile download only. It is used to transmit profile data one step at a time to the EZT-560i. See the Profile Parameters section for the list of registers and their use. If this command is used to write to registers other than the correct profile step registers, the EZT-560i will respond with an acknowledge that the message was received; however, the command will not be executed.



2.4 EZT-560i Control Registers

The EZT-560i is capable of utilizing up to five control loops and eight monitor inputs. The register list in this section of the manual lists the associated values for all of the loops, inputs and their associated alarms by the loop or monitor input number, i.e., 1 - 5 and 1 - 8. While the monitor inputs will be easy to decipher, since they are shipped from the factory with the relative number in their tag name, the loops are not. The loop names are defined by the chamber process they control, i.e., temperature, humidity, etc., thus the number of control loops required and their function can vary between different chamber models.

The EZT-560i displays all control loops and monitor inputs in sequential order. The loop/monitor order can be viewed from the “All Loops View” screen. Starting at the top of the list and counting down, the first entry is loop 1, the second is loop 2, and so on. The following chart provides a loop number to controlled process reference for use in selecting the desired parameter from the register list.

LOOP	TEMPERATURE/HUMIDITY MODELS		ALTITUDE MODELS		VTS/TSB MODELS	DTS MODELS
1	TEMPERATURE	TEMPERATURE	TEMPERATURE	TEMPERATURE	HOT CHAMBER (BATH)	LEFT CHAMBER
2	PRODUCT	HUMIDITY	ALTITUDE	HUMIDITY	COLD CHAMBER (BATH)	CENTER CHAMBER
3	-	PRODUCT	PRODUCT	ALTITUDE	PRODUCT	RIGHT CHAMBER
4	-	-	-	PRODUCT		DUT LEFT BASKET
5	-	-	-	-		DUT RIGHT BASKET

The chamber events also vary based on the model of chamber and options present. In order to turn the chamber and associated options on and off, it is necessary to set the proper event. The chart below provides the chamber event number and its associated function based on the chamber model. The chamber events are listed in order from top to bottom on the EZT-560i “Manual Event Control” screen.

EVENT	STANDARD MODELS (TMP/RH/ALTITUDE)	VTS/TSB MODELS	DTS MODELS
1	CHAMBER	HOT CHAMBER / BATH ON	LEFT CHAMBER
2	HUMIDITY	COLD CHAMBER (VTS ONLY)	CENTER CHAMBER
3	AUX COOL	AUX COOL	AUX COOL
4	PURGE	PURGE	PURGE
5	ALTITUDE	XFR HOT	XFR LEFT
6	-	XFR COLD	XFR RIGHT
7	-	XFR UNLOAD (TSB ONLY)	RIGHT CHAMBER
8	-	-	-
9	INITIATE DEFROST	INITIATE DEFROST (VTS ONLY)	INITIATE DEFROST (CENTER ONLY)
10	PRODUCT CONTROL	PRODUCT CONTROL	PRODUCT CONTROL
11	REMOTE SETPOINT 1	REMOTE SETPOINT 1	REMOTE SETPOINT 1
12	REMOTE SETPOINT 2	REMOTE SETPOINT 2	REMOTE SETPOINT 2
13	REMOTE SETPOINT 3	REMOTE SETPOINT 3	REMOTE SETPOINT 3
14	REMOTE SETPOINT 4	REMOTE SETPOINT 4	REMOTE SETPOINT 4
15	REMOTE SETPOINT 5	REMOTE SETPOINT 5	REMOTE SETPOINT 5

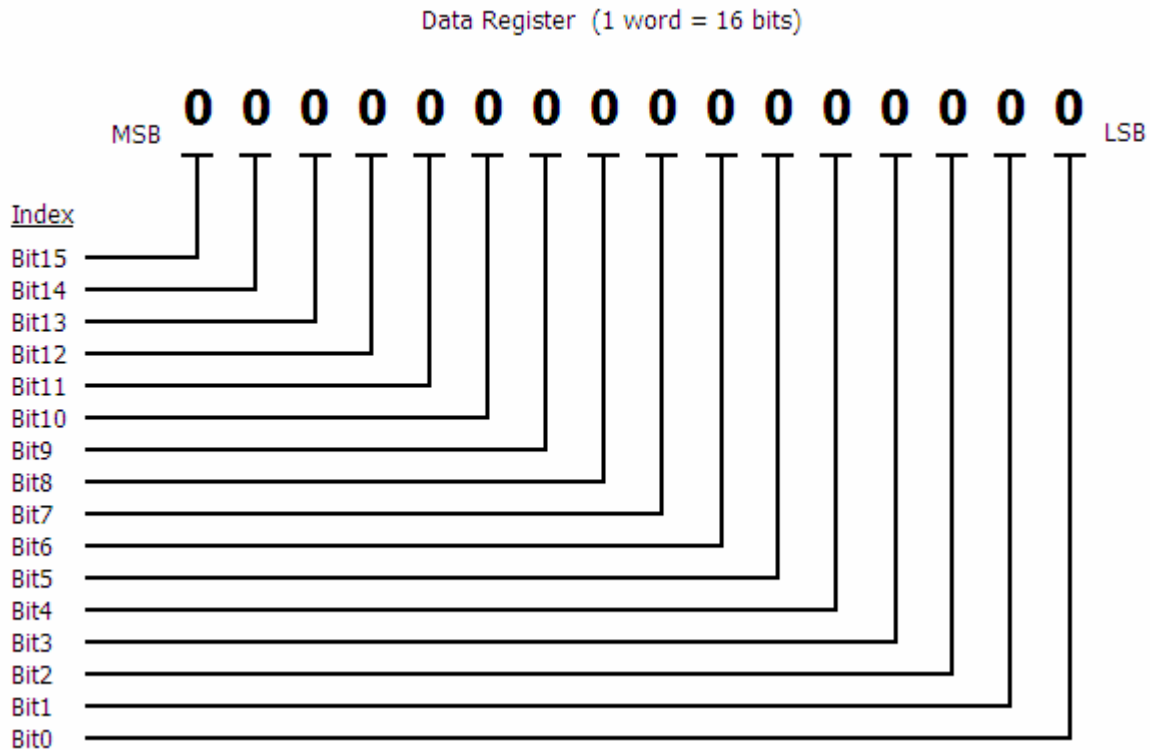
The control registers are grouped into three blocks of 60 (for a total of 180) registers relating to the specific types of data they contain. The first group of 60 registers (0 – 59) contains the configuration settings for various options on the EZT-560i as well as all of the alarm status, profile status and manual on/off settings for the chamber. The second group of 60 registers (60 – 119) contains all of the loop control/monitor settings which include the setpoint and alarm settings for each loop. The third group of 60 registers (120 – 179) contains all of the optional monitor input settings including the individual alarm settings for each.

Bit Oriented Parameters

Some of the values contained in the EZT-560i's register base contain bit oriented values. This means that each bit of the word indicates an on/off status for a specific setting or condition. In handling these values, it is recommended that the word be converted to its binary equivalent.

By converting the value to its binary equivalent, it produces a Boolean array of true [bit on (1)] and false [bit off (0)] values. This allows each bit to be examined individually. In the same manner, creating a Boolean array of 16 bits produces an equivalent decimal value that can be sent to the EZT-560i in order to set a control value.

For the purpose of this manual, parameters defined as bit oriented will have the function of each bit associated with the bit's index number in the data word. The index number is equal to that of a typical array function. Thus, an index number of zero, selects the first bit in the word. An index number of 1 selects the second bit in the word, and so on. This helps eliminate offset selection errors that may occur when coding software and using array functions to select which bit in the word that is required for examination.





Adhere to the following list of registers and their allowable data ranges. Do not attempt to write to any other register number than those listed below. Do not write to registers that are for options your chamber does not have. Failure to adhere to this requirement can result in erratic control and/or damage to equipment. Note that register numbers listed are relative values. To convert to absolute values, add 40001.

- 0 (0x0000) EZT-560i Operational Mode**
 - 0 Offline (system maintenance mode)
 - 1 Online

When the EZT-560i is in maintenance mode, it no longer updates any of the control register values. They remain at their previous setting until the EZT-560i is placed back into normal operation. When offline, the chamber is not in operation.

- 1 (0x0001) Clock (YY/MM)**
 - 0-99 Year (high byte)
 - 1-12 Month (low byte)
 - 1 January
 - 2 February
 - 3 March
 - 4 April
 - 5 May
 - 6 June
 - 7 July
 - 8 August
 - 9 September
 - 10 October
 - 11 November
 - 12 December

This parameter contains both the current year and month of the EZT-560i's clock. The data contained in the word will be an integer value based on the combined bytes of both the year and month. In order to obtain the individual values, split the word into its two component bytes.

Example: 0x0801 read from EZT-560i

Splitting the word into its two component bytes yields 0x08 for the high byte and 0x01 for the low byte. The high byte of 0x08 when converted to decimal is 8 (year of 2008). The low byte of 0x01 when converted to decimal is 1 (month of January).

2	(0x0002)	Clock (DAY/DOW)
r		1 - 31 Day of Month (high byte) 0 - 6 Day of Week (low byte) 0 Sunday 1 Monday 2 Tuesday 3 Wednesday 4 Thursday 5 Friday 6 Saturday

This parameter contains both the current day of month and day of week of the EZT-560i's clock. The data contained in the word will be an integer value based on the combined bytes of both the day of month and day of week. In order to obtain the individual values, split the word into its two component bytes.

Example: 0x1702 read from EZT-560i

Splitting the word into its two component bytes yields 0x17 for the high byte and 0x02 for the low byte. The high byte of 0x17 when converted to decimal is 23 (23rd day of the month). The low byte of 0x02 when converted to decimal is 2 (Tuesday).

3	(0x0003)	Clock (HH/MM)
r		1 - 23 Hours (high byte) 0 - 59 Minutes (low byte)

This parameter contains both the current hours and minutes of the EZT-560i's clock. The data contained in the word will be an integer value based on the combined bytes of both the hours and minutes. In order to obtain the individual values, split the word into its two component bytes.

Example: 0x1131 read from EZT-560i

Splitting the word into its two component bytes yields 0x11 for the high byte and 0x31 for the low byte. The high byte of 0x11 when converted to decimal is 17 (17:00 hours – 5pm). The low byte of 0x31 when converted to decimal is 49 (minutes). The resultant time is 17:49 or 5:49pm.

4	(0x0004)	Clock (seconds)
r		0 - 59 seconds

5	(0x0005)	Power Recovery Mode
r/w		0 Continue 1 Hold 2 Terminate 4 Reset 8 Resume

6	(0x0006)	Power Out Time
r/w		0 - 32767 seconds

- 7 (0x0007) Defrost Operating Mode**
r/w
 - 0 Disabled
 - 1 Manual Mode Selected
 - 2 Auto Mode Selected

- 8 (0x0008) Auto Defrost Temperature Setpoint**
r/w
 - 32768 – 32767 (-3276.8 – 3276.7 degrees)

This parameter contains an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the value contained in this register must be divided by 10 to get the actual value. Likewise, when sending a setpoint, the setpoint must be multiplied by 10 in order for it to be set properly in the EZT-560i.

- 9 (0x0009) Auto Defrost Time Interval**
r/w
 - 0 - 32767 minutes

- 10 (0x000A) Defrost Status**
r
 - 0 Not in Defrost
 - 1 In Defrost
 - 2 In Prechill

- 11 (0x000B) Time Remaining Until Next Defrost**
r
 - 0 - 32767 minutes

- 12 (0x000C) Product Control**
r/w
 - 0 Off
 - 1 Deviation
 - 2 Process
 - 4 Off
 - 5 Deviation using Event for enable
 - 6 Process using Event for enable

- 13 (0x000D) Product Control Upper Setpoint**
- 14 (0x000E) Product Control Lower Setpoint**
r/w
 - 32768 – 32767 (-3276.8 – 3276.7 degrees)

These parameters contain an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the values contained in these registers must be divided by 10 to get the actual values. Likewise, when sending the setpoints, the setpoints must be multiplied by 10 in order for it to be set properly in the EZT-560i.

- 15 (0x000F) Condensation Control**
r/w
 - 0 Off
 - 1 On

- 16 (0x0010) Condensation Control Monitor Mode**
r/w
 - 1 Use Single Input
 - 2 Use Lowest Input
 - 4 Use Highest Input
 - 8 Use Average of all Inputs

17 (0x0011) Condensation Control Input Selection
r/w

- Bit0 Product
- Bit1 PV1 (monitor)
- Bit2 PV2 (monitor)
- Bit3 PV3 (monitor)
- Bit4 PV4 (monitor)
- Bit5 PV5 (monitor)
- Bit6 PV6 (monitor)
- Bit7 PV7 (monitor)
- Bit8 PV8 (monitor)

This parameter is bit oriented, i.e., enabling the different bits of the word enables (1) or disables (0) the use of the input for condensation control. Note that if monitor inputs are not available on your chamber, the associated bits should be set to zero.

Example: Select the product, PV1, PV5, PV6 and PV7 inputs for control

Set the bits of the word for the selected inputs. The bit number defines the position (index) of the bit (element) in the word which can be thought of as an array of bits starting at the LSB.

The bit values then become: 00000000 11100011. The decimal equivalent of the binary array is 227 (0x00E3). By setting register 17 to a value of 227, the selected inputs will be used.

18 (0x0012) Condensation Control Temperature Ramp Rate Limit
r/w

- 0 - 100 (0.0 – 10.0 degrees C)
- 0 - 180 (0.0 – 18.0 degrees F)

This parameter contains an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the value contained in this register must be divided by 10 to get the actual value. Likewise, when sending a setpoint, the setpoint must be multiplied by 10 in order for it to be set properly in the EZT-560i.

19 (0x0013) Condensation Control Dewpoint Limit
20 (0x0014) Condensation Control Dewpoint Actual
r

- 32768 – 32767 (-3276.8 – 3276.7 degrees)

These parameters contain an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the values contained in these registers must be divided by 10 to get the actual values.

21 (0x0015) Chamber Light Control
r/w

- 0 Chamber Light Off
- 1 Chamber Light On

22	(0x0016)	Chamber Manual Event Control
23	(0x0017)	Customer Manual Event Control
r/w		Bit0 Event 1
		Bit1 Event 2
		Bit2 Event 3
		Bit3 Event 4
		Bit4 Event 5
		Bit5 Event 6
		Bit6 Event 7
		Bit7 Event 8
		Bit8 Event 9
		Bit9 Event 10
		Bit10 Event 11
		Bit11 Event 12
		Bit12 Event 13
		Bit13 Event 14
		Bit14 Event 15

These parameters are bit oriented, i.e., enabling the different bits of the word turns on (1) or turns off (0) the event. Note that if an event is for controlling an option not available on your chamber, the associated bit should be set to zero.

Example: Turn on the chamber.

According to the event table at the beginning of this section, the chamber event for a standard temperature/humidity chamber is event 1. The bit number for event 1 is zero, thus the bit at position (index) zero of the word should be set.

The bit values of the word then become: 00000000 00000001. The decimal equivalent of the binary array is 1 (0x0001). By setting register 22 to a value of 1, the chamber will turn on.

Example: Turn on customer events 7, 10, 14 and 15.

By comparing the event numbers to their bit positions, set the bits in the word accordingly: 0110001001000000. The decimal equivalent is 25152 (0x6240). Setting register 23 to a value of 25152 will turn on each of the selected customer events.

24	(0x0018)	Profile Control/Status
r/w		<ul style="list-style-type: none"> 0 Stop/Off 1 Stop/All Off 2 Hold 4 Run/Resume 8 Autostart 16 Wait 32 Ramp 64 Soak 128 Guaranteed Soak

This parameter is used to control (start/stop) a profile and to monitor the operating status of the profile. The control values; 0, 1, 2 and 4 are used to stop, hold or run a profile. The status values; 8, 16, 32, 64 and 128 are used to indicate the mode of operation of the profile. These values are set by the EZT-560i based on the step type or operating condition that the profile is in.

If the profile is placed into hold by setting a value of 2 the register, it will remain in hold until it is changed back to run by setting a value of 4 to the register. Once placed back into run, the EZT-560i will then change the value of the register back to one of the five status values based on operating status.

25	(0x0019)	Profile Advance Step
w		<ul style="list-style-type: none"> 1 Advance Previous Step 2 Advance Next Step

This parameter automatically resets to zero once the command is executed.

26	(0x001A)	Profile Name (characters 1 and 2)
27	(0x001B)	Profile Name (characters 3 and 4)
28	(0x001C)	Profile Name (characters 5 and 6)
29	(0x001D)	Profile Name (characters 7 and 8)
30	(0x001E)	Profile Name (characters 9 and 10)
r		<ul style="list-style-type: none"> 32 – 126 (high byte) 32 – 126 (low byte)

These parameters store the profile name up to 10 characters in length. The decimal values are representative of the standard ASCII character set for printable characters. The characters of the profile name are stored in sequence through the registers from low byte to high byte starting with register 26.

Example: Read registers 26 – 30 from EZT-560i and convert byte values to ASCII equivalents:

0x4554	0x5453	0x3120	0x3332	0x2020
E T	T S	1	3 2	

Put ASCII values in order from low to high byte starting with register 26 in order to assemble the profile name: TEST 123 . Note that null characters are not used at the end of the profile name. A space is used in place of a null character to maintain the 10 character name length if the profile name is not at least ten characters long.

31	(0x001F)	Profile Start Date (YY/MM)
34	(0x0022)	Profile Stop Date (YY/MM)
r		0-99 Year (high byte)
		1-12 Month (low byte)
		1 January
		2 February
		3 March
		4 April
		5 May
		6 June
		7 July
		8 August
		9 September
		10 October
		11 November
		12 December

These parameters contain both the current year and month of the profile's start/stop dates. The data contained in the word will be an integer value based on the combined bytes of both the year and month. In order to obtain the individual values, split the word into its two component bytes.

Example: 0x0801 read from EZT-560i

Splitting the word into its two component bytes yields 0x08 for the high byte and 0x01 for the low byte. The high byte of 0x08 when converted to decimal is 8 (year of 2008). The low byte of 0x01 when converted to decimal is 1 (month of January).

32	(0x0020)	Profile Start Date (DAY/DOW)
35	(0x0023)	Profile Stop Date (DAY/DOW)
r		1 - 31 Day of Month (high byte)
		0 - 6 Day of Week (low byte)
		0 Sunday
		1 Monday
		2 Tuesday
		3 Wednesday
		4 Thursday
		5 Friday
		6 Saturday

These parameters contain both the current day of month and day of week of the profiles start/stop date. The data contained in the word will be an integer value based on the combined bytes of both the day of month and day of week. In order to obtain the individual values, split the word into its two component bytes.

Example: 0x1702 read from EZT-560i

Splitting the word into its two component bytes yields 0x17 for the high byte and 0x02 for the low byte. The high byte of 0x17 when converted to decimal is 23 (23rd day of the month). The low byte of 0x02 when converted to decimal is 2 (Tuesday).

- 33 (0x0021) Profile Start Date (HH/MM)**
36 (0x0024) Profile Stop Date (HH/MM)
 r 1 - 23 Hours (high byte)
 0 - 59 Minutes (low byte)

These parameters contain both the current hours and minutes of the profiles start/stop date. The data contained in the word will be an integer value based on the combined bytes of both the hours and minutes. In order to obtain the individual values, split the word into its two component bytes.

Example: 0x1131 read from EZT-560i

Splitting the word into its two component bytes yields 0x11 for the high byte and 0x31 for the low byte. The high byte of 0x11 when converted to decimal is 17 (17:00 hours – 5pm). The low byte of 0x31 when converted to decimal is 49 (minutes). The resultant time is 17:49 or 5:49pm.

- 37 (0x0025) Profile Start Step**
 r/w 1 - 99

This parameter is used to set the step that the profile will start on. Note that once the profile has started, this register will be set to zero. The EZT-560i requires that this register be set prior to starting a profile each time, in order to prevent a profile from being started on the wrong step.

- 38 (0x0026) Profile Current Step**
39 (0x0027) Profile Last Step
 r 1 - 99

- 40 (0x0028) Profile Time Left in Current Step (HHH)**
 r 1 – 999 Hours

- 41 (0x0029) Profile Time Left in Current Step (MM/SS)**
 r 0 - 59 Minutes (high byte)
 0 - 59 Seconds (low byte)

These parameters contain both the current minutes and seconds left in the current step of the profiles. The data contained in the word will be an integer value based on the combined bytes of both the minutes and seconds. In order to obtain the individual values, split the word into its two component bytes.

Example: 0x1131 read from EZT-560i

Splitting the word into its two component bytes yields 0x11 for the high byte and 0x31 for the low byte. The high byte of 0x11 when converted to decimal is 17. The low byte of 0x31 when converted to decimal is 49. The resultant time left is then 17 minutes and 31 seconds.

42	(0x002A)	Profile Wait for Status
r		0 Not Waiting
		1 Input 1
		2 Input 2
		4 Input 3
		8 Input 4
		16 Input 5
		32 Input 6
		64 Input 7
		128 Input 8
		256 Input 9
		512 Input 10
		1024 Input 11
		2048 Input 12
		4096 Input 13
		8192 Digital Input

43	(0x002B)	Profile Wait for Setpoint
r		-32768 – 32767 (-3276.8 – 3276.7)

This parameter contains an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the value contained in this register must be divided by 10 to get the actual value.

If the EZT-560i is in a “wait for digital input” state, the value contained in this register will be the number of the digital input that the EZT-560i is waiting for.

44	(0x002C)	Profile Current Jump Step
r		1 – 99

45	(0x002D)	Profile Jumps Remaining in Current Step
r		0 – 999

46	(0x002E)	Profile Loop 1 Target Setpoint
47	(0x002F)	Profile Loop 2 Target Setpoint
48	(0x0030)	Profile Loop 3 Target Setpoint
49	(0x0031)	Profile Loop 4 Target Setpoint
50	(0x0032)	Profile Loop 5 Target Setpoint
r		-32768 – 32767 (-3276.8 – 3276.7)

These parameters contain an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the values contained in these registers must be divided by 10 to get the actual values.

51	(0x0033)	Profile Last Jump from Step
52	(0x0034)	Profile Last Jump to Step
r		1 – 99

53	(0x0035)	Profile Total Jumps Made
r		0 – 32767

- 54 (0x0036) Alarm Acknowledge**
 w 1 Alarm Silence
 2 Pumpdown Reset

This parameter automatically resets to zero once the command is executed.

- 55 (0x0037) EZT-560i Alarm Status**
 r Bit0 Input 1 Sensor Break
 Bit1 Input 2 Sensor Break
 Bit2 Input 3 Sensor Break
 Bit3 Input 4 Sensor Break
 Bit4 Input 5 Sensor Break
 Bit5 Input 6 Sensor Break
 Bit6 Input 7 Sensor Break
 Bit7 Input 8 Sensor Break
 Bit8 Input 9 Sensor Break
 Bit9 Input 10 Sensor Break
 Bit10 Input 11 Sensor Break
 Bit11 Input 12 Sensor Break
 Bit12 Input 13 Sensor Break
 Bit13 (not assigned)
 Bit14 EZT-560i Loop Communications Fault

This parameter is bit oriented, i.e., the individual bits of the word indicate a specific alarm condition. When the bit is on (1) the alarm is present. Note that more than one alarm can be present at a time.

- 56 (0x0038) Input Alarm Status**
 r Bit0 Input 1 Alarm
 Bit1 Input 2 Alarm
 Bit2 Input 3 Alarm
 Bit3 Input 4 Alarm
 Bit4 Input 5 Alarm
 Bit5 Input 6 Alarm
 Bit6 Input 7 Alarm
 Bit7 Input 8 Alarm
 Bit8 Input 9 Alarm
 Bit9 Input 10 Alarm
 Bit10 Input 11 Alarm
 Bit11 Input 12 Alarm
 Bit12 Input 13 Alarm
 Bit13 (not assigned)
 Bit14 (not assigned)

This parameter is bit oriented, i.e., the individual bits of the word indicate a specific alarm condition. When the bit is on (1) the alarm is present. Note that more than one alarm can be present at a time.

The alarm point is determined by the alarm settings for the associated loop or monitor input. If the alarm has not been configured, the alarm bit will remain off.

57	(0x0039)	Chamber Alarm Status
r		Bit0 Chamber High Limit
		Bit1 External Product Safety
		Bit2 Boiler Over-Temperature
		Bit3 Boiler Low Water
		Bit4 Dehumidifier System Fault
		Bit5 Motor Overload
		Bit6 Fluid System High Limit
		Bit7 Fluid System High Pressure
		Bit8 Fluid System Low Flow
		Bit9 (not assigned)
		Bit10 (not assigned)
		Bit11 (not assigned)
		Bit12 Emergency Stop
		Bit13 Power Failure
		Bit14 Transfer Error

This parameter is bit oriented, i.e., the individual bits of the word indicate a specific alarm condition. When the bit is on (1) the alarm is present. Note that more than one alarm can be present at a time.

58	(0x003A)	Refrigeration Alarm Status
r		Bit0 System 1 High/Low Pressure
		Bit1 System 1 Low Oil Pressure
		Bit2 System 1 High Discharge Temperature
		Bit3 System 1 Compressor Protection Module
		Bit4 Pumpdown Disabled
		Bit5 (not assigned)
		Bit6 (not assigned)
		Bit7 (not assigned)
		Bit8 System 2 High/Low Pressure
		Bit9 System 2 Low Oil Pressure
		Bit10 System 2 High Discharge Temperature
		Bit11 System 2 Compressor Protection Module
		Bit12 (not assigned)
		Bit13 (not assigned)
		Bit14 (not assigned)

This parameter is bit oriented, i.e., the individual bits of the word indicate a specific alarm condition. When the bit is on (1) the alarm is present. Note that more than one alarm can be present at a time.

59	(0x003B)	System Status Monitor
r		Bit0 Humidity Water Reservoir Low Bit1 Humidity Disabled (temperature out-of-range) Bit2 Humidity High Dewpoint Limit Bit3 Humidity Low Dewpoint Limit Bit4 Door Open Bit5 (not assigned) Bit6 (not assigned) Bit7 (not assigned) Bit8 Service Air Circulators Bit9 Service Heating/Cooling System Bit10 Service Humidity System Bit11 Service Purge System Bit12 Service Altitude System Bit13 Service Transfer Mechanism Bit14 (not assigned)

This parameter is bit oriented, i.e., the individual bits of the word indicate a specific alarm condition. When the bit is on (1) the alarm is present. Note that more than one alarm can be present at a time. The alarms are in sequential order as they appear on the "System Status Monitor" screen on the EZT-560i.

60	(0x003C)	Loop 1 Setpoint (SP)
72	(0x0048)	Loop 2 Setpoint (SP)
84	(0x0054)	Loop 3 Setpoint (SP)
96	(0x0060)	Loop 4 Setpoint (SP)
108	(0x006C)	Loop 5 Setpoint (SP)
r/w		-32768 – 32767 (-3276.8 – 3276.7)

These parameters contain an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the values contained in these registers must be divided by 10 to get the actual values. Likewise, when sending the setpoints, the setpoints must be multiplied by 10 in order for it to be set properly in the EZT-560i.

61	(0x003D)	Loop 1 Process Value (PV)
73	(0x0049)	Loop 2 Process Value (PV)
85	(0x0055)	Loop 3 Process Value (PV)
97	(0x0061)	Loop 4 Process Value (PV)
109	(0x006D)	Loop 5 Process Value (PV)
r		-32768 – 32767 (-3276.8 – 3276.7)

These parameters contain an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the values contained in these registers must be divided by 10 to get the actual values.

EZT-560i User Communication Reference Manual

62	(0x003E)	Loop 1 Percent Output (%out)
74	(0x004A)	Loop 2 Percent Output (%out)
86	(0x0056)	Loop 3 Percent Output (%out)
98	(0x0062)	Loop 4 Percent Output (%out)
110	(0x006E)	Loop 5 Percent Output (%out)
r		-10000 – 10000 (-100.00 – 100.00)

These parameters contain an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the values contained in these registers must be divided by 100 to get the actual values.

63	(0x003F)	Loop 1 Autotune Status
75	(0x004B)	Loop 2 Autotune Status
87	(0x0057)	Loop 3 Autotune Status
99	(0x0063)	Loop 4 Autotune Status
111	(0x006F)	Loop 5 Autotune Status
r/w		0 Autotune Off
		1 Start Autotune
		2 Autotune In Progress
		4 Cancel Autotune

64	(0x0040)	Loop 1 Upper Setpoint Limit
65	(0x0041)	Loop 1 Lower Setpoint Limit
r/w		-32768 – 32767 (-3276.8 – 3276.7)

76	(0x004C)	Loop 2 Upper Setpoint Limit
77	(0x004D)	Loop 2 Lower Setpoint Limit
r/w		-32768 – 32767 (-3276.8 – 3276.7)

88	(0x0058)	Loop 3 Upper Setpoint Limit
89	(0x0059)	Loop 3 Lower Setpoint Limit
r/w		-32768 – 32767 (-3276.8 – 3276.7)

100	(0x0064)	Loop 4 Upper Setpoint Limit
101	(0x0065)	Loop 4 Lower Setpoint Limit
r/w		-32768 – 32767 (-3276.8 – 3276.7)

112	(0x0070)	Loop 5 Upper Setpoint Limit
113	(0x0071)	Loop 5 Lower Setpoint Limit
r/w		-32768 – 32767 (-3276.8 – 3276.7)

These parameters contain an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the values contained in these registers must be divided by 10 to get the actual values. Likewise, when sending the setpoints, the setpoints must be multiplied by 10 in order for it to be set properly in the EZT-560i.



The loop setpoint limits are typically protected from adjustment through the EZT-560i's security settings. However, these values are not protected over the communications interface and can be changed remotely. Safeguards should be put in place in user software to prevent setpoint limits from exceeded chamber design limits.

66	(0x0042)	Loop 1 Alarm Type
78	(0x004E)	Loop 2 Alarm Type
90	(0x005A)	Loop 3 Alarm Type
102	(0x0066)	Loop 4 Alarm Type
114	(0x0072)	Loop 5 Alarm Type
r/w		0 Alarm Off
		3 Process High
		5 Process Low
		7 Process Both
		24 Deviation High
		40 Deviation Low
		56 Deviation Both

67	(0x0043)	Loop 1 Alarm Mode
79	(0x004F)	Loop 2 Alarm Mode
91	(0x005B)	Loop 3 Alarm Mode
103	(0x0067)	Loop 4 Alarm Mode
115	(0x0073)	Loop 5 Alarm Mode
r/w		Bit0 Alarm Self Clears (0) Alarm Latches (1)
		Bit1 Close on Alarm (0) – action of assigned alarm output Open on Alarm (1) – action of assigned alarm output
		Bit4 Audible Alarm Off (0) Audible Alarm On (1)
		Bit5 Chamber Continues On Alarm (0) Chamber Shuts Down On Alarm (1)

Parameter is bit oriented. Note that only bits listed perform control action. The state of the other bits do not affect operation.

68	(0x0044)	Loop 1 Alarm Output Assignment
80	(0x0050)	Loop 2 Alarm Output Assignment
92	(0x005C)	Loop 3 Alarm Output Assignment
104	(0x0068)	Loop 4 Alarm Output Assignment
116	(0x0074)	Loop 5 Alarm Output Assignment
r/w		0 No Output Selected
		1 Digital Output (Customer Event) 1 Selected
		2 Digital Output (Customer Event) 2 Selected
		4 Digital Output (Customer Event) 3 Selected
		8 Digital Output (Customer Event) 4 Selected
		16 Digital Output (Customer Event) 5 Selected
		32 Digital Output (Customer Event) 6 Selected
		64 Digital Output (Customer Event) 7 Selected
		128 Digital Output (Customer Event) 8 Selected
		256 Digital Output (Customer Event) 9 Selected
		512 Digital Output (Customer Event) 10 Selected
		1024 Digital Output (Customer Event) 11 Selected
		2048 Digital Output (Customer Event) 12 Selected
		4096 Digital Output (Customer Event) 13 Selected
		8192 Digital Output (Customer Event) 14 Selected
		16384 Digital Output (Customer Event) 15 Selected

EZT-560i User Communication Reference Manual

69	(0x0045)	Loop 1 High Alarm Setpoint
70	(0x0046)	Loop 1 Low Alarm Setpoint
r/w		-32768 – 32767 (-3276.8 – 3276.7)
81	(0x0051)	Loop 2 High Alarm Setpoint
82	(0x0052)	Loop 2 Low Alarm Setpoint
r/w		-32768 – 32767 (-3276.8 – 3276.7)
93	(0x005D)	Loop 3 High Alarm Setpoint
94	(0x005E)	Loop 3 Low Alarm Setpoint
r/w		-32768 – 32767 (-3276.8 – 3276.7)
105	(0x0069)	Loop 4 High Alarm Setpoint
106	(0x006A)	Loop 4 Low Alarm Setpoint
r/w		-32768 – 32767 (-3276.8 – 3276.7)
117	(0x0075)	Loop 5 High Alarm Setpoint
118	(0x0076)	Loop 5 Low Alarm Setpoint
r/w		-32768 – 32767 (-3276.8 – 3276.7)

These parameters contain an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the values contained in these registers must be divided by 10 to get the actual values. Likewise, when sending the setpoints, the setpoints must be multiplied by 10 in order for it to be set properly in the EZT-560i.

71	(0x0047)	Loop 1 Alarm Hysteresis
83	(0x0053)	Loop 2 Alarm Hysteresis
95	(0x005F)	Loop 3 Alarm Hysteresis
107	(0x006B)	Loop 4 Alarm Hysteresis
119	(0x0077)	Loop 5 Alarm Hysteresis
r/w		0 – 32767 (0.0 – 3276.7)

These parameters contain an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the values contained in these registers must be divided by 10 to get the actual values. Likewise, when sending the setpoints, the setpoints must be multiplied by 10 in order for it to be set properly in the EZT-560i.

120	(0x0078)	Monitor Input 1 Process Value (PV)
127	(0x007F)	Monitor Input 2 Process Value (PV)
134	(0x0086)	Monitor Input 3 Process Value (PV)
141	(0x008D)	Monitor Input 4 Process Value (PV)
148	(0x0094)	Monitor Input 5 Process Value (PV)
155	(0x009B)	Monitor Input 6 Process Value (PV)
162	(0x00A2)	Monitor Input 7 Process Value (PV)
169	(0x00A9)	Monitor Input 8 Process Value (PV)
r		-32768 – 32767 (-3276.8 – 3276.7)

These parameters contain an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the values contained in these registers must be divided by 10 to get the actual values.

121	(0x0079)	Monitor Input 1 Alarm Type
128	(0x0080)	Monitor Input 2 Alarm Type
135	(0x0087)	Monitor Input 3 Alarm Type
142	(0x008E)	Monitor Input 4 Alarm Type
149	(0x0095)	Monitor Input 5 Alarm Type
156	(0x009C)	Monitor Input 6 Alarm Type
163	(0x00A3)	Monitor Input 7 Alarm Type
170	(0x00AA)	Monitor Input 8 Alarm Type
r/w		0 Alarm Off
		3 Process High
		5 Process Low
		7 Process Both

122	(0x007A)	Monitor Input 1 Alarm Mode
129	(0x0081)	Monitor Input 2 Alarm Mode
136	(0x0088)	Monitor Input 3 Alarm Mode
143	(0x008F)	Monitor Input 4 Alarm Mode
150	(0x0096)	Monitor Input 5 Alarm Mode
157	(0x009D)	Monitor Input 6 Alarm Mode
164	(0x00A4)	Monitor Input 7 Alarm Mode
171	(0x00AB)	Monitor Input 8 Alarm Mode
r/w		Bit0 Alarm Self Clears (0) Alarm Latches (1)
		Bit1 Close on Alarm (0) – action of assigned alarm output Open on Alarm (1) – action of assigned alarm output
		Bit4 Audible Alarm Off (0) Audible Alarm On (1)
		Bit5 Chamber Continues On Alarm (0) Chamber Shuts Down On Alarm (1)

Parameter is bit oriented. Note that only bits listed perform control action. The state of the other bits do not affect operation.

EZT-560i User Communication Reference Manual

123	(0x007B)	Monitor Input 1 Alarm Output Assignment
130	(0x0082)	Monitor Input 2 Alarm Output Assignment
137	(0x0089)	Monitor Input 3 Alarm Output Assignment
144	(0x0090)	Monitor Input 4 Alarm Output Assignment
151	(0x0097)	Monitor Input 5 Alarm Output Assignment
158	(0x009E)	Monitor Input 6 Alarm Output Assignment
165	(0x00A5)	Monitor Input 7 Alarm Output Assignment
172	(0x00AC)	Monitor Input 8 Alarm Output Assignment
r/w		0 No Output Selected
		1 Digital Output (Customer Event) 1 Selected
		2 Digital Output (Customer Event) 2 Selected
		4 Digital Output (Customer Event) 3 Selected
		8 Digital Output (Customer Event) 4 Selected
		16 Digital Output (Customer Event) 5 Selected
		32 Digital Output (Customer Event) 6 Selected
		64 Digital Output (Customer Event) 7 Selected
		128 Digital Output (Customer Event) 8 Selected
		256 Digital Output (Customer Event) 9 Selected
		512 Digital Output (Customer Event) 10 Selected
		1024 Digital Output (Customer Event) 11 Selected
		2048 Digital Output (Customer Event) 12 Selected
		4096 Digital Output (Customer Event) 13 Selected
		8192 Digital Output (Customer Event) 14 Selected
		16384 Digital Output (Customer Event) 15 Selected

124 (0x007C) **Monitor Input 1 High Alarm Setpoint**
 125 (0x007D) **Monitor Input 1 Low Alarm Setpoint**
 r/w -32768 – 32767 (-3276.8 – 3276.7)

131 (0x0083) **Monitor Input 2 High Alarm Setpoint**
 132 (0x0084) **Monitor Input 2 Low Alarm Setpoint**
 r/w -32768 – 32767 (-3276.8 – 3276.7)

138 (0x008A) **Monitor Input 3 High Alarm Setpoint**
 139 (0x008B) **Monitor Input 3 Low Alarm Setpoint**
 r/w -32768 – 32767 (-3276.8 – 3276.7)

145 (0x0091) **Monitor Input 4 High Alarm Setpoint**
 146 (0x0092) **Monitor Input 4 Low Alarm Setpoint**
 r/w -32768 – 32767 (-3276.8 – 3276.7)

152 (0x0098) **Monitor Input 5 High Alarm Setpoint**
 153 (0x0099) **Monitor Input 5 Low Alarm Setpoint**
 r/w -32768 – 32767 (-3276.8 – 3276.7)

159 (0x009F) **Monitor Input 6 High Alarm Setpoint**
 160 (0x00A0) **Monitor Input 6 Low Alarm Setpoint**
 r/w -32768 – 32767 (-3276.8 – 3276.7)

166 (0x00A6) **Monitor Input 7 High Alarm Setpoint**
 167 (0x00A7) **Monitor Input 7 Low Alarm Setpoint**
 r/w -32768 – 32767 (-3276.8 – 3276.7)

- 173 (0x00AD) **Monitor Input 8 High Alarm Setpoint**
 174 (0x00AE) **Monitor Input 8 Low Alarm Setpoint**
 r/w -32768 – 32767 (-3276.8 – 3276.7)

These parameters contain an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the values contained in these registers must be divided by 10 to get the actual values. Likewise, when sending the setpoints, the setpoints must be multiplied by 10 in order for it to be set properly in the EZT-560i.

- 126 (0x007E) **Monitor Input 1 Alarm Hysteresis**
 133 (0x0085) **Monitor Input 2 Alarm Hysteresis**
 140 (0x008C) **Monitor Input 3 Alarm Hysteresis**
 147 (0x0093) **Monitor Input 4 Alarm Hysteresis**
 154 (0x009A) **Monitor Input 5 Alarm Hysteresis**
 161 (0x00A1) **Monitor Input 6 Alarm Hysteresis**
 168 (0x00A8) **Monitor Input 7 Alarm Hysteresis**
 175 (0x00AF) **Monitor Input 8 Alarm Hysteresis**
 r/w 0 – 32767 (0.0 – 3276.7)

These parameters contain an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the values contained in these registers must be divided by 10 to get the actual values. Likewise, when sending the setpoints, the setpoints must be multiplied by 10 in order for it to be set properly in the EZT-560i.

- 179 (0x00B3) **Profile Step Time Adjustment**
 w 0 – 32767 minutes

- 180 (0x00B4) **EZT560i Offline/Downloading Profile**
 r 0 Online
 1 Offline/Downloading Profile

When the EZT-560i is offline/downloading a profile, refrain from writing to any control registers. In offline mode, there are no updates made to any control registers.

After a profile transfer to the EZT-560i from a PC, the EZT-560i will go into profile download. Do not write to any registers until profile download is complete. Once profile download complete, normal operation may commence.

2.5 EZT-560i Profile Registers

The profile parameters are a separate group of registers that are used for sending profiles to the EZT-560i. The manner in which the profile steps are sent to the EZT-560i is specific and must be followed exactly. Each step of the profile consists of 15 data registers. A profile is written one step at a time, using a single command to transmit the data for all 15 registers at once. The steps must be sent one at a time with a 1 second pause between steps.

You must use the write multiple registers command (0x10) to transmit the profile to the EZT-560i. See Section 2.3.1, Packet Syntax, for the command format.



It is the user’s responsibility to make sure that the format and data ranges for each step of the profile is followed correctly. Failure to do so can result in erratic control and or damage to equipment. Note that register numbers listed are relative values. To convert to absolute values, add 40001.

The first 15 registers of the profile download contain specific values related to the profile. These include the Autostart settings, profile name, length of the profile and guaranteed soak settings. These values are always transmitted as the first “step” of the profile:

200	(0x00C8)	Autostart	
w		0	Off
		1	Start by Date
		2	Start by Day
201	(0x00C9)	Autostart Time (YY/MM)	
w		0-99	Year (high byte)
		1-12	Month (low byte)
		1	January
		2	February
		3	March
		4	April
		5	May
		6	June
		7	July
		8	August
		9	September
		10	October
		11	November
		12	December

This parameter contains both the current year and month of the profile’s autostart date. The data contained in the word will be an integer value based on the combined bytes of both the year and month. In order to create the value, join the individual bytes into a single word.

Example: Set the year to 2010 and the month to February

To create the proper data value, convert the two digit year “10” to hexadecimal (x0A). Convert the month “2” to hexadecimal (x02). Combine the two hexadecimal bytes to create the word “x0A02”.

202	(0x00CA)	Autostart Time (DAY/DOW)
w	1 - 31	Day of Month (high byte)
	0 - 6	Day of Week (low byte)
	0	Sunday
	1	Monday
	2	Tuesday
	3	Wednesday
	4	Thursday
	5	Friday
	6	Saturday

This parameter contains both the current day of month and day of week of the EZT-560i's clock. The data contained in the word will be an integer value based on the combined bytes of both the day of month and day of week. In order to create the value, join the individual bytes into a single word.

Example: Set the day to 23 and the day of week to Sunday.

To create the proper data value, convert the day of 23 to hexadecimal (x17). Convert the day of week, Sunday (0) to hexadecimal (x00). Combine the two hexadecimal bytes to create the word "x1700".

203	(0x00CB)	Autostart Time (HH/MM)
w	1 - 23	Hours (high byte)
	0 - 59	Minutes (low byte)

This parameter contains the time in both hours and minutes for when the profile is to start when Autostart is enabled. The data contained in the word will be an integer value based on the combined bytes of both the hours and minutes. In order to create the value, join the individual bytes into a single word.

Example: Set a start time of 5:49pm.

Convert the hours into the 24 hour clock format (5+12) which yields 17 (x11). Convert the seconds into hexadecimal (x31). Combine the two bytes to form the word "x1131"

204	(0x00CC)	Profile Name (characters 1 and 2)
205	(0x00CD)	Profile Name (characters 3 and 4)
206	(0x00CE)	Profile Name (characters 5 and 6)
207	(0x00CF)	Profile Name (characters 7 and 8)
208	(0x00D0)	Profile Name (characters 9 and 10)
w		32 – 126 (high byte)
		32 – 126 (low byte)

These parameters hold the profile name up to 10 characters in length. The decimal values are representative of the standard ASCII character set for printable characters. The characters of the profile name are stored in sequence through the registers from low byte to high byte starting with register 204.

Example: Set the profile name to JESD22A.

First, convert the ASCII characters to their equivalent hexadecimal values:

J = x4A E = x45 S = x53 D = x44 2 = x32 2 = x32 A = x41

Note that null characters should not be used. The remaining three characters should thus be set as white spaces (x20). Placing the values in order from low byte to high byte yields register values of:

204 = x454A
 205 = x4453
 206 = x3232
 207 = x2041
 208 = x2020

209	(0x00D1)	Total Number of Steps in profile
w		1 – 99

This value must be set to the total number of operating steps in the profile. It indicates the number of steps that the EZT is to expect from a remote PC when a profile is sent. If this value is not equal to the number of steps sent, the profile download will not take place properly.

210	(0x00D2)	Guaranteed Soak Band Loop 1
211	(0x00D3)	Guaranteed Soak Band Loop 2
212	(0x00D4)	Guaranteed Soak Band Loop 3
213	(0x00D5)	Guaranteed Soak Band Loop 4
214	(0x00D6)	Guaranteed Soak Band Loop 5
w		-32768 – 32767 (-3276.8 – 3276.7)

These parameters contain an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the values contained in these registers must be divided by 10 to get the actual values. Likewise, when sending the setpoints, the setpoints must be multiplied by 10 in order for it to be set properly in the EZT-560i.

The following 1485 registers of the profile download contain specific values related to each operating step of the profile. When sending a profile to the EZT, transmit only the number of steps as set in register 209.

215 (0x00D7) Profile Step 1 Time (hours)
w 0 - 999

216 (0x00D8) Profile Step 1 Time (MM/SS)
w 0 - 59 Minutes (high byte)
0 - 59 Seconds (low byte)

This parameter contains both minutes and seconds for the profile step duration. The value will be an integer value based on the combined bytes of both the minutes and seconds. In order to set the value, combine the two component bytes into the word.

Example: Set a time of 1 minute, 30 seconds:

Converting the two decimal values into their hexadecimal equivalents yields 0x01 (1 minute) for the high byte and 0x1E (30 seconds) for the low byte. The resultant value to be written to the EZT is the combined bytes: x001E.

217 (0x00D9) Profile Step 1 Chamber Events
218 (0x00DA) Profile Step 1 Customer Events
w

- Bit0 Event 1
- Bit1 Event 2
- Bit2 Event 3
- Bit3 Event 4
- Bit4 Event 5
- Bit5 Event 6
- Bit6 Event 7
- Bit7 Event 8
- Bit8 Event 9
- Bit9 Event 10
- Bit10 Event 11
- Bit11 Event 12
- Bit12 Event 13
- Bit13 Event 14
- Bit14 Event 15

These parameters are bit oriented, i.e., enabling the different bits of the word turns on (1) or turns off (0) the event. Note that if an event is for controlling an option not available on your chamber, the associated bit should be set to zero.

219 (0x00DB) Profile Step 1 Guaranteed Soak / Wait for Digital Input Events

w	Bit0	Guaranteed Soak Loop 1
	Bit1	Guaranteed Soak Loop 2
	Bit2	Guaranteed Soak Loop 3
	Bit3	Guaranteed Soak Loop 4
	Bit4	Guaranteed Soak Loop 5
	Bit5	Digital Input 1 Wait For
	Bit6	Digital Input 2 Wait For
	Bit7	Digital Input 3 Wait For
	Bit8	Digital Input 4 Wait For
	Bit9	Digital Input 5 Wait For
	Bit10	Digital Input 6 Wait For
	Bit11	Digital Input 7 Wait For
	Bit12	Digital Input 8 Wait For

This parameter is bit oriented, i.e., enabling the different bits of the word turns on (1) or turns off (0) the event. Note that if an event is for controlling an option not available on your chamber, the associated bit should be set to zero.


Multiple guaranteed soak events can be enabled at a time; however, only one “digital input wait for” should be enabled at time. The guaranteed soak and “wait for” events can be used concurrently on the same step.

220 (0x00DC) Profile Step 1 Wait For Loop Events

w	0	Wait for Disabled (no loop selected)
	1	Loop 1 Selected
	2	Loop 2 Selected
	4	Loop 3 Selected
	8	Loop 4 Selected
	16	Loop 5 Selected

221 (0x00DD) Profile Step 1 Wait For Monitor Events

w	0	Wait for Disabled (no input selected)
	1	Monitor Input 1 Selected
	2	Monitor Input 2 Selected
	4	Monitor Input 3 Selected
	8	Monitor Input 4 Selected
	16	Monitor Input 5 Selected
	32	Monitor Input 6 Selected
	64	Monitor Input 7 Selected
	128	Monitor Input 8 Selected

 Only one “wait for” can be enabled per step. Do not enable more than one at a time. If multiple wait for conditions are desired, they must be enabled on sequential steps. The profile may not continue properly if more than one “wait for” is enabled on a step.

222 (0x00DE) Profile Step 1 Wait For Setpoint
w -32768 – 32767 (-3276.8 – 3276.7)

This parameter contains an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the values contained in these registers must be divided by 10 to get the actual values. Likewise, when sending the setpoints, the setpoints must be multiplied by 10 in order for it to be set properly in the EZT-560i.

223 (0x00DF) Profile Step 1 Jump Step Number
w 1 – 99

224 (0x00E0) Profile Step 1 Jump Count
w 0 – 999

A jump count greater than zero will initiate a jump to the step number entered in the jump step field when the current step is complete. To disable the jump, set a value of zero to this field. At the end of the step, the profile will then continue to the next step in sequence.

225 (0x00E1) Profile Step 1 Target Setpoint for Loop 1
226 (0x00E2) Profile Step 1 Target Setpoint for Loop 2
227 (0x00E3) Profile Step 1 Target Setpoint for Loop 3
228 (0x00E4) Profile Step 1 Target Setpoint for Loop 4
229 (0x00E5) Profile Step 1 Target Setpoint for Loop 5
w -32768 – 32767 (-3276.8 – 3276.7)

These parameters contain an assumed decimal point. Since only whole numbers can be sent to and received from the EZT-560i using Modbus protocol, the values contained in these registers must be divided by 10 to get the actual values. Likewise, when sending the setpoints, the setpoints must be multiplied by 10 in order for it to be set properly in the EZT-560i.

All remaining steps of the profile follow the same format and data structure as in step 1. By storing profiles as a 2-dimensional array, each row of the file can be read, and then transmitted to the EZT one step at a time in sequential order of 15 registers for each step.

230 (0x00E6) – 244 (0x00F4)	Profile Step 2 Data Registers
245 (0x00F5) – 259 (0x0103)	Profile Step 3 Data Registers
260 (0x0104) – 274 (0x0112)	Profile Step 4 Data Registers
275 (0x0113) – 289 (0x0121)	Profile Step 5 Data Registers
290 (0x0122) – 304 (0x0130)	Profile Step 6 Data Registers
305 (0x0131) – 319 (0x013F)	Profile Step 7 Data Registers
320 (0x0140) – 334 (0x014E)	Profile Step 8 Data Registers
335 (0x014F) – 349 (0x015D)	Profile Step 9 Data Registers

through	

1685 (0x0695) – 1699 (0x06A3)	Profile Step 99 Data Registers

2.5.1 Profile Download Algorithm

An example download profile procedure as well as the timer procedure is provided below in VB format. These are sample procedures and CSZ does not take responsibility for end user actions. The user must pay close attention regarding ranges and user interaction during profile download.

```
Private Sub downloadProfile()
'This sub will initiate the profile download and start the timer for profile download. Timer should be set at 1 second intervals
'to make sure timeout from EZT controller does not occur.
'-----
'Variables used in procedure. User can select static, form, global variables or class type based on preference.
G_vary(x,x) is array for each EZT in system. First element is EZT number, second element is EZT register. EZT register
are a total of 180 registers per EZT.
f_curProfileName is form var that holds currently loaded profile.
f_profRegsToWrite is form var that holds the current profile array point being written when timer ticks are active.
f_writeNum is form var that holds the current segment being written when timer ticks are active.
'-----

'User error checking for sub should go here

'Check to see if profile is running before downloading profile (see EZT register list for profile status register)

'if EZT is offline mode or downloading a profile do not start profile download from PC and alert user
If G_vAry(f_EztNum, 180) = 1 Then
    MsgBox "Local EZT-560 is in the offline mode or running/downloading a profile!" & vbCrLf & _
        "Writes from the PC can not be initiated at this time.", _
        vbInformation
    Exit Sub
End If

'Before download, make sure to save the profile name to disk as well as to array registers within profile list.
'EZT manual outlines profile register list. Profile name is 5 registers (10 chars total) and must be converted from
'text to integer values (2 chars per register)
If f_curProfileName = "" Then
    MsgBoxOver "You must load or save a Profile before download to processor!", vbInformation, "Download Profile", Me.hWnd
    Exit Sub
End If
'following should disable graphic user interface for profile download while profile is being downloaded. User dependent code but
'user should not press download button again during download of profile from PC to EZT controller.
f_profRegsToWrite = 0 'set var to zero each time profile download is initiated.
f_writeNum = 0 'set var to zero each time profile download is initiated.
f_regAdd = 201 'registers for profile data start at register 200 in HMI. Modbus master dependent based on zero based addressing.
End Sub
```

```

Private Sub downloadTimer_Timer()
'timer code that is initiated by downloadProfile procedure. Timer should be set at a minimum of 1 second intervals.
'write profile data array to PLC
-----
'variables used in procedure. User can select static, form, global variables or class type based on preference.
Dim x As Integer, writeValue As Integer 'write values to EZT must be integer values.
profileData(x,x) is array that holds profile data for currently active profile that is being edited (14,99) elements zero based.
'refer to downloadProfile procedure for form or global variables used.
-----
'Error checking should go here

f_profWrite = True 'profile write in progress flag form var. Not required - user dependent to block other actions during download
If f_writeNum < (totalNumSegs) Then 'totalNumSegs should be var or reference to total number of segs created for profile.

'for all element in profile array (15 elements total per segment. zero based)
For x = 0 To 14 '14 total elements in profile
  If f_profRegsToWrite = 0 Then 'first element that hold data global to complete profile
    Select Case x
      Case 0 To 9 'autostart and profile name requires no scaling in element 0
        writeValue = CInt(profileData(x, f_profRegsToWrite))
      Case 10 To 14 'gs soak band requires scale by 10
        'values are for SP's so scale by 10 for PLC
        writeValue = CInt(profileData(x, f_profRegsToWrite) * 10)
    End Select
  Else 'elements 1 to 99 of profarray which holds data for each segment
    Select Case x
      Case 0 To 6
        'hours, mins,/secs, chamber events, cust events, gs events
        'wait or loop input, wait for monitor input.
        'first 6 elements require no scale by 10
        writeValue = CInt(profileData(x, f_profRegsToWrite))
      Case 7 'wait for input setpoint. value for SP so scale by 10
        writeValue = CInt(profileData(x, f_profRegsToWrite) * 10)
      Case 8, 9 'jump step and jump count are integers, dont scale by 10
        writeValue = CInt(profileData(x, f_profRegsToWrite))
      Case 10 To 14 'setpoint values for loop1 - 5. scale by 10
        writeValue = CInt(profileData(x, f_profRegsToWrite) * 10)
    End Select
  End If
  ModProf.WordVal(x) = writeValue 'modbus master array. code line is modbus server dependent for multi-writes.
Next
ModProf.Address = f_regAdd 'start register for profile download. code is modbus server dependent
ModProf.Size = 15 'size of array. code is modbus server dependent
ModProf.Trigger 'trigger comms write. command is modbus master dependent
'increment writes for end of loop when total writes equal
'total number of segments in profile.
f_writeNum = f_writeNum + 1
f_profRegsToWrite = f_profRegsToWrite + 1 'increment registers within profile for write
f_regAdd = f_regAdd + 15 'increment to next 15 profile registers
Else
'profile writes are complete. Segments written = total number of segments created for profile.
downloadTimerer2.Enabled = False 'stop download timer
'hide progress bar and download messages. Unlock screen if user locked to block other actions here.
f_profWrite = False 'profile write completed reset flag if used.
End If
End Sub

```

2.5.2 Sending a Profile to the EZT-560i

Profiles are sent to the EZT in a step-by-step process. The download sequence must be followed in order and must complete without errors to be valid. If a write error is detected during the transfer of a profile from a PC to the EZT (no response from EZT or NACK returned), the profile download must be aborted and restarted.

The EZT-560i is put into profile transfer mode when the first group of registers containing the profile specific data is sent (registers 200-214). The EZT then begins looking for the number of steps of the profile to be sent as was set in register 209. As each step is received, it increments the count. Once all steps have been received, the EZT downloads the profile into the profiler memory. During this transfer, register 180 will be set to 1 to indicate that the process is taking place. Once the register value returns to zero, the profile is ready to be started.

If an error occurs during the transfer process from the PC to the EZT, the profile transfer process should be stopped at the PC. The data sent to the EZT was either corrupted in transmission or not received properly. It is not possible to resend the failed step because it is not known if any of the previous data was received by the EZT properly. On the transmission error, the EZT will enter a 15 second timeout process. At the end of the timeout period, the buffer will be cleared and the profile can be resent. In order to insure that the new download begins properly, induce a 20 second wait period on the host PC after the failed transmission attempt to insure that enough time has elapsed.

Modbus Timeout

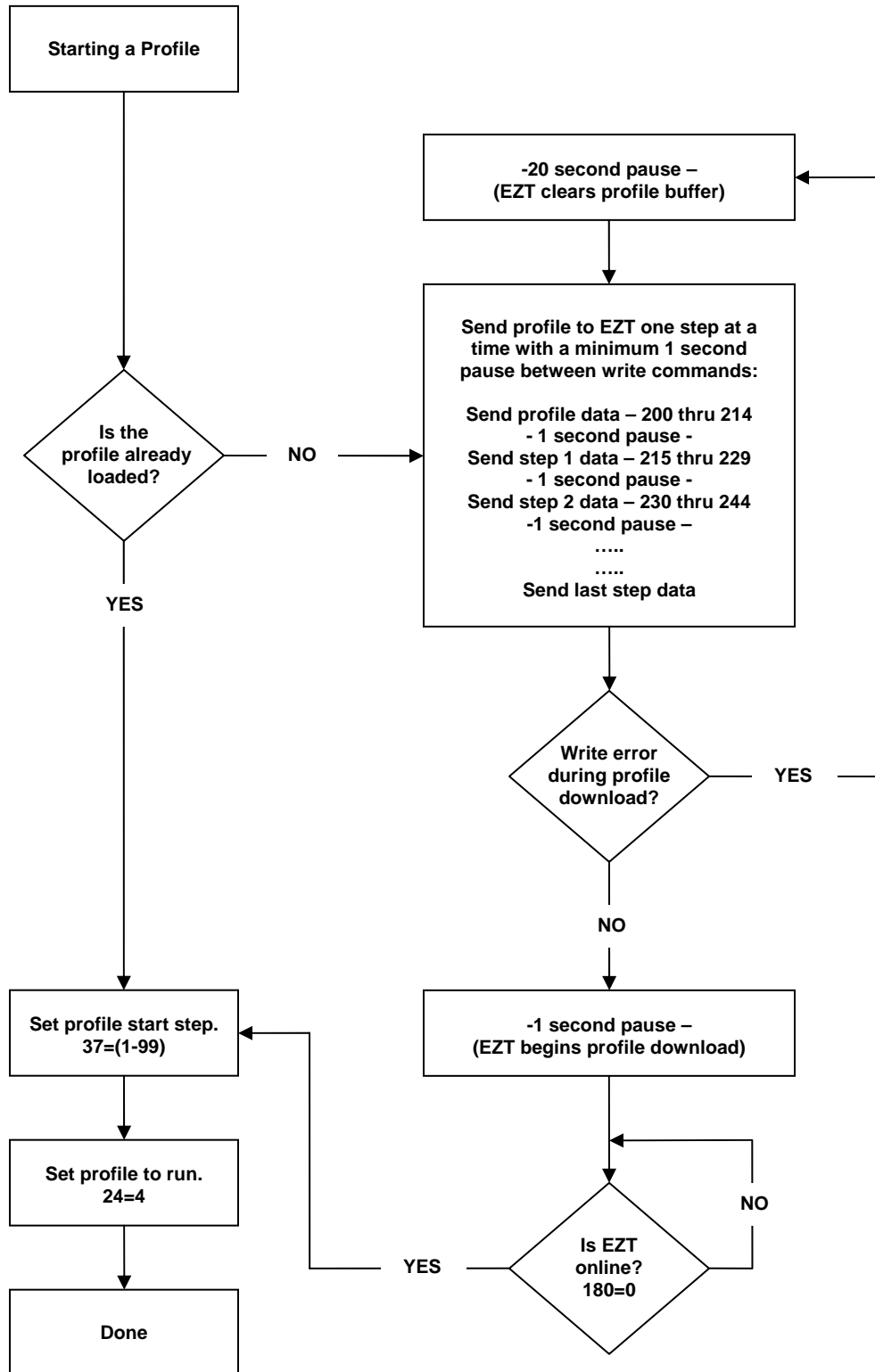
The Modbus timeout setting on the Web Server/Modbus/VNC communication setup screen is provided as a means of adjusting the EZT's internal communication buffer to help in correcting write errors during the profile transfer process from a PC.

The EZT's processor supports the GUI, data logging processes, internal controller communications, VNC and web server connections as well as communications to the PC. Depending upon the current work load on the EZT's based on operating processes, the EZT may not properly process the entire step data message during the transfer process before its own serial port buffer times out.

By increasing the timeout setting, it will provide more time for the EZT to process the command while it is cycling through the other multi-threaded processes. If frequent write errors are occurring between the EZT and the PC, try increasing this value slightly in order to minimize the transmission failures. Note that this should not be used to correct issues relating to poor connections or wiring practices. Good wiring practices should always be followed when making communication connections between all devices.

It is important to note that increasing the timeout period too much will also slow down overall performance when multiple chambers are connected on a single RS485 link. Each EZT on the link will see the response of other EZT's as a message. Since it is not a valid command for them, they will initiate the timeout to clear their buffer before they can accept another valid message. Thus, once a response from an EZT is received, the host computer must wait a minimum of the longest timeout setting prior to sending another message to another EZT on the link. Otherwise, the EZT will not receive the command properly and will not respond.

2.5.3 Starting a Profile in the EZT-560i



Appendix

Common Terms and Definitions

address – A unique designator for a location of data or a controller that allows each location or controller on a single communications bus to respond to its own message.

ASCII (pronounced AS-KEY) – American Standard Code for Information Interchange. A universal standard for encoding alphanumeric characters into 7 or 8 binary bits.

Asynchronous – Communications where characters can be transmitted at an unsynchronized point in time. In other words, it can start and stop anytime. The time between transmitted characters may be of varying lengths. Communication is controlled by “start” and “stop” bits at the beginning and end of each character.

Baud – Unit of signaling speed derived from the number of events per second (i.e., bits per second).

Baud rate – The rate of information transfer in serial communications, measured in bits per second.

Binary – Number based system where only two characters exist, 0 and 1. Counting is 0, 1, 10, 11...

Bit – Derived from “B I nary digi T”, a one or zero condition in the binary system.

Byte – A term referring to eight associated bits of information, sometimes called a “character”.

Character – Letter, numeral, punctuation, control figure or any other symbol contained in a message. Typically this is encoded in one byte.

Communications – The use of digital computer messages to link components. (See serial communications and baud rate)

Converter – This device will convert from one hardware interface to another such as from EIA-232 to EIA-485. The converter may be transparent to the software, which means you do not have to give any special considerations to software programming.

CRC – When data is corrupted during transmission, a method is used to return the data to its correct value. This can be accomplished through several methods: parity, checksum and CRC (cyclic redundancy checksum) are three of these. Cyclic R edundancy C hecksum is an error-checking mechanism using a polynomial algorithm based on the content of a message frame at the transmitter and included in a field appended to the frame. At the receiver, it is then compared with the results of the calculation that is performed by the receiver.

Data – The information that is transferred across the communications bus. This may be a setpoint, setup parameter, or any character. This information is transferred to an address or register.

DB-9 – A standardized connector shaped like the letter “D” when viewed on edge. This connector has 9 contacts. It is utilized on most IBM AT compatible PCs as the serial port.

DB-15 – A standardized connector shaped like the letter “D” when viewed on edge. This connector has 15 contacts. It is utilized on most IBM AT compatible PCs as the game/midi port.

DB-25 – A standardized connector shaped like the letter “D” when viewed on edge. This connector has 25 contacts. It is utilized on most IBM AT compatible PC's as the parallel port when the PC end contains socket contacts. Can also be the serial port when the PC end contains pin contacts.

Common Terms and Definitions (cont'd)

Decode – This is the reverse of encode. When a piece of data has information embedded in it, decode is to extract that information. Example: to extract an “A” from 01000001.

Duplex – The ability to send and receive data at the same time. “To listen and talk at the same time.”

EIA-232 – Electronic Industries Association developed this standard hardware interface to allow one device to talk to another device in full duplex mode. This method uses a differential voltage between one wire and ground. Also called an unbalanced system since the ground wire carries the sum of current of all lines. Transmission is limited to about 50 feet.

EIA-485 – Electronic Industries Association developed this standard hardware interface to allow up to 32 devices to be on a bus at one time. This method uses a differential voltage between two wires. Also called a balanced system since each wire carries the same current value. This has the advantage of being immune to outside electrical disturbances.

EIA/TIA -232 and -485 – Data communications standards set by the Electronic Industries Association and Telecommunications Industry Association. Formerly referred to as RS- (Recommended Standard). (See EIA-232 and EIA-485)

Electronic Industries Association (EIA) – An association in the US that establishes standards for electronics and data communications.

Encode – To embed information into a piece of data. This is the reverse of decode. Example: let 01000001 stand for an “A”.

Error Correction – When an inconsistency is in the data, a method is used to detect and/or return the data to its correct value. This can be done through several methods, parity, checksum and CRC (cyclic redundancy checksum) area three of these.

Even – This term is used with parity. See parity.

Firmware – Instruction or data stored in an IC (integrated circuit) or on a read only disk. This data is programmed once and cannot easily be changed as software can.

Full Duplex – Full is used to mean the duplex's full capability. The ability to send and receive data at the same time. The same as duplex.

GPIB – See IEEE488

Half Duplex – The ability to send or receive data, but not at the same time. “To listen or talk, but not both at the same time.”

Handshake (Handshaking) – Exchange of predetermined signals between two devices establishing a connection. Using extra wires or software signals to coordinate communications, signals can be sent to tell the transmitter the current status of the other device receiver. Example: Are you busy or are you ready?

Hex or Hexadecimal – Number based system where sixteen characters exist, 0 to 9, A to F. Counting is 0..9,A,B,C...

HMI – Human to Machine Interface typically performed in software on a personal computer. Also called MMI.

Common Terms and Definitions (cont'd)

IEEE488 – Bus developed by Hewlett-Packard in 1965 as HP-IB. Also referred to as GPIB (General Purpose Interface Bus). Consists of 8 data lines and 8 control lines. Bus length limited to 20.0 meters. Supports 15 devices on the bus at one time.

Logic Level – A voltage measurement system where only two stable voltage values exist. Example: 0v and 5V, or -3v and +3v.

Mark – Represents the transmission of data bit logic 1 (see logic level). Usually this is the most negative voltage value in serial communications.

Master – The device on the bus that controls all communications. Only the master can initiate conversation.

Modbus – A software protocol developed by Gould Modicon (now AEG) for process control systems. No hardware interface is defined. Modbus is accessed on the master/slave principle, the protocol providing for one master and up to 247 slaves. Only the master can initiate a transaction. This is a half duplex protocol.

MMI – Man to Machine Interface typically performed in software on a personal computer. Also called HMI.

Network – When two or more devices share communication lines, the devices are “networked”.

Node – A point of interconnection to a network.

Noise Immunity – The ability of communication lines to ignore electrical noise generated in the lines by nearby magnetic and electrostatic fields.

Odd – This term is used with parity. See parity.

Parallel – Communication using this method, transfers eight bits or one byte at a time over eight data wires and one ground wire. This method is eight times faster than using serial but utilizes more hardware.

Parity – A bit is assigned at the beginning of a byte to stand for parity. When the ‘1’ bits are counted, the number will be even or odd. A parity bit is used to ensure that the answer is always even if even parity or odd if odd parity. If the receiving end counts the ‘1’ bits and the sum is not the same odd or even, an error is generated. Parity is used to detect errors caused by noise in data transmission.

Protocol – A set of rules for communication. This will specify what method to transfer information, packet size, information headers and who should talk when. It is used to coordinate communication activity.

Receive – To accept data sent from another device. The device that receives the data is the receiver.

Register – An area of memory that provides temporary storage of digital data.

RJ11 – A connector used on most telephones that has four terminals.

Common Terms and Definitions (cont'd)

Slave – A device that only responds to commands. This device never starts communication on its own. Only the Master can do this. (See Master)

SCADA – Supervisory Control and Data Acquisition

Serial – To process something in order. First item, second item, etc.

Serial Communications – A method of transmitting information between devices by sending all bits serially (see serial) over a single communication channel.

Software – Information of data or program stored in an easily changeable format. (RAM, Floppy Disk, Hard Disk)

Space – Represents the transmission of a data bit logic 0 (see logic level). Usually this is the most positive voltage value in serial communications.

Start Bit – A binary bit or logic level that represents when the serial data information is about to start (at the beginning of a character or byte). This voltage level is positive.

Stop Bit – A binary bit or logic level that represents when the serial data information is complete (at the end of a character or byte). This voltage level is negative.

Synchronous – When data is transmitted on a data line and a clock signal is used on another line to determine when to check the data line for a logic level. This clock is said to “synchronize” the data.

Transmit – To send data from one device to another. The device that sends the data is the transmitter.

Word – Two bytes make a word. This contains 16 bits.