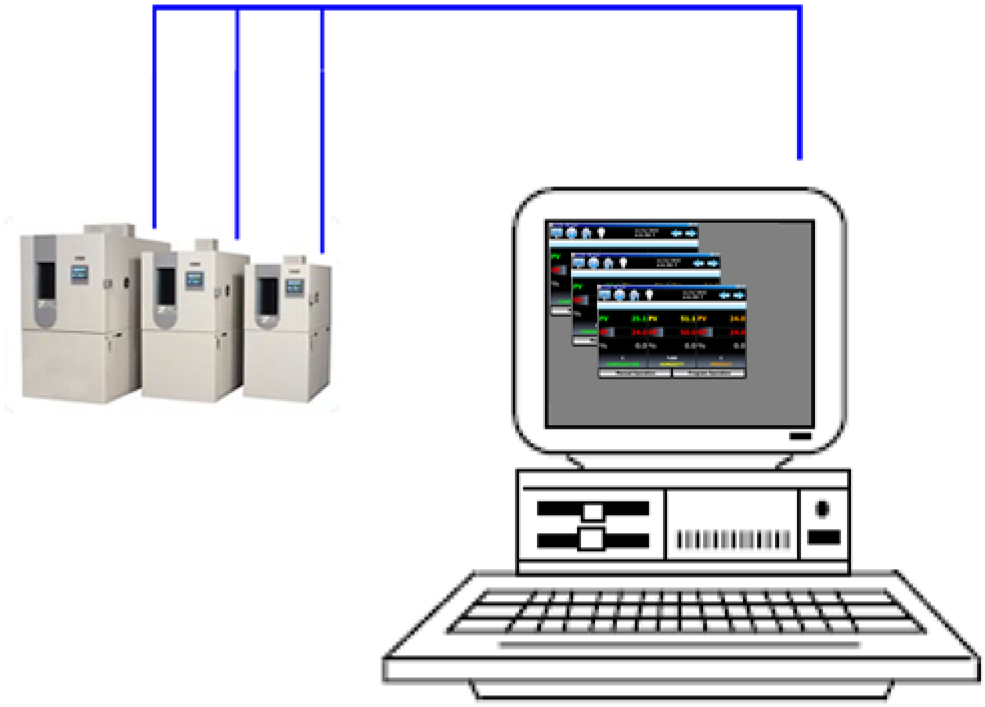
Cincinnati Sub-Zero

EZT-570S

User Communication Reference Manual



EZT570S User Communication Reference Manual RevC.docC
(450 V1.2 / 730 V1.0)

TABLE OF CONTENTS

1. Introduction

This document is targeted towards new users interested in using data communications with CSZ EZT-570S 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-570S 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-570S 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 ‘	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-570S 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. Baud refers to the 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-570S. *Thus a device used to communicate with the EZT-570S 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-570S. *A device used to communicate with the EZT-570S 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 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 was in effect and the binary number 1101 0110 was 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-570S can be set for even, odd or no parity. A device used to communicate with the EZT-570S must also have its serial port set to use the same parity setting as the EZT-570S 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-570S 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-570S 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-570S 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-570S 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 a data 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-570S 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-570S 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-570S'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-570S 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 content 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-570S 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-570S has its parity set to even as default from the factory. If a different parity setting is desired, just set the EZT-570S 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 28 bits (3.5 characters). To determine this time in seconds, divide 28 by the baud rate. In the case of EZT-570S communications at 9600 baud, this calculates to a minimum period of 3ms.

4. Values containing decimal points such as process values and set points, 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 set point of 42.4 degrees must be sent as a value of 424 in order for the EZT-570S to be set correctly. Likewise, a process value read from the EZT-570S 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 provided 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-570S 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 software and determine whether to resend the message or halt for operator intervention.

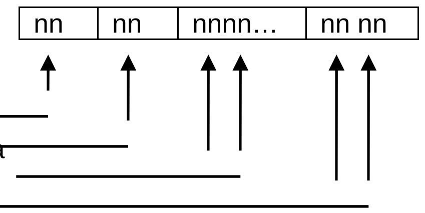
User Responsibility

Refrain from altering prompts that do not appear on the EZT-570S 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.

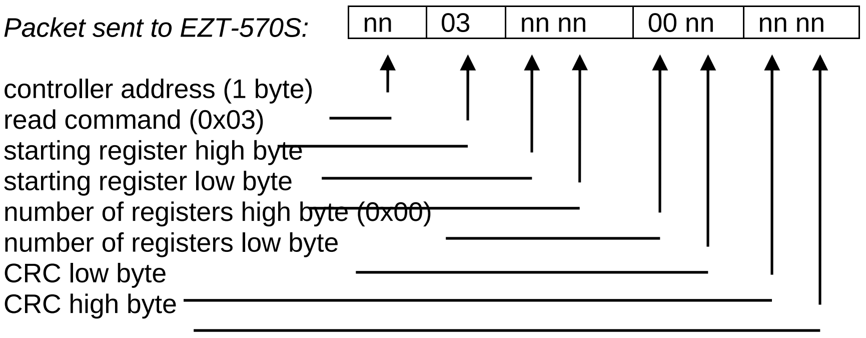
Packet format:



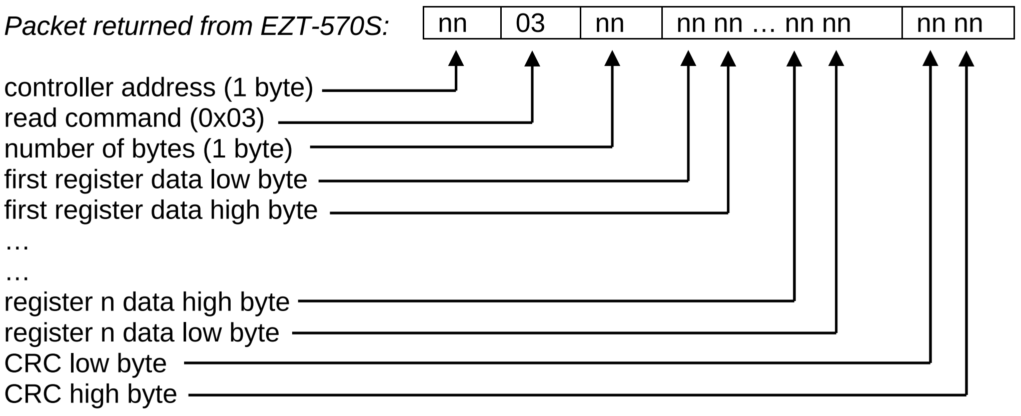
Read Register(s) Command (0x03)

This command returns from 1 to 60 registers. This command is to be used for reading one or more data locations from the EZT-570S.

Packet sent to EZT-570S:


nn
03
nn nn
00 nn
nn nn
controller address (1 byte)
read command (0x03)
starting register high byte
starting register low byte
number of registers high byte (0x00)
number of registers low byte
CRC low byte
CRC high byte

Packet returned from EZT-570S:



<u>*Example 1:*</u> 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 data: 236 (0x**00EC**) – temperature is 23.6 degrees

<u>*Example 2:*</u> Read registers 60 and 61 (chamber set point and temperature) of controller at address 1.

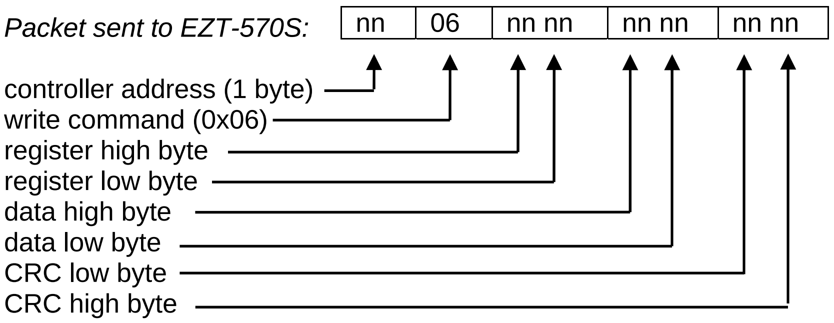
Sent: 01 03 00 3C 00 02 04 07
Received: 01 03 04 **01 90 01 48** FA 44

Message data: 400 (0x**0190**) and 328 (0x**0148**) – set point is 40.0 and temperature is 32.8 degrees

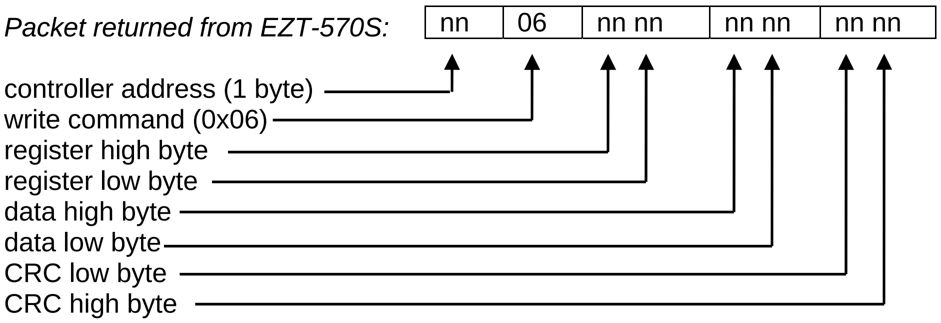
Write Register Command (0x06)

This command writes a value to a single register. This command is to be used for setting control values in the EZT-570S. To set multiple values, repeat the command for each data location.

Packet sent to EZT-570S:



Packet returned from EZT-570S:

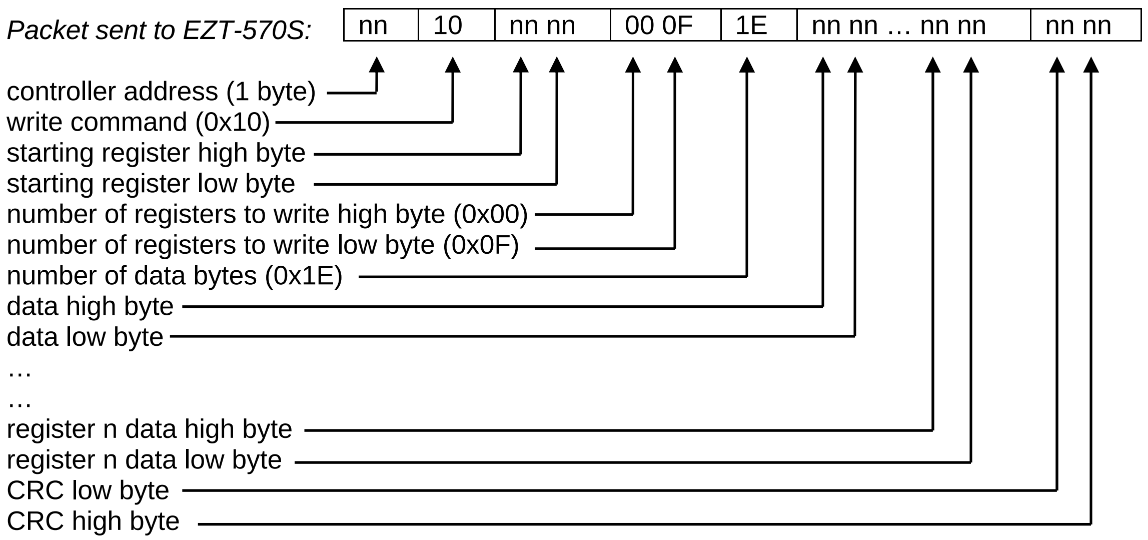


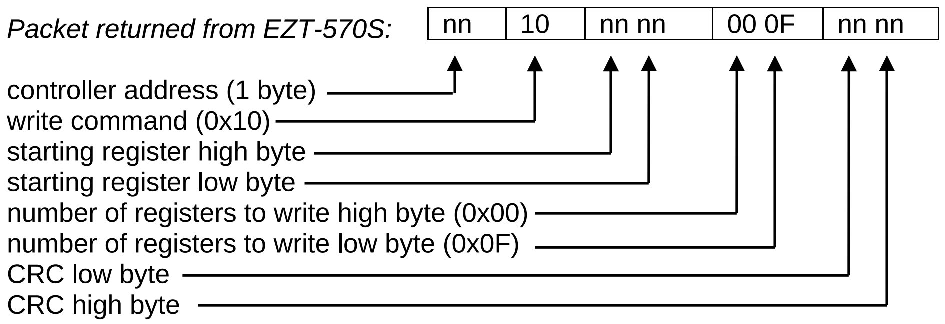
<u>*Example:*</u> Write register 60 (temperature set point) of controller at address one to 20 degrees (0x**00C8**).

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 automatic ramp/soak program download only. It is used to transmit program data one step at a time to the EZT-570S. See the Ramp/Soak Program Parameters section for the list of registers and their use. If this command is used to write to registers other than the correct program step registers, the EZT-570S will respond with an acknowledgement that the message was received; however, the command will not be executed.

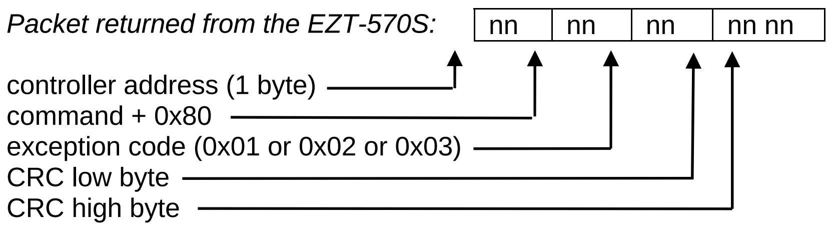

Packet sent to EZT-570S:
nn
10
nn nn
00 0F
1E
nn nn … nn nn
nn nn
controller address (1 byte)
write command (0x10)
starting register high byte
starting register low byte
number of registers to write high byte (0x00)
number of registers to write low byte (0x0F)
number of data bytes (0x1E)
data high byte
data low byte
…
…
register n data high byte
register n data low byte
CRC low byte
CRC high byte


Packet returned from EZT-570S:
nn
10
nn nn
00 0F
nn nn
controller address (1 byte)
write command (0x10)
starting register high byte
starting register low byte
number of registers to write high byte (0x00)
number of registers to write low byte (0x0F)
CRC low byte
CRC high byte

Exception Responses

When the EZT-570S cannot process a command, it returns an exception response and sets the high bit (0x80) of the command.

0x01	illegal command
0x02	illegal data address
0x03	illegal data value


Packet returned from the EZT-570S:
nn
nn
nn
nn nn
controller address (1 byte)
command + 0x80
exception code (0x01 or 0x02 or 0x03)
CRC low byte
CRC high byte

2.3.2 Error Checking

In Modbus communications, every message sent from the master (your software) receives a response from the slave (EZT-570S), including write commands. Thus, after each command sent, you must read the controller response before sending the next message. This provides the method of error checking in order to verify that the message you sent was received correctly, and that the controller is operating accordingly. This allows you to then determine the appropriate recovery response in case the message was not received correctly by the controller, and what action is to be taken by an operator and/or the software itself.

The exception responses provide a basic form of error checking. When an exception response is received, the code provided in the response will tell you what the error was in the sent message. However, this is only valid if the controller receives the message you sent, and there was an out-of-range value or simple transmission error in the message. It does not validate incomplete or failed transmissions. To insure that the data you receive from a read command is correct, and that the controller properly received a write command, you must parse the controller's response and validate the return message to insure it is correct.

In order to validate that the message you received is correct, you must calculate the CRC for the received message and compare it with the CRC that the controller appended to the message. This verifies that the data you received was what the EZT-570S sent. If the CRC's do not match, there was an error in the transmission and the entire message should be ignored. This could then be followed by an attempt to resend the failed command, or halt operation and alert an operator.

Example: Read registers 60 and 61 (loop 1 set point and process variable) of the EZT-570S at address 1.

Command sent to the EZT-570S:	01 03 00 3C 00 02 04 07
Message received from the EZT-570S:	01 03 04 03 0D 01 F3 2A 61

Calculated CRC:	2A61	(calculated from message 01 03 04 03 0D 01 F3)
Received CRC:	2A61	

The calculated CRC matches the received CRC, the message is valid. Note that the last two bytes of the received message are not used to calculate the CRC. The last two bytes are the CRC that the EZT-570S appended to the message. Do not include them when calculating the CRC.

2.3.3 Transmitting and Receiving Messages

In order to reliably communicate with the EZT-570S, it is important to develop an efficient means of transmitting and receiving messages. Modbus is a structured protocol and it must be properly followed. It is recommended, if possible, to locate an existing communication driver to incorporate into your software. Developing one from scratch can be challenging. However, if one is not available, or you choose to develop one yourself, the following guidelines may be of assistance.

Transmitting Messages

When sending a message to the EZT-570S, it is important to remember that Modbus RTU protocol does not have start-of-transmission or end-of-transmission characters. All messages are “framed” using timeouts between characters. A timeout between characters is a pause of at least 1.5 characters in length, and a timeout between frames is a pause of at least 3.5 characters in length. If either of these periods are exceeded while a message is being sent to the EZT-570S, it will discard the data it has received and wait for the first frame of the next valid communication.

At 9600 baud, the timeout between characters is a little over 1ms, and the EZT-570S will take any characters after a delay of as little as 3ms, as the beginning of a new message. This is an important consideration, because in creating your message, there are several steps that must be executed in order to build the packet and format the data properly into hexadecimal to send out the serial port of your PC. If you write code in a manner that steps byte by byte through sending the message out the serial port, formatting each piece of data prior to sending it, there is a good possibility that two much time may pass between characters, thus causing a failed transmission.

Therefore, it is recommended that the entire message, including the CRC, be created and assembled prior to being sent to the serial port. By assembling the main body of the message first, you can then pass it to the CRC algorithm which can step sequentially through the message, generate the CRC and append it to the message body. Once the message is completely assembled, it can then be sent out the serial port as a completed packet. This will insure that the message reaches the EZT-570S within the proper framing.

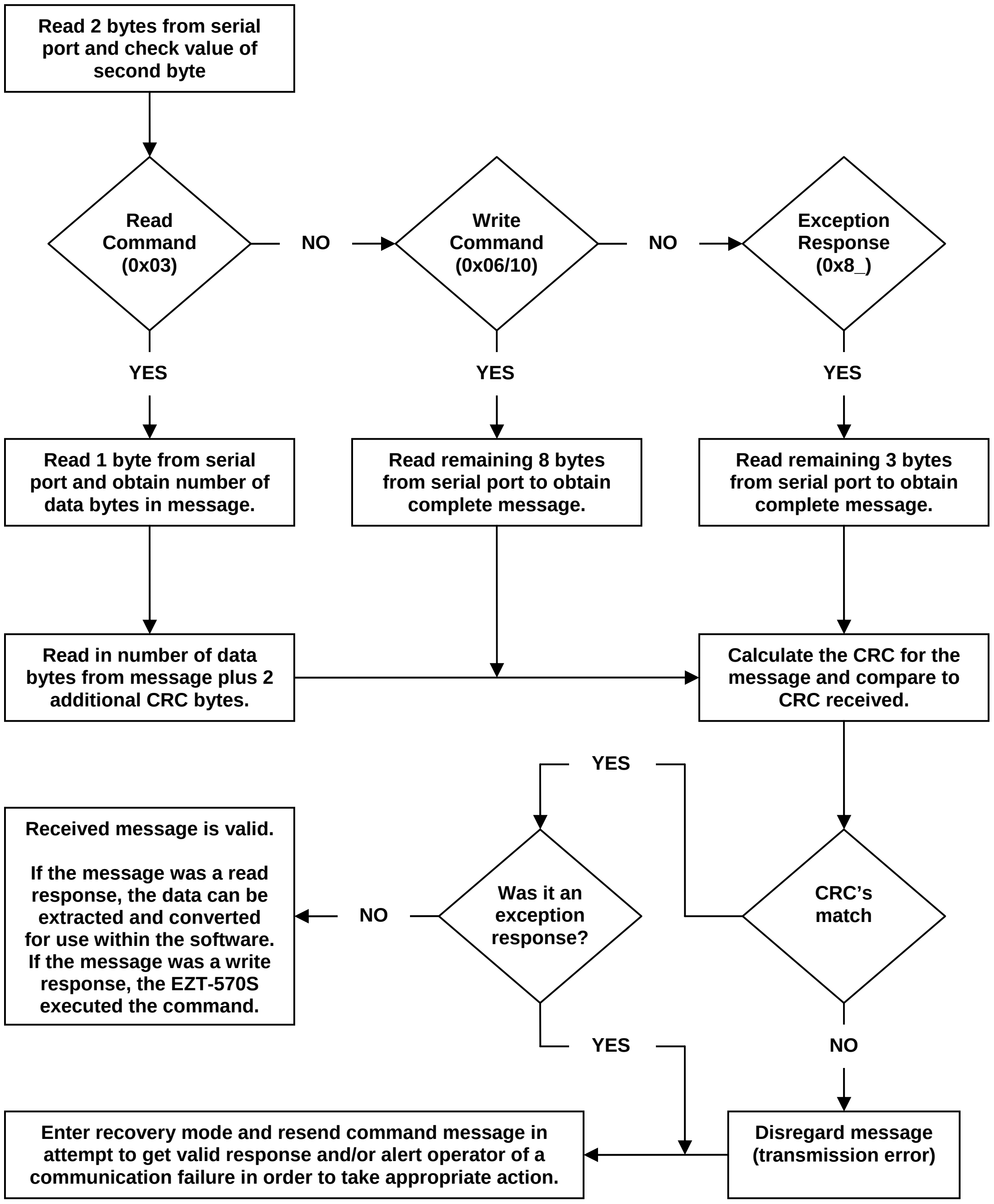
Receiving Messages

Due to the fact that Modbus RTU protocol does not have start-of-transmission or end-of-transmission characters, if the serial port driver you are using does not support an interval timeout setting allowing you to automatically terminate a read after a specified time passes between bytes (signaling the end of a message), you must know how long the message will be that you are receiving. That allows you to know how many bytes to read from your serial port and when you have received the entire message. If you rely on a maximum timeout period to terminate the read, depending upon the length of the received message, you will either loose a portion of the message or have to set the timeout period so high, that it will greatly affect the throughput of your code.

As can be seen from the previous examples for read and write commands in Section 2.3.1, the length of the returned message will vary based on the type of command, and for read commands, how many registers are being returned. Response messages can vary in length from as little as 5 bytes for an exception response to as many as 133 bytes for a read command. Therefore, in order to read in the message efficiently, you need to know what type of command it is in response to.

The response messages are always coded with the first two bytes of the message as the controller address and command type. When executing a read, read in only the first 2 bytes of data at the serial port. Examine the second byte and determine what the command is. If it is a write command (0x06 or 0x10), you know the response message is 8 bytes long. You can then read in the next 6 bytes of data from the serial port to complete the message. You can then calculate the CRC for the first 6 bytes of that message, and compare it to the last 2 bytes. If they match, then the communication completed successfully.

If the response is to a read command (0x03), you must then perform a single byte read from your serial port in order to get the next byte of the message. The third byte in a read response message is the number of data bytes in the message. By reading in this value, you then know how many data bytes follow. Note that this value does not include the 2 bytes for the CRC. Thus, when reading in the rest of the message, you will read in the number of data bytes plus an additional two, in order to get the CRC. You can then calculate the CRC for the message and compare it to the last two bytes. If they match, the data you received is valid.


Read 2 bytes from serial port and check value of second byte
Read Command (0x03)
NO
Write Command (0x06/10)
NO
Exception Response (0x8_)
YES
YES
YES
Read 1 byte from serial port and obtain number of data bytes in message.
Read remaining 8 bytes from serial port to obtain complete message.
Read remaining 3 bytes from serial port to obtain complete message.
Read in number of data bytes from message plus 2 additional CRC bytes.
Calculate the CRC for the message and compare to CRC received.
YES
Received message is valid.
If the message was a read response, the data can be extracted and converted for use within the software. If the message was a write response, the EZT-570S executed the command.
NO
Was it an exception response?
CRC's match
YES
NO
Enter recovery mode and resend command message in attempt to get valid response and/or alert operator of a communication failure in order to take appropriate action.
Disregard message (transmission error)

.

2.4 EZT-570S Control Registers

The EZT-570S 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-570S 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.

	TEMPERATURE/HUMIDITY MODELS		ALTITUDE MODELS		VTS/TSB MODELS	DTS MODELS
<u>LOOP</u>						
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-570S "Overview" screen.

	STANDARD MODELS (TMP/RH/ALTITUDE)	VTS/TSB MODELS	DTS MODELS
<u>EVENT</u>			
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

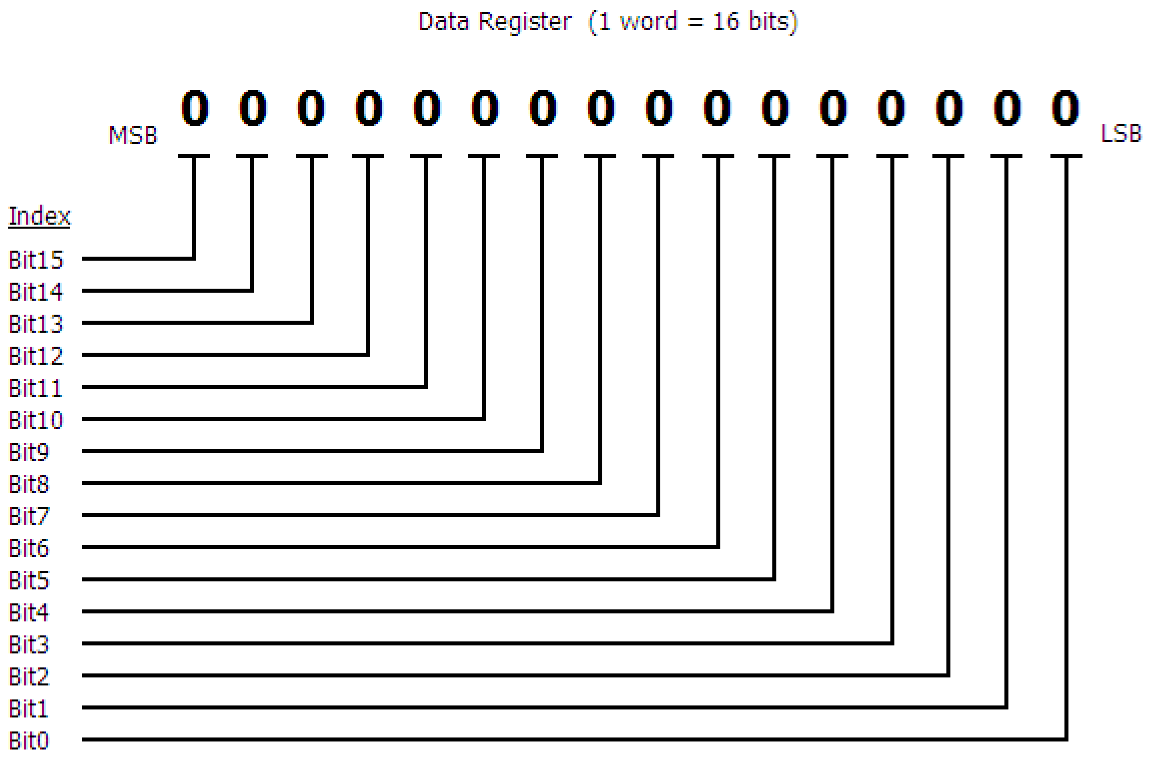
The control registers are grouped into three blocks of 60 (for a total of 180) registers relating to the specific types of data they contain. The first group of 60 registers (0 – 59) contains the configuration settings for various options on the EZT-570S as well as all of the alarm status, ramp/soak program 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 set point 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-570S'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-570S 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.

All register numbers listed are relative values. To convert to absolute values, add 400001.

Register Address		Parameter Description	Data *A Type	Range *B Low	High	*C Unit
0	**(0x0000)**	System Mode	R/W	*B1	*B1	-
1	**(0x0001)**	Clock (Year/Month)	R	*B2	*B2	-
2	**(0x0002)**	Clock (Day/DOW)	R	*B3	*B3	-
3	**(0x0003)**	Clock (Hours/Minutes)	R	*B4	*B4	-
4	**(0x0004)**	Clock (Seconds)	R	0	59	seconds
5	**(0x0005)**	Power Recovery Mode	R/W	*B5	*B5	-
6	**(0x0006)**	Power Recovery Time	R/W	0	32767	seconds
7	**(0x0007)**	Defrost Operating Mode	R/W	*B6	*B6	-
8	**(0x0008)**	Defrost Temperature Set point	R/W	-3276.8	3276.7	*C1
9	**(0x0009)**	Defrost Interval	R/W	0	32767	minutes
10	**(0x000A)**	Defrost Status	R	*B7	*B7	-
11	**(0x000B)**	Time Remaining Until Next Defrost	R	0	32767	minutes
12	**(0x000C)**	Product Control	R	*B8	*B8	-
13	**(0x000D)**	Product Control Upper Set Point	R/W	-3276.8	3276.7	*C1
14	**(0x000E)**	Product Control Lower Set Point	R/W	-3276.8	3276.7	*C1
15	**(0x000F)**	Condensation Control (Off/On)	R/W	0	1	-
16	**(0x0010)**	Condensation Control Monitor Mode	R/W	*B9	*B9	-
17	**(0x0011)**	Condensation Control Input Selection	R/W	*B10	*B10	-
18	**(0x0012)**	Condensation Control Ramp Rate Limit	R/W	*B11	*B11	-
19	**(0x0013)**	Condensation Control Dewpoint Limit	R	-3276.8	3276.7	*C1
20	**(0x0014)**	Condensation Control Actual Dewpoint	R	-3276.8	3276.7	*C1
21	**(0x0015)**	Chamber Light Control (Off/On)	R/W	0	1	-
22	**(0x0016)**	Chamber Events	R/W	*B12	*B12	-
23	**(0x0017)**	Customer Events	R/W	*B12	*B12	-
24	**(0x0018)**	Program Control/Status	R/W	*B13	*B13	-
25	**(0x0019)**	Program Advance Previous/Next Step	R/W	*B14	*B14	-
26	**(0x001A)**	Program Name Characters 1 & 2	R	*B15	*B15	-
27	**(0x001B)**	Program Name Characters 3 & 4	R	*B15	*B15	-
28	**(0x001C)**	Program Name Characters 5 & 6	R	*B15	*B15	-
29	**(0x001D)**	Program Name Characters 7 & 8	R	*B15	*B15	-
30	**(0x001E)**	Program Name Characters 9 & 10	R	*B15	*B15	-
31	**(0x001F)**	Year/Month Program Started	R	*B2	*B2	-

Register Address		Parameter Description	Data *A Type	Range *B Low	High	*C Unit
32	**(0x0020)**	Day/DOW Program Started	R	*B3	*B3	-
33	**(0x0021)**	Hour/Minute Program Started	R	*B4	*B4	-
34	**(0x0022)**	Year/Month Estimated Program End	R	*B2	*B2	-
35	**(0x0023)**	Day/DOW Estimated Program End	R	*B3	*B3	-
36	**(0x0024)**	Hour/Minute Estimated Program End	R	*B4	*B4	-
37	**(0x0025)**	Program Start Step Number	W	1	99	-
38	**(0x0026)**	Current Step of Program	R	1	99	-
39	**(0x0027)**	Last Step of Program	R	1	99	-
40	**(0x0028)**	Hours Left in Current Step	R	0	999	hours
41	**(0x0029)**	Minutes/Seconds Left in Current Step	R	*B16	*B16	-
42	**(0x002A)**	Program Wait Status	R	*B17	*B17	-
43	**(0x002B)**	Wait Set Point (Digital Input Number)	R	-3276.8	3276.7	-
44	**(0x002C)**	Current Step Jump Step Number	R	1	99	-
45	**(0x002D)**	Current Step Cycles Remaining	R	0	999	-
46	**(0x002E)**	Program Loop 1 Target Set Point	R	-3276.8	3276.7	*C1
47	**(0x002F)**	Program Loop 2 Target Set Point	R	-3276.8	3276.7	*C1
48	**(0x0030)**	Program Loop 3 Target Set Point	R	-3276.8	3276.7	*C1
49	**(0x0031)**	Program Loop 4 Target Set Point	R	-3276.8	3276.7	*C1
50	**(0x0032)**	Program Loop 5 Target Set Point	R	-3276.8	3276.7	*C1
51	**(0x0033)**	Last Jump Made from Step	R	1	99	-
52	**(0x0034)**	Last Jump Made to Step	R	1	99	-
53	**(0x0035)**	Total Jumps Made	R	0	32767	-
54	**(0x0036)**	Alarm Reset	W	*B18	*B18	-
55	**(0x0037)**	Input Alarm Status	R	*B19	*B19	-
56	**(0x0038)**	Loop/Monitor Alarm Status	R	* B20	*B20	-
57	**(0x0039)**	Chamber Critical Alarm Status	R	*B21	*B21	-
58	**(0x003A)**	Refrigeration Alarm Status	R	*B22	*B22	-
59	**(0x003B)**	System Status Monitor	R	*B23	*B23	-
60	**(0x003C)**	Loop 1 Set Point (SP)	R/W	-3276.8	3276.7	*C1
61	**(0x003D)**	Loop 1 Process value (PV)	R	-3276.8	3276.7	*C1
62	**(0x003E)**	Loop 1 Percentage of Output (%Out)	R	-100.00	100.00	*C1
63	**(0x003F)**	Loop 1 Autotune Status	R/W	*B24	*B24	-
64	**(0x0040)**	Loop 1 Upper Set Point Limit	R/W	-3276.8	3276.7	*C1
65	**(0x0041)**	Loop 1 Lower Set Point Limit	R/W	-3276.8	3276.7	*C1
66	**(0x0042)**	Loop 1 Alarm Type	R/W	*B25	*B25	-
67	**(0x0043)**	Loop 1 Alarm Modes	R/W	*B26	*B26	-
68	**(0x0044)**	Loop 1 Alarm Output Selection	R/W	*B27	*B27	-
69	**(0x0045)**	Loop 1 High Alarm Set Point	R/W	-3276.8	3276.7	*C1

Register Address		Parameter Description	Data *A Type	Range *B Low	High	*C Unit
70	**(0x0046)**	Loop 1 Low Alarm Set Point	R/W	-3276.8	3276.7	*C1
71	**(0x0047)**	Loop 1 Alarm Differential	R/W	0.0	3276.7	*C1
72	**(0x0048)**	Loop 2 Set Point (SP)	R/W	-3276.8	3276.7	*C1
73	**(0x0049)**	Loop 2 Process value (PV)	R	-3276.8	3276.7	*C1
74	**(0x004A)**	Loop 2 Percentage of Output (%Out)	R	-100.00	100.00	*C1
75	**(0x004B)**	Loop 2 Autotune Status	R/W	*B24	*B24	-
76	**(0x004C)**	Loop 2 Upper Set Point Limit	R/W	-3276.8	3276.7	*C1
77	**(0x004D)**	Loop 2 Lower Set Point Limit	R/W	-3276.8	3276.7	*C1
78	**(0x004E)**	Loop 2 Alarm Type	R/W	*B25	*B25	-
79	**(0x004F)**	Loop 2 Alarm Modes	R/W	*B26	*B26	-
80	**(0x0050)**	Loop 2 Alarm Output Selection	R/W	*B27	*B27	-
81	**(0x0051)**	Loop 2 High Alarm Set Point	R/W	-3276.8	3276.7	*C1
82	**(0x0052)**	Loop 2 Low Alarm Set Point	R/W	-3276.8	3276.7	*C1
83	**(0x0053)**	Loop 2 Alarm Differential	R/W	0.0	3276.7	*C1
84	**(0x0054)**	Loop 3 Set Point (SP)	R/W	-3276.8	3276.7	*C1
85	**(0x0055)**	Loop 3 Process value (PV)	R	-3276.8	3276.7	*C1
86	**(0x0056)**	Loop 3 Percentage of Output (%Out)	R	-100.00	100.00	*C1
87	**(0x0057)**	Loop 3 Autotune Status	R/W	*B24	*B24	-
88	**(0x0058)**	Loop 3 Upper Set Point Limit	R/W	-3276.8	3276.7	*C1
89	**(0x0059)**	Loop 3 Lower Set Point Limit	R/W	-3276.8	3276.7	*C1
90	**(0x005A)**	Loop 3 Alarm Type	R/W	*B25	*B25	-
91	**(0x005B)**	Loop 3 Alarm Modes	R/W	*B26	*B26	-
92	**(0x005C)**	Loop 3 Alarm Output Selection	R/W	*B27	*B27	-
93	**(0x005D)**	Loop 3 High Alarm Set Point	R/W	-3276.8	3276.7	*C1
94	**(0x005E)**	Loop 3 Low Alarm Set Point	R/W	-3276.8	3276.7	*C1
95	**(0x005F)**	Loop 3 Alarm Differential	R/W	0.0	3276.7	*C1
96	**(0x0060)**	Loop 4 Set Point (SP)	R/W	-3276.8	3276.7	*C1
97	**(0x0061)**	Loop 4 Process value (PV)	R	-3276.8	3276.7	*C1
98	**(0x0062)**	Loop 4 Percentage of Output (%Out)	R	-100.00	100.00	*C1
99	**(0x0063)**	Loop 4 Autotune Status	R/W	*B24	*B24	-
100	**(0x0064)**	Loop 4 Upper Set Point Limit	R/W	-3276.8	3276.7	*C1
101	**(0x0065)**	Loop 4 Lower Set Point Limit	R/W	-3276.8	3276.7	*C1
102	**(0x0066)**	Loop 4 Alarm Type	R/W	*B25	*B25	-
103	**(0x0067)**	Loop 4 Alarm Modes	R/W	*B26	*B26	-
104	**(0x0068)**	Loop 4 Alarm Output Selection	R/W	*B27	*B27	-
105	**(0x0069)**	Loop 4 High Alarm Set Point	R/W	-3276.8	3276.7	*C1
106	**(0x006A)**	Loop 4 Low Alarm Set Point	R/W	-3276.8	3276.7	*C1
107	**(0x006B)**	Loop 4 Alarm Differential	R/W	0.0	3276.7	*C1

Register Address		Parameter Description	Data *A Type	Range *B Low	High	*C Unit
108	(0x006C)	Loop 5 Set Point (SP)	R/W	-3276.8	3276.7	*C1
109	(0x006D)	Loop 5 Process value (PV)	R	-3276.8	3276.7	*C1
110	(0x006E)	Loop 5 Percentage of Output (%Out)	R	-100.00	100.00	*C1
111	(0x006F)	Loop 5 Autotune Status	R/W	*B24	*B24	-
112	(0x0070)	Loop 5 Upper Set Point Limit	R/W	-3276.8	3276.7	*C1
113	(0x0071)	Loop 5 Lower Set Point Limit	R/W	-3276.8	3276.7	*C1
114	(0x0072)	Loop 5 Alarm Type	R/W	*B25	*B25	-
115	(0x0073)	Loop 5 Alarm Modes	R/W	*B26	*B26	-
116	(0x0074)	Loop 5 Alarm Output Selection	R/W	*B27	*B27	-
117	(0x0075)	Loop 5 High Alarm Set Point	R/W	-3276.8	3276.7	*C1
118	(0x0076)	Loop 5 Low Alarm Set Point	R/W	-3276.8	3276.7	*C1
119	(0x0077)	Loop 5 Alarm Differential	R/W	0.0	3276.7	*C1
120	(0x0078)	Monitor 1 Process value (PV)	R	-3276.8	3276.7	*C1
121	(0x0079)	Monitor 1 Alarm Type	R/W	*B25	*B25	-
122	(0x007A)	Monitor 1 Alarm Modes	R/W	*B26	*B26	-
123	(0x007B)	Monitor 1 Alarm Output Selection	R/W	*B27	*B27	-
124	(0x007C)	Monitor 1 High Alarm Set Point	R/W	-3276.8	3276.7	*C1
125	(0x007D)	Monitor 1 Low Alarm Set Point	R/W	-3276.8	3276.7	*C1
126	(0x007E)	Monitor 1 Alarm Differential	R/W	0.0	3276.7	*C1
127	(0x007F)	Monitor 2 Process value (PV)	R	-3276.8	3276.7	*C1
128	(0x0080)	Monitor 2 Alarm Type	R/W	*B25	*B25	-
129	(0x0081)	Monitor 2 Alarm Modes	R/W	*B26	*B26	-
130	(0x0082)	Monitor 2 Alarm Output Selection	R/W	*B27	*B27	-
131	(0x0083)	Monitor 2 High Alarm Set Point	R/W	-3276.8	3276.7	*C1
132	(0x0084)	Monitor 2 Low Alarm Set Point	R/W	-3276.8	3276.7	*C1
133	(0x0085)	Monitor 2 Alarm Differential	R/W	0.0	3276.7	*C1
134	(0x0086)	Monitor 3 Process value (PV)	R	-3276.8	3276.7	*C1
135	(0x0087)	Monitor 3 Alarm Type	R/W	*B25	*B25	-
136	(0x0088)	Monitor 3 Alarm Modes	R/W	*B26	*B26	-
137	(0x0089)	Monitor 3 Alarm Output Selection	R/W	*B27	*B27	-
138	(0x008A)	Monitor 3 High Alarm Set Point	R/W	-3276.8	3276.7	*C1
139	(0x008B)	Monitor 3 Low Alarm Set Point	R/W	-3276.8	3276.7	*C1
140	(0x008C)	Monitor 3 Alarm Differential	R/W	0.0	3276.7	*C1
141	(0x008D)	Monitor 4 Process value (PV)	R	-3276.8	3276.7	*C1
142	(0x008E)	Monitor 4 Alarm Type	R/W	*B25	*B25	-
143	(0x008F)	Monitor 4 Alarm Modes	R/W	*B26	*B26	-
144	(0x0090)	Monitor 4 Alarm Output Selection	R/W	*B27	*B27	-
145	(0x0091)	Monitor 4 High Alarm Set Point	R/W	-3276.8	3276.7	*C1

Register Address		Parameter Description	Data *A Type	Range *B Low	High	*C Unit
146	**(0x0092)**	Monitor 4 Low Alarm Set Point	R/W	-3276.8	3276.7	*C1
147	**(0x0093)**	Monitor 4 Alarm Differential	R/W	0.0	3276.7	*C1
148	**(0x0094)**	Monitor 5 Process value (PV)	R	-3276.8	3276.7	*C1
149	**(0x0095)**	Monitor 5 Alarm Type	R/W	*B25	*B25	-
150	**(0x0096)**	Monitor 5 Alarm Modes	R/W	*B26	*B26	-
151	**(0x0097)**	Monitor 5 Alarm Output Selection	R/W	*B27	*B27	-
152	**(0x0098)**	Monitor 5 High Alarm Set Point	R/W	-3276.8	3276.7	*C1
153	**(0x0099)**	Monitor 5 Low Alarm Set Point	R/W	-3276.8	3276.7	*C1
154	**(0x009A)**	Monitor 5 Alarm Differential	R/W	0.0	3276.7	*C1
155	**(0x009B)**	Monitor 6 Process value (PV)	R	-3276.8	3276.7	*C1
156	**(0x009C)**	Monitor 6 Alarm Type	R/W	*B25	*B25	-
157	**(0x009D)**	Monitor 6 Alarm Modes	R/W	*B26	*B26	-
158	**(0x009E)**	Monitor 6 Alarm Output Selection	R/W	*B27	*B27	-
159	**(0x009F)**	Monitor 6 High Alarm Set Point	R/W	-3276.8	3276.7	*C1
160	**(0x00A0)**	Monitor 6 Low Alarm Set Point	R/W	-3276.8	3276.7	*C1
161	**(0x00A1)**	Monitor 6 Alarm Differential	R/W	0.0	3276.7	*C1
162	**(0x00A2)**	Monitor 7 Process value (PV)	R	-3276.8	3276.7	*C1
163	**(0x00A3)**	Monitor 7 Alarm Type	R/W	*B25	*B25	-
164	**(0x00A4)**	Monitor 7 Alarm Modes	R/W	*B26	*B26	-
165	**(0x00A5)**	Monitor 7 Alarm Output Selection	R/W	*B27	*B27	-
166	**(0x00A6)**	Monitor 7 High Alarm Set Point	R/W	-3276.8	3276.7	*C1
167	**(0x00A7)**	Monitor 7 Low Alarm Set Point	R/W	-3276.8	3276.7	*C1
168	**(0x00A8)**	Monitor 7 Alarm Differential	R/W	0.0	3276.7	*C1
169	**(0x00A9)**	Monitor 8 Process value (PV)	R	-3276.8	3276.7	*C1
170	**(0x00AA)**	Monitor 8 Alarm Type	R/W	*B25	*B25	-
171	**(0x00AB)**	Monitor 8 Alarm Modes	R/W	*B26	*B26	-
172	**(0x00AC)**	Monitor 8 Alarm Output Selection	R/W	*B27	*B27	-
173	**(0x00AD)**	Monitor 8 High Alarm Set Point	R/W	-3276.8	3276.7	*C1
174	**(0x00AE)**	Monitor 8 Low Alarm Set Point	R/W	-3276.8	3276.7	*C1
175	**(0x00AF)**	Monitor 8 Alarm Differential	R/W	0.0	3276.7	*C1
176	**(0x00B0)**					
177	**(0x00B1)**					
178	**(0x00B2)**					
179	**(0x00B3)**	Program Step Time Addition	W	0	32767	minutes
180	**(0x00B4)**	EZT-570S Program Download/Offline	R	*B28	*B28	-

Legend:

*A R/W Specifies readable / writable data, R specifies read only data and W specifies a write only control value.

*B The range of certain parameters is dependent upon system options. Consult the following range tables for information regarding the use of these parameters.

Reading bit oriented parameters
The value contained in these parameters is dependant upon the combination of "on" bits (1). Therefore, only the individual status of each bit has meaning, not the value of the parameter.

Setting bit oriented parameters
The value that must be written to these parameters is dependant upon the combination of "on" bits. Therefore, it is necessary to know the current value of the parameter before setting it so that only the bit status you want to update is changed. Otherwise, sending a value derived from only the bit you wish to set will turn off all other functions related to the other bits in the parameter.

*B1

Parameter Value	Description
Bit0	EZT-570S Online
Bit1 - Bit15	Not Assigned

DO NOT alter the state of this register. Bit0 is the system online bit and is set by the EZT-570S HMI when the unit is ready for operation. Turning off this bit will turn off the system.

The status of this register should be used for information only, as a means of determining if the system is ready for operation.

*B2

Parameter Value	Range Low	Range High	Description
High Byte	0	99	Year
Low Byte	1	12	Month

*B3

Parameter Value	Range Low	Range High	Description
High Byte	1	31	Day
Low Byte	0	6	Day of Week**

**The days of the week are represented as numbers:
0=Sun, 1=Mon, 2=Tue, 3=Wed, 4=Thu, 5=Fri, 6=Sat

*B4

Parameter Value	Range Low	Range High	Description
High Byte	0	23	Hour
Low Byte	0	59	Minutes

Example
Read of registers 1 to 4 for the current time from the EZT-570S returns the following values:

```
Register Values:       0x0A 0B     0x04 04     0x0A 1D     0x00 20
Decimal Equivalent:     10  11       4   4       10  29         32
```

Translating the values into an actual date and time provides a date and time of Thursday November 4, 2010 at 10:29:32 am.

*B5

Parameter Value	Description
0	Continue
1	Hold
2	Off
4	Start Over
8	Resume

*B6

Parameter Value	Description
0	Disabled
1	Manual
2	Auto

*B7

Parameter Value	Description
0	Inactive
1	In Defrost
2	In Prechill

*B8

Parameter Value	Description
0	Off
1	Deviation
2	Process
5	Deviation with Event Enable/Disable
6	Process with Event Enable/Disable

*B9

Parameter Value	Description
1	Single
2	Lowest
4	Highest
8	Average

*B10

Parameter Value	Description
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)
Bit9-15	Not Assigned

Setting the individual bits in the word enables (1) or disables (0) the use of the input for condensation control. Bits associated with inputs not available on your chamber should be set off.

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 off 227, the selected inputs will be used.

*B11

Parameter Value	Range Low	Range High
Degrees C	0	10.0
Degrees F	0	18.0

*B12

Parameter Value	Description
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
Bit15	Not Assigned

Setting the individual bits in the word turns on (1) or turns off (0) the event. If an event is for controlling an option that is not available on your chamber, the associated bit should be set off.

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.

*B13

Parameter Value	Description
0	Stop/Off
1	Stop/All Off
2	Hold
4	Run/Resume
8	Autostart**
16	Wait **
32	Ramp**
64	Soak**
128	Guaranteed Soak**

**These values are set by the EZT-570S to indicate the operating status of the program and can not be set directly.

*B14

Parameter Value	Description
1	Program Advance to Previous Step
2	Program Advance to Next Step

This parameter only performs its function when the program is in hold. Once the set function is executed, this parameter automatically resets to zero (0).

*B15

Parameter Value	High Order Byte	Low Order Byte	Description
Range Low	32	32	Program Name Character (ASCII Table)
Range High	126	126	Program Name Character (ASCII Table)

See the ASCII character chart in Section 3.1 for the character representation of these values.

Example
Read command of registers 26 to 30 from the EZT-570S returns the following values:

```
Register Values:    0x74 53    0x72 6F        0x20 65    0x65 54    0x74 73
ASCII Equivalent:     t  S        r  o             e        e  T       t  s
```

Assemble the ASCII characters in order from low to high byte starting with register 26 in order to assemble the Program name: “Store Test”. Note that null characters are not used in the Program name. A space (0x20) will be used is used in place of a null character to maintain the 10 character name length if the Program name is not ten characters long.

*B16

Parameter Value	Range Low	Range High	Description
High Byte	0	59	Minutes
Low Byte	0	59	Seconds

*B17

Parameter Value	Description
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

*B18

Parameter Value	Description
1	Alarm Reset (Silence)
2	Pumpdown Reset

Once the set function is executed, this parameter automatically resets to zero (0).

*B19

Parameter Value	Description
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	Loop Communications Failure
Bit15	Not Assigned

The individual bits of the word indicate specific alarm conditions. When the bit is on (1) the alarm is present. More than one alarm can be present at a time.

*B20

Parameter Value	Description
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
Bit13-15	Not Assigned

The individual bits of the word indicate specific alarm conditions. When the bit is on (1) the alarm is present. More than one alarm can be present at a time.

*B21

Parameter Value	Description
Bit0	Heater High Limit (Plenum A)
Bit1	External Product Safety
Bit2	Boiler Over-Temperature (Plenum A)
Bit3	Boiler Low Water (Plenum A)
Bit4	Dehumidifier System Fault (System B Boiler Over-Temperature)
Bit5	Motor Overload (Plenum A)
Bit6	Fluid System High Limit (Plenum B Heater High Limit)
Bit7	Fluid System High Pressure (Plenum B Motor Overload)
Bit8	Fluid System Low Flow
Bit9	Door Open
Bit10	(System B Boiler Low Water)
Bit11	Not Assigned
Bit12	Emergency Stop
Bit13	Power Failure
Bit14	Transfer Error
Bit15	Not Assigned

y
The individual bits of the word indicate specific alarm conditions. When the bit is on (1) the alarm is present. More than one alarm can be present at a time.

The “A” and “B” plenum designations are for chambers that are equipped with two conditioning plenums. For standard chambers, the default alarm descriptions apply.

*B22

Parameter Value	Description
Bit0	System 1(A) High/Low Pressure
Bit1	System 1(A) Low Oil Pressure
Bit2	System 1(A) Discharge Temperature
Bit3	System 1(A) Protection Module
Bit4	Pumpdown Disabled
Bit5	System 1(A) Floodback Monitor
Bit6	Not Assigned
Bit7	Not Assigned
Bit8	System 2(B) High/Low Pressure
Bit9	System 2(B) Low Oil Pressure
Bit10	System 2(B) Discharge Temperature
Bit11	System 2(B) Protection Module
Bit12	Not Assigned
Bit13	System B Floodback Monitor
Bit14-15	Not Assigned

The individual bits of the word indicate specific alarm conditions. When the bit is on (1) the alarm is present. More than one alarm can be present at a time.

The “A” and “B” designations for the refrigeration system safeties correspond to the system number if the chamber is equipped with dual refrigeration, i.e., dual single stage or cascade refrigeration systems. For cascade systems, the refrigeration system safeties for both the system 1 and system 2 compressors of each system will be combined. Thus, system A safeties are the combination of the system 1 and system 2 safeties for refrigeration system A, and system B safeties are the combined safeties of the second system.

*B23

Parameter Value	Description
Bit0	Humidity Water Reservoir Low
Bit1	Humidity Disabled (temp out-of-range)
Bit2	Humidity High Dewpoint Limit
Bit3	Humidity Low Dewpoint Limit
Bit4	Door Open
Bit5	Vibration Door Switch Bypass (Battery Backup Off/Battery Fail)
Bit6	Vibration Inhibit
Bit7	Vibration Enable
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
Bit14-15	Not Assigned

The individual bits of the word indicate specific alarm conditions. When the bit is on (1) the alarm is present. More than one alarm can be present at a time.

The alarms are in sequential order as they appear on the "Status" view screen on the EZT-570S.

*B24

Parameter Value	Description
0	Autotune Off**
1	Start Autotune
2	Autotune in Progress**
4	Cancel Autotune

**These values are set by the EZT-570S to indicate the autotune status and can not be set directly.

*B25

Parameter Value	Description
0	Alarm Off
1	Absolute High
5	Absolute Low
7	Absolute Both
24	Deviation High
40	Deviation Low
56	Deviation Both

For monitor input alarms, only the absolute high, low or both selections are valid. Monitor inputs are not associated with set points so the deviation alarm modes can not be used for the monitor input alarms. Deviation mode selections are only valid for loop alarms.

*B26

Parameter Value	Description
Bit0	Latching Alarm
Bit1	Reverse Output (Open on Alarm)
Bit2-3	Not Assigned
Bit4	Audible Alarm On (Silent Off)
Bit5	Shut Down on Alarm
Bit6-15	Not Assigned

Only the bits listed perform the control actions as specified. The state of the other bits does not affect alarm operation.

*B27

Parameter Value	Description
0	No Output Selected
1	Digital Output (Customer Event) 1
2	Digital Output (Customer Event) 2
4	Digital Output (Customer Event) 3
8	Digital Output (Customer Event) 4
16	Digital Output (Customer Event) 5
32	Digital Output (Customer Event) 6
64	Digital Output (Customer Event) 7
128	Digital Output (Customer Event) 8
256	Digital Output (Customer Event) 9
512	Digital Output (Customer Event) 10
1024	Digital Output (Customer Event) 11
2048	Digital Output (Customer Event) 12
4096	Digital Output (Customer Event) 13
8192	Digital Output (Customer Event) 14
16384	Digital Output (Customer Event) 15

The individual bits of the word indicate specific alarm conditions. When the bit is on (1) the alarm is present. More than one alarm can be present at a time.

*B28

Parameter Value	Description
0	Online
1	Offline/Downloading Program

When the EZT-570S is offline/downloading a program to the control module, refrain from writing to any control registers. In offline mode, there are no updates made to any control registers.

After a program transfer to the EZT-570S from a PC, the EZT-570S will go into program download. Do not write to any registers until the download is complete.

*C1 The units of measure and range of a loop or monitor input is dependant upon the configuration of the input and/or the units of temperature selection (Celsius or Fahrenheit) of the EZT-570S. The decimal point position for the loop or monitor input is an implied value. Thus, a register value of 345 will represent an actual process value of 34.5.

2.5 EZT-570S Automatic Ramp/Soak Program Registers

The program parameters are a separate group of registers that are used for sending ramp/soak programs to the EZT-570S. The manner in which the program steps are sent to the EZT-570S is specific and must be followed exactly.

Each program step consists of 15 data registers. Programs must be written one step at a time, using a multiple write command (0x10) to write the data for all 15 registers at once. This allows programs to be stored as two-dimensional arrays, of which code can be written to simply index through the array step-by-step, and transmit the program to the EZT-570S.

The first 15 registers of the Program contain specific settings related to the program. These include autostart settings, the program name, the length of the program (number of steps), and guaranteed soak band settings. These values are always transmitted as the first "step" of the program.

Register Address		Parameter Description	Data *D Type	Range *E		*F Unit
				Low	High	
200	**(0x00C8)**	Autostart On/Off	W	*E1	*E1	-
201	**(0x00C9)**	Year/Month for Autostart	W	*E2	*E2	-
202	**(0x00CA)**	Day/DOW for Autotstart	W	*E3	*E3	-
203	**(0x00CB)**	Time of Day for Autostart	W	*E4	*E4	-
204	**(0x00CC)**	Program Name (Chars 1 & 2)	W	*E5	*E5	-
205	**(0x00CD)**	Program Name (Chars 3 & 4)	W	*E5	*E5	-
206	**(0x00CE)**	Program Name (Chars 5 & 6)	W	*E5	*E5	-
207	**(0x00CF)**	Program Name (Chars 7 & 8)	W	*E5	*E5	-
208	**(0x00D0)**	Program Name (Chars 9 & 10)	W	*E5	*E5	-
209	**(0x00D1)**	Total Number of Steps in Program	W	1	99	-
210	**(0x00D2)**	Guaranteed Soak Band Loop 1	W	0.0	3276.7	PV
211	**(0x00D3)**	Guaranteed Soak Band Loop 2	W	0.0	3276.7	PV
212	**(0x00D4)**	Guaranteed Soak Band Loop 3	W	0.0	3276.7	PV
213	**(0x00D5)**	Guaranteed Soak Band Loop 4	W	0.0	3276.7	PV
214	**(0x00D6)**	Guaranteed Soak Band Loop 5	W	0.0	3276.7	PV

The following 15 registers of the Program contain the data for step 1 of the Program.

Register Address		Parameter Description	Data *D Type	Range *E Low	High	*F Unit
215	**(0x00D7)**	Step Time Hours	W	0	9999	-
216	**(0x00D8)**	Step Time Minutes/Seconds	W	*E6	*E6	-
217	**(0x00D9)**	Chamber Events	W	*E7	*E7	-
218	**(0x00DA)**	Customer Events	W	*E7	*E7	-
219	**(0x00DB)**	Guaranteed Soak/Wait for Digital	W	*E8	*E8	-
220	**(0x00DC)**	Wait For Loop	W	*E9	*E9	-
221	**(0x00DD)**	Wait For Monitor	W	*E10	*E10	-
222	**(0x00DE)**	Wait For Loop/Monitor Set Point	W	-3276.8	3276.7	-
223	**(0x00DF)**	Jump Step	W	1	99	-
224	**(0x00E0)**	Cycle Count	W	0	999	-
225	**(0x00E1)**	Loop 1 Set Point	W	-3276.8	3276.7	PV
226	**(0x00E2)**	Loop 2 Set Point	W	-3276.8	3276.7	PV
227	**(0x00E3)**	Loop 3 Set Point	W	-3276.8	3276.7	PV
228	**(0x00E4)**	Loop 4 Set Point	W	-3276.8	3276.7	PV
229	**(0x00E5)**	Loop 5 Set Point	W	-3276.8	3276.7	PV

All remaining steps of the Program follow the same format and data structure as is represented for step one above. Up to the following 1470 registers are used to contain the additional step data of the Program as required for steps 2 through 99. Since few if any programs will contain the maximum of 99 steps, it is only necessary to write the step data for the number steps used in the Program.

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

Through

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

Legend:

*D W Specifies writable data.

*E1

Parameter Value	Description
0	Autostart Off
1	Autostart by Date
2	Autostart by Day

*E2 See *B2 in Section 2.4 for information on the range of this parameter.

*E3 See *B3 in Section 2.4 for information on the range of this parameter.

*E4 See *B4 in Section 2.4 for information on the range of this parameter.

*E5 These parameters contain data which represent up to ten ASCII characters in order to display the name of the currently loaded (or operating) program in the EZT-570S.

See *B15 in Section 2.4 for information on the range of these parameters.

*E6

Parameter Value	Range Low	Range High	Description
High Byte	0	59	Minutes
Low Byte	0	59	Seconds

*E7 See *B12 in Section 2.4 for information on the range of this parameters.

*E8

Parameter Value	Description
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	Wait For Digital Input 1
Bit6	Wait For Digital Input 2
Bit7	Wait For Digital Input 3
Bit8	Wait For Digital Input 4
Bit9	Wait For Digital Input 5
Bit10	Wait For Digital Input 6
Bit11	Wait For Digital Input 7
Bit12	Wait For Digital Input 8
Bit13-15	Not Assigned

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 “wait for digital input” can be enabled at time. The guaranteed soak and “wait for” events can be used concurrently on the same step.

*E9

Parameter Value	Description
0	Wait for Disabled (no loop selected)
1	Wait For Loop 1
2	Wait For Loop 2
4	Wait For Loop 3
8	Wait For Loop 4
16	Wait For Loop 5

*E10

Parameter Value	Description
0	Wait for Disabled (no input selected)
1	Wait For Monitor 1
2	Wait For Monitor 2
4	Wait For Monitor 3
8	Wait For Monitor 4
16	Wait For Monitor 5
32	Wait For Monitor 6
64	Wait For Monitor 7
128	Wait For Monitor 8

2.5.1 Progran Download Algorithm

An example download 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 program 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-570S 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 segement. 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
```

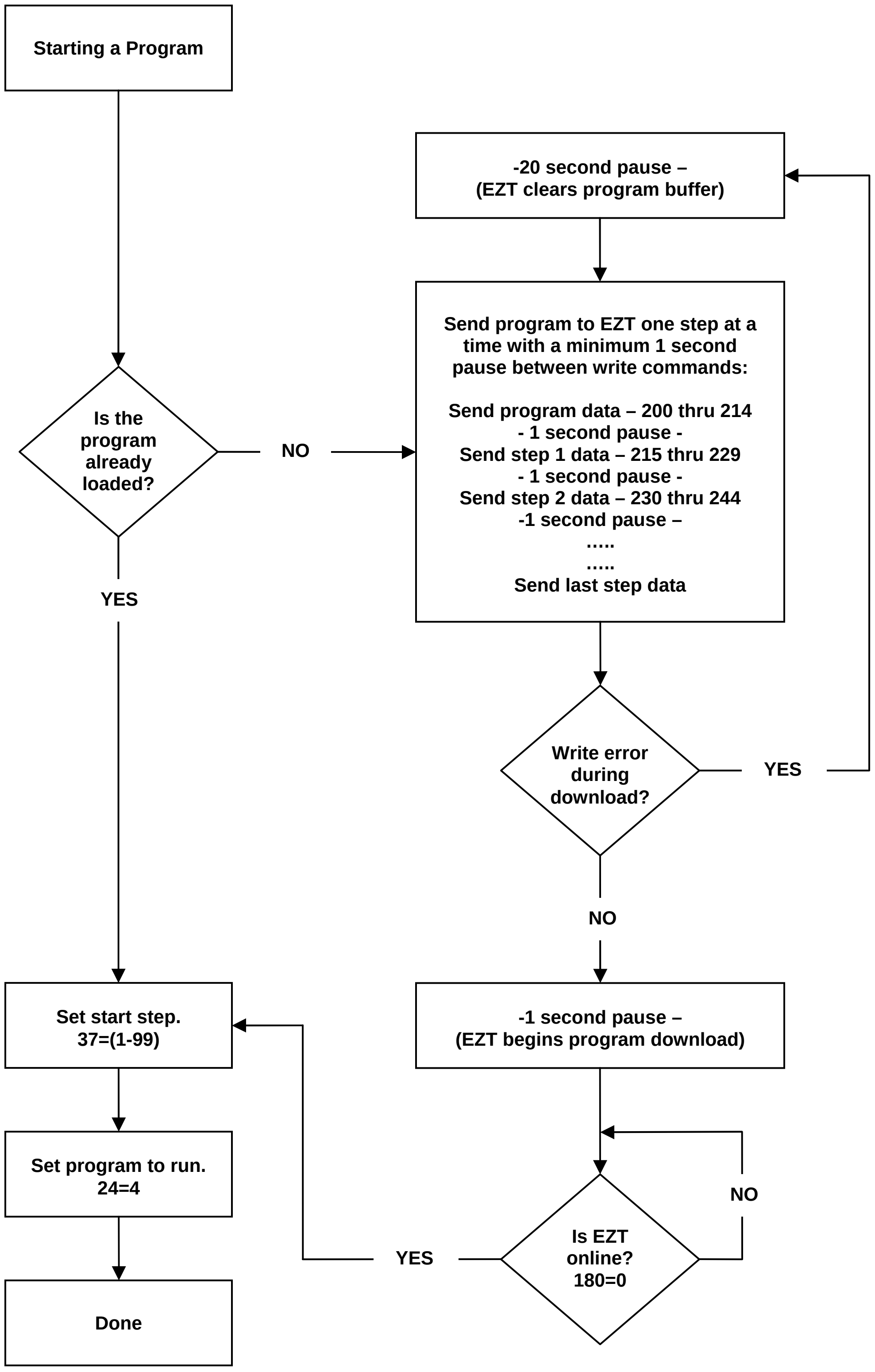
2.5.2 Sending a Ramp/Soak Program to the EZT-570S

Programs 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 program from a PC to the EZT (no response from EZT or NACK returned), the program download must be aborted and restarted.

The EZT-570S is put into program transfer mode when the first group of registers containing the program specific data is sent (registers 200-214). The EZT then begins looking for the number of steps of the program 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 program into the control module 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 program is ready to be started.

If an error occurs during the transfer process from the PC to the EZT, the program 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 program 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.

2.5.3 Starting a Ramp/Soak Program in the EZT-570S


Starting a Program
Is the program already loaded?
NO
YES
-20 second pause –
(EZT clears program buffer)
Send program to EZT one step at a time with a minimum 1 second pause between write commands:
Send program data – 200 thru 214
- 1 second pause -
Send step 1 data – 215 thru 229
- 1 second pause -
Send step 2 data – 230 thru 244
-1 second pause –
.....
.....
Send last step data
Write error during download?
YES
NO
-1 second pause –
(EZT begins program download)
Is EZT online? 180=0
NO
YES
Set start step. 37=(1-99)
Set program to run. 24=4
Done

Appendix

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. C yclic 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.

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.

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.

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.