Cincinnati Sub-Zero

EZT-430i

Single/Dual Loop Controller
User Communications Reference Manual

Safety Information in this Manual

Notes, cautions and warnings appear throughout this book to draw your attention to important operational and safety information.

A "**NOTE**" marks a short message to alert you to an important detail.

A "**CAUTION**" safety alert appears with information that is important for protecting your equipment and performance.

A "**WARNING**" safety alert appears with information that is important for protecting you, others and equipment from damage. Pay very close attention to all warnings that apply to your application.

⚠ This symbol (an exclamation point in a triangle) precedes a general CAUTION or WARNING statement.

⚡ This symbol (a lightning bolt in a lightning bolt in a triangle) precedes an electric shock hazard CAUTION or WARNING safety statement.

Technical Assistance

If you encounter a problem with your EZT-430i controller, review all of your configuration information to verify that your selections are consistent with your application: inputs; outputs; alarms; limits; etc. If the problem persists after checking the above, you can get technical assistance by dialing 1 (513) 772-8810 or by faxing your request to 1 (513) 772-9119, Monday thru Friday, 7:30 a.m. to 5:30 p.m. Eastern Standard Time. You can also email your request to support@cszproducts.com.

An applications engineer will discuss your application with you.

Please have the following information available:

• Complete Model #'s and/or Serial #'s for Component(s) in Question
• Complete Software Version #'s
• All Configuration Information
• All User Manuals

Warranty and return information is on the back cover of this manual.

Your Comments

Your comments or suggestions on this manual are welcome. Please send them to:
Telephone: +1 (513) 326-5252 Fax: +1 (513) 326-5258
support@cszproducts.com

1 What is EZT-430i?

The EZT-430i system combines all of the features of a loop controller, video/chart recorder and data logging system into a single/intuitive device. Email, SMS (text messaging), FTP (file transfer protocol for automated data backup) and remote view/control (Web server/VNC server) are standard with EZT-430i and can be accessed via LAN/WAN using a PC, tablet or smart phone device.

Future Designs "EZT-430i" provides a 4.3"color touch screen interface with standard "Smart Device" user interface features for single and dual loop OEM control applications. All loop configuration and runtime user access is configurable at the device with no PC software required. OEM's have the ability to configure runtime features (screen availability, menus, language, etc...) to easily customize the system for their requirements. These configurations can be imported/exported to any other EZT-430i single/dual loop device for setup (from scratch) within minutes.

Individual high performance board level PID loop control boards (one for each loop) offer up to four control outputs each, powerful profiling capabilities with up to three events and full auto tune functionality with high resolution process inputs.

1.1 Features

Each of the EZT-430i loop control boards provide a single digital input that can be programmed as a automatic program control input for run, hold or abort, a manual mode or failure transfer control input or a program advance to next segment control input.

Each of the loop control boards also provide up to four control outputs which can be used as PID control outputs for heat/cool, direct outputs for controlling external equipment related to the application through software switches called events, or be programmed to act as system alarm outputs.

EZT-430i can be operated in single set point or automatic program control mode. Program entry is made easy through the use of copy, paste and delete menu selections. Programs can be copied to the external USB memory stick and then imported to another EZT-430i controller which eliminates the need to enter duplicate programs into multiple systems.

Data file analysis tools make looking at historical data a simple task. Any control variable saved to the data file can be plotted on the historical data chart for any time frame within the data file's total time range.

The built in Ethernet functionality includes a 'Web Server' to provides access to all EZT-430i data (view only), a VNC interface for remote control and monitoring and an NTS clock, all available via a local Intranet connection (wired or wireless), or the World Wide Web using standard software like Microsoft's Internet Explorer.

EZT-430i provides a rich set of tools for control interaction and process monitoring. Views include single and dual loop views, charts, alarm, automated program status as well as historical data, alarm log and audit trail views. The menu driven interface eliminates screen "clutter" by providing an easy to use "Smart Device" interface for interaction between the user and EZT-430i.

EZT-430i can store more than one year of data on its SD memory card. Data logging can be enabled manually or automatically during program operation. Data backup is provided with a USB memory stick for plug and play transfer of files to any PC running Microsoft Windows XP operating systems and via the FTP back-up utility.

EZT-430i protects system access with 4 level security (user rights based), audit trails that document all user activity and ensures data integrity by digitally signing all data files and audit trails to meet regulatory requirements.

The EZT-430i controller includes the following features:

- Single/Dual loop controller models (automatic program operation included).
- Touch screen, “Smart Device” user interface (UI).
- Video recorder mode for view only applications.
- Email, SMS, FTP, VNC and Web functionality standard.
- Remote View/Control using PC, Tablet or Smartphone.
- Detailed maintenance, alarm monitoring and alarm history.
- User configurable data logging and historical data viewer.
- 4 level security with digitally signed audit trails and data files.
- National time server connectivity with daylight savings.
- Multi-lingual user interface supports over 25 languages.
- 30,000 hour LED display

2 Communications Wiring

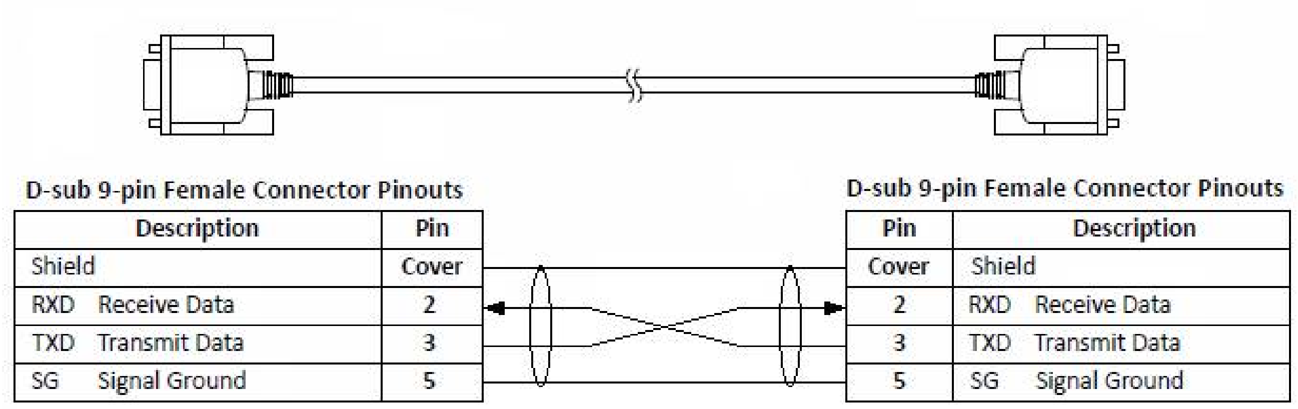
WARNING:

- To avoid potential electric shock and other hazards, all mounting and wiring for EZT-430i must conform to the National Electric Code (NEC) and other locally applicable codes.
- Special expertise is required to install, wire, configure and operate the EZT-430i controller. Personnel without such expertise should not install, wire or operate nCompass.

CAUTION:

- Prevent metal fragments and pieces of wire from dropping inside the housing of any EZT-430i component. If necessary, place a cover over the component during installation and wiring. Ingress of such fragments and chips may cause a fire hazard, damage or malfunction of the device.
- Locate the EZT-430i touch screen and all related control components away from AC power/motor wiring and sources of direct heat output such as transformers, heaters or large capacity resistors.

The EZT-430i touch screen provides an RS232C (COM1) user communications port for connecting EZT-430i to a PC running software such as CSZ's Envision. In order to connect EZT-430i to a PC, a cable must be made according to the diagram below.



D-sub 9-pin Female Connector Pinouts

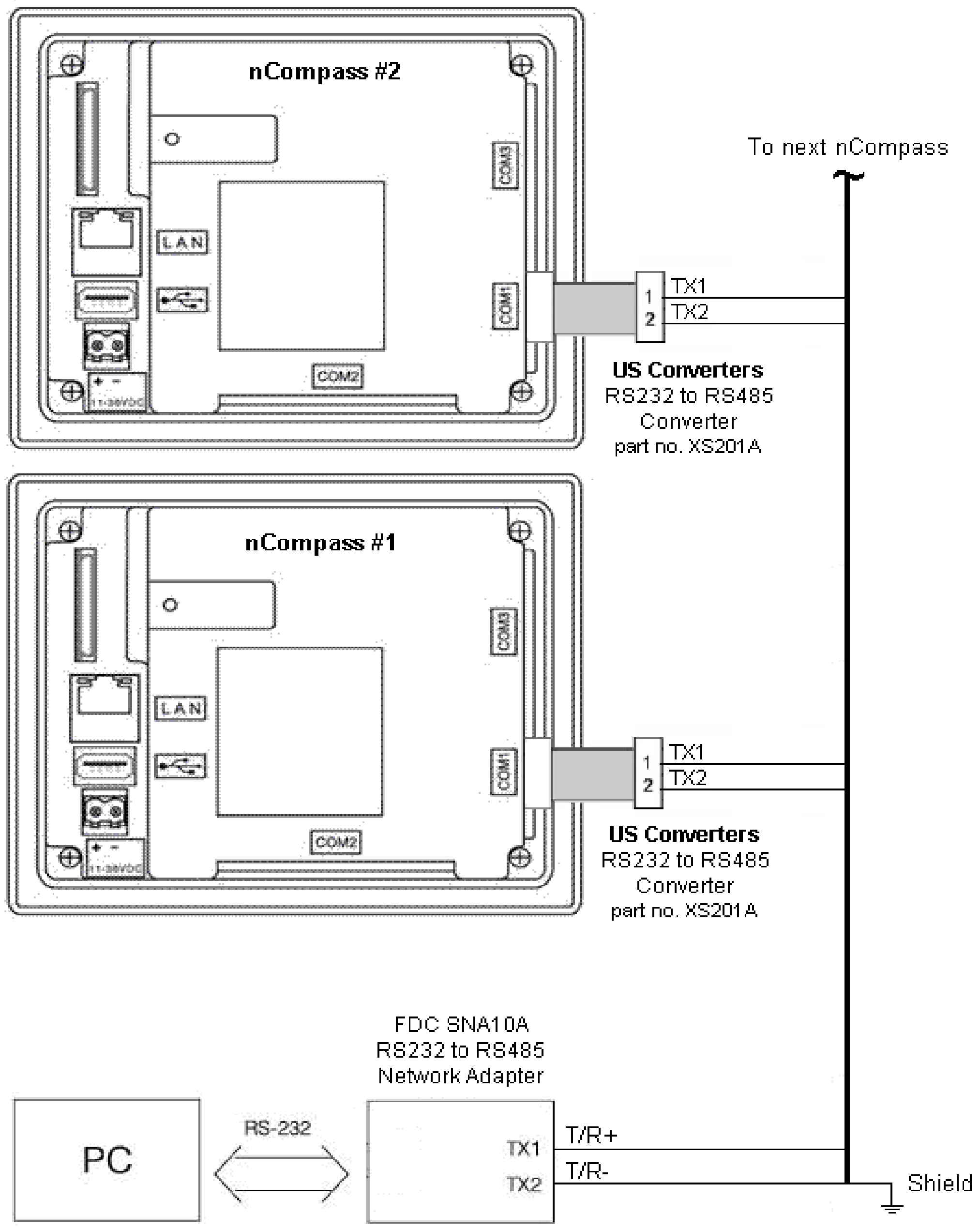
Description	Pin
Shield	Cover
RXD Receive Data	2
TXD Transmit Data	3
SG Signal Ground	5

D-sub 9-pin Female Connector Pinouts

Pin	Description
Cover	Shield
2	RXD Receive Data
3	TXD Transmit Data
5	SG Signal Ground

NOTE: *DO NOT use a standard null-modem cable to connect EZT-430i to a PC. Most computers do not provide a standard serial port and a USB to serial converter must be used. Incompatibilities may exist between EZT-430i and certain USB to serial adapters which will cause EZT-430i to malfunction when connected using a standard null-modem cable.*

If more than one EZT-430i controller is to be placed on the communication link with a PC, an RS232/485 converter will be required for each EZT-430i and the PC in order to convert the RS232 communications port to RS485 multi-drop communications network for connecting multiple EZT-430i controllers on the PC communications link.



NOTE: *The connection requires a single twisted-pair cable that is daisy-chained from one EZT-430i to the next. When using shielded twisted-pair cable, be sure to ground only when end of the cable, preferably at the RS232 to RS485 network adapter. Allowing any other portion of the cable shield to come in contact with ground, or grounding both ends, will cause ground loop currents to flow in that section of the cable which can cause communication errors.*

3 Communication Basics

The purpose of this document is to provide users interested in using data communications with the EZT-430i, the ability to set up and use a simple network of one or more EZT-430i controllers by providing a basic understanding of data communications using standard definitions, interfaces and protocols.

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.

3.1 Explanation 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 require 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 data in both directions (half duplex).

Character Format

The code or character format for EZT-430i data communications is shared by virtually everyone in the electronics industry. This code defines a 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 **bi**nary digi**t**. 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

4 Serial Communication

The user communications interface for EZT-430i employs serial communication, which is the exchange of data in a one-bit-at-a-time, sequential manner on a single data line or channel. Serial contrasts with parallel communication, which sends several bits of information simultaneously over multiple lines or channels. Not only is serial data communication simpler than parallel, it is also less costly.

Baud Rate

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

Typical baud rates used for computers are 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600 and 115200 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. When operating in the standard interface mode, EZT-430i uses a 9600 baud rate. When operating in the simulated Watlow F4S/D mode, EZT-430i uses a 19200 baud rate. *In order for a device to communicate with EZT-430i , it must have its serial port set for either 9600 baud or 19200 baud based on the selected interface type in order 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. EZT-430i uses the default – and most common – length of 1 period for the stop bit. *A device used to communicate with EZT-430i 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. There are several defined parity selections available for serial communications. They are even parity, odd parity, mark parity, space parity or none at all can be used.

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. 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.

When operating in the standard interface mode, EZT-430i offers parity settings of Even, Odd and None. When operating in the simulated Watlow F4S/D interface mode, the parity is defaulted to “None” and is not adjustable. *In order for a device to communicate with EZT-430i , it must have its serial port set to use the same parity setting in order for data communications to work properly.*

4.1 Interface Standards

An interface is a means for electronic systems to interact. It's a specific kind of electrical wiring configuration. It has nothing to do with how data is sent over that connection. The two most common interfaces used today are RS-232, which provides a simple 1 to 1 connection and RS-485, which provides a multi-drop connection where more than one device can be placed on the same line. The EZT-430i communications interface is RS-232, but can be changed to RS-485 through the use of external RS232/485 adapters.

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 slave 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 an EIA-485 multi-drop system. The standards do not specify the wire size and type. Use of 24 AWG 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.

4.1.1 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-430i 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 without 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 an external power supply. This is especially necessary when using a laptop computer. See the documentation that is provided with your converter for more information.

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. The following converter has been tested and is compatible with EZT-430i . The converter is 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.

US Converters
405 W. Fairmont Dr.
Tempe, AZ 85282
E-mail: mail@usconverters.com
www.USconverters.com

Part # **XS201A** RS232 to RS485 Converter

Future Design Controls
7524 West 98th Place
Bridgeview, IL 60455
Phone: 888-751-5444
Fax: 888-307-8014
E-mail: csr@futuredesigncontrols.com
www.futuredesigncontrols.com

Part # **SNA10A** Smart Network Adapter
Part # **DB9M-DB9F-6ft** (Cable Accessory to connect SNA10A to PC)

4.2 Protocol

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

The protocol portion of EZT-430i 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-430i 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. EZT-430i uses Modbus RTU as the protocol of choice. Modbus RTU enables a PC to read and write directly to registers containing the EZT-430i parameters. With it, you can read all of the controller's parameters with just a single read command.

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. 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 is exchanged in hex numbers. EZT-430i only supports the binary version of Modbus, referenced as RTU. The ASCII version is less efficient and is not supported. Therefore, you must be certain to format all data in hexadecimal.

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

1. Load a 16-bit register (called CRC register) with 0xFFFF

2. Exclusive OR the first 8-bit byte of the command message with the low order byte of the 16-bit CRC register, putting the result in the CRC register.

3. Shift the CRC register one bit to the right with MSB zero filling. Extract and examine the LSB.

4. If the LSB of the CRC register is zero, repeat step 3, else Exclusive OR the CRC register with the polynomial value 0xA001.

5. Repeat steps 3 and 4 until eight shifts have been performed. When this is done, a complete 8-bit byte will have been processed.

6. Repeat steps 2 through 5 for the next 8-bit byte of the command message. Continue doing this until all bytes of the command message have been processed. The final contents of the CRC register is the CRC value.

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

Example Cyclical Redundancy Checksum (CRC) Algorithm

```
unsigned int ca_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);
}
```

4.3 Creating your own Modbus Application

Listed below are a few of the more common software packages that claim to support the Modbus protocol. This list is provided as informational only. Contact the software manufacturer for more information on applying their software.

LabView by National Instruments	Wonderware by Wonderware	SpecView by SpecView Corporation
11500 N Mopac Expwy	26561 Rancho Pkwy. South	13409 53rd Ave NW
Austin, TX 78759-3504	Lake Forest, CA 92630	Gig Harbor, WA 98332
Phone 800-683-8411	Phone 949-727-3200	Phone 253-853-3199
http://www.natinst.com	http://www.wonderware.com	http://www.specview.com

If you already have a software application that uses Modbus, you can simply skip to the EZT-430i 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 must code messages in eight-bit bytes, with even parity, one stop bit (8, even, 1). EZT-430i has its parity set to even as default from the factory.

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-430i communications at 9600 baud, this caulates to a minimum period of ~3ms.

   In addition, the EZT-430i timeout period must be added to that in order to properly time the send and receive messages between the host computer and multiple EZT-430i controllers on the serial link. With a default timeout period in EZT-430i of 135ms, it makes a total pause of 138ms minimum. Thus, after you receive a response from an EZT-430i controller at your PC, you must wait a minimum of 138ms before sending the next command.

4. Values containing decimal points such as process values and set points, have the decimal point implied, i.e., the data exchange can only be performed using whole numbers. Thus, the value must be scaled appropriately in order to exchange the data correctly. For example, a setpoint of 78.4 degrees must be sent as a value of 784 in order for EZT-430i to be set correctly. Likewise, a process value read from EZT-430i with a value of 827 is actually 82.7 degrees. Consult the parameter table for the proper format and allowable range 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.

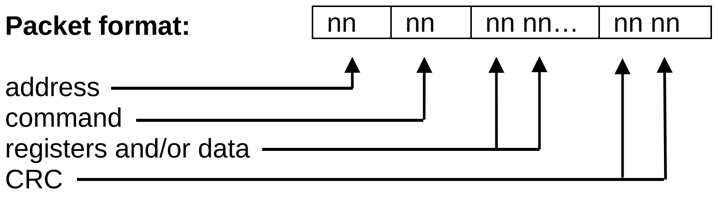
Handling Communication Errors
Messages with the wrong format or illegal values will receive an exception response. Messages with the wrong CRC or timing will receive no response. It is the user’s responsibility to handle the error appropriately within their own software and determine whether to resend the message or halt for operator intervention.

User Responsibility
Refrain from reading or writing from/to a register that does not exist or is currently disabled. Writing values to unassigned registers could cause system instability, malfunction or failure. Care must also be taken in 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.

4.3.1 Packet Syntax

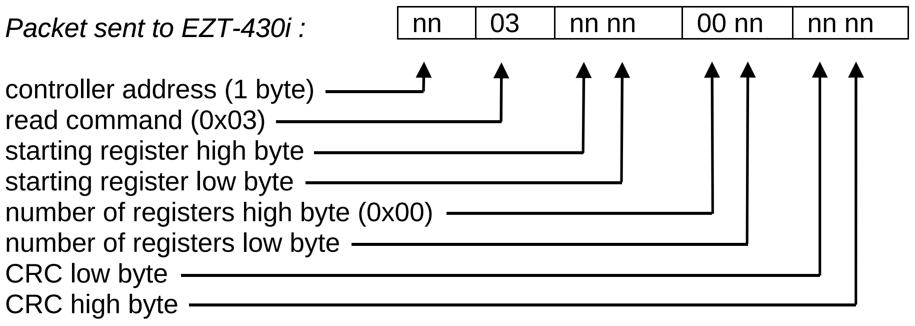
Each message packet begins with a one-byte controller address, from 0x01 to 0x1F. The second byte in the message packet identifies the message command: read (0x03); write single (0x06) or write multiple (0x10). The next "n" bytes of the message packet contain register addresses and/or data. The last two bytes in the message packet contain a two-byte Cyclical Redundancy Checksum (CRC) for error detection.

Packet format:

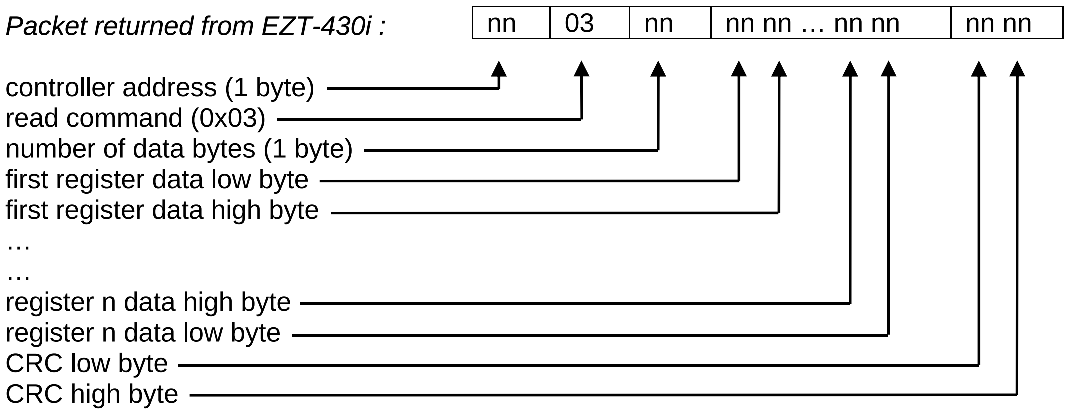


Read Register(s) Command (0x03)
This command returns from 1 to 60 registers. This command is used for reading one or more data locations from EZT-430i .

Packet sent to EZT-430i :


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-430i :


nn
03
nn
nn nn … nn nn
nn nn
controller address (1 byte)
read command (0x03)
number of data bytes (1 byte)
first register data low byte
first register data high byte
…
…
register n data high byte
register n data low byte
CRC low byte
CRC high byte

Example: Read registers 35 and 36 (loop 1 process variable and setpoint) of controller at address 1 configured for 1 decimal point.

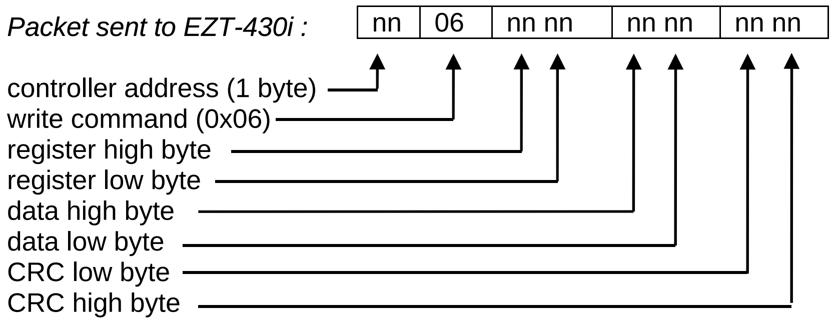
Sent: 01 03 00 23 00 02 35 C1
Received: 01 03 04 **03 0D 01 F3** 2A 61

Message data: 781 (0x**030D**) = process variable of 78.1
499 (0x**01F3**) = setpoint of 49.9

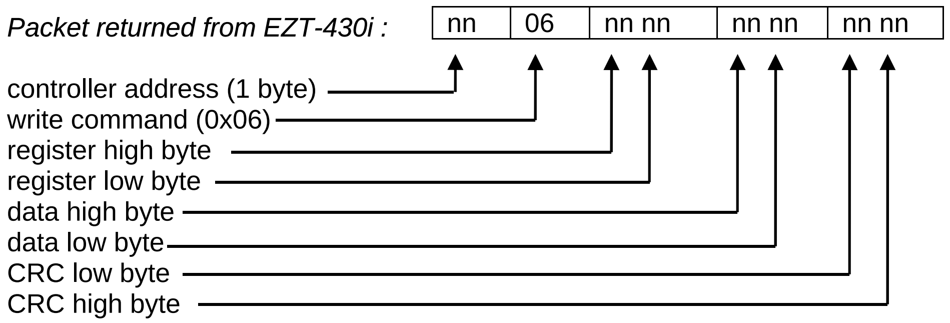
Write Register Command (0x06)

This command writes a value to a single register. This command is to be used for setting control values in EZT-430i . To set multiple values, repeat the command for each data location.

Packet sent to EZT-430i :



Packet returned from EZT-430i :



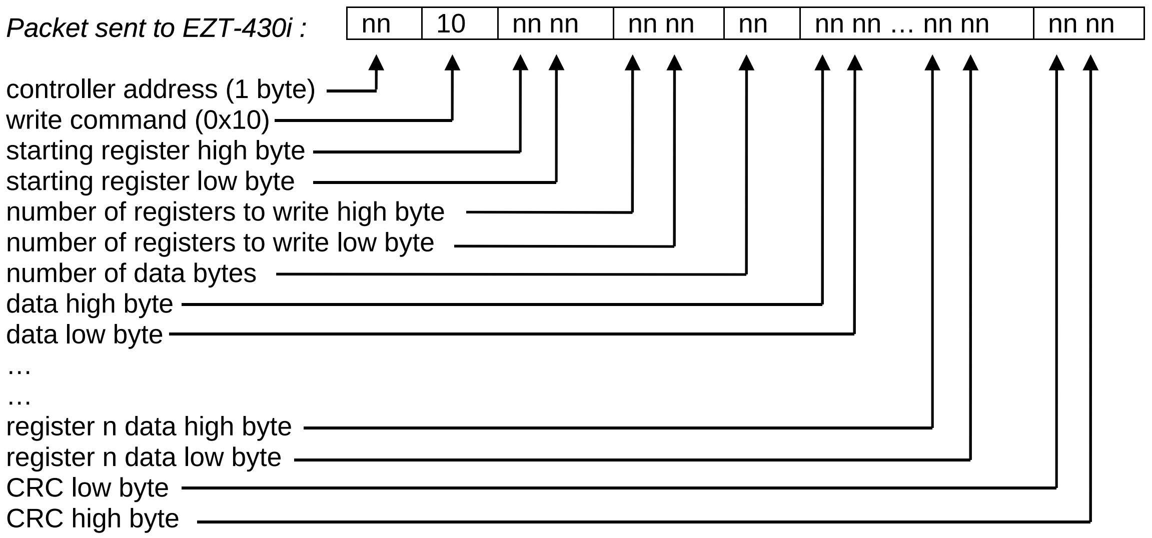
Example: Write register 41 (loop 2 set point) of controller at address one configured with no decimal point to 75 degrees (0x**004B**).

Sent: 01 06 00 29 **00 4B** 18 35
Received: 01 06 00 29 00 4B 18 35

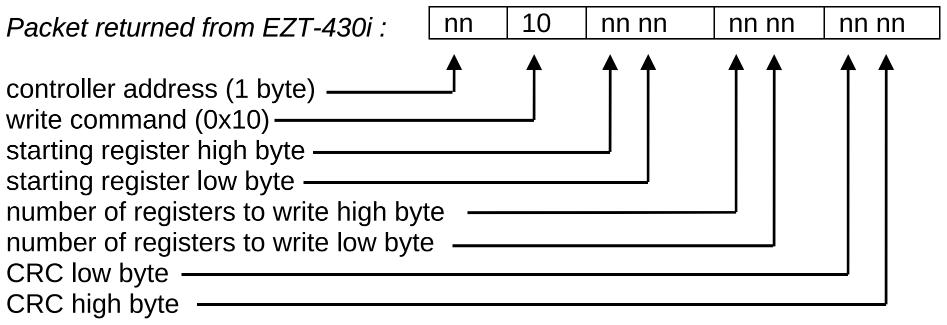
Write Registers Command (0x10)

This command writes values to multiple registers in sequential order. It is used for automatic program download only to transmit program data one step at a time to EZT-430i . See the Automatic 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, EZT-430i will respond with an acknowledgment that the message was received; however, the command will not be executed.

Packet sent to EZT-430i :


nn
10
nn nn
nn nn
nn
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
number of registers to write low byte
number of data bytes
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-430i :

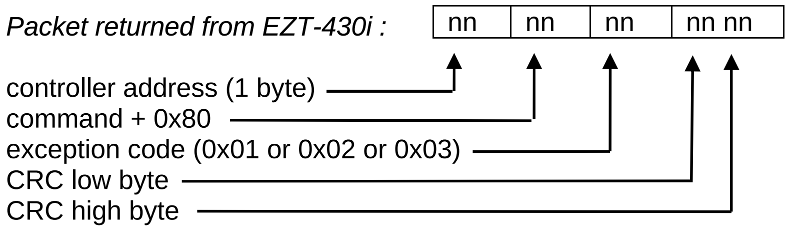

nn
10
nn nn
nn nn
nn nn
controller address (1 byte)
write command (0x10)
starting register high byte
starting register low byte
number of registers to write high byte
number of registers to write low byte
CRC low byte
CRC high byte

Exception Responses

When EZT-430i 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 EZT-430i :


nn
nn
nn
nn nn
controller address (1 byte)
command + 0x80
exception code (0x01 or 0x02 or 0x03)
CRC low byte
CRC high byte

4.3.2 Error Checking

In Modbus communications, every message sent from the master (your software) receives a response from the slave (EZT-430i), 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 caulate 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 EZT-430i 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 35 and 36 (loop 1 process variable and setpoint) of controller at address 1.

Command sent to EZT-430i :	01 03 00 23 00 02 35 C1
Message received from EZT-430i :	01 03 04 03 0D 01 F3 2A 61

Caulated CRC:	2A61 (caulated from message 01 03 04 03 0D 01 F3)
Received CRC:	2A61

The caulated CRC matches the received CRC, the message is valid. Note that the last two bytes of the received message are not used to caulate the CRC. The last two bytes are the CRC that EZT-430i appended to the message. Do not include them when caulating the CRC.

Transmitting and Receiving Messages

In order to reliably communicate with EZT-430i , 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 EZT-430i , 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. EZT-430i uses its own fixed timeout setting of 135ms. Thus, if the entire message is not received by EZT-430i within 135ms, it will discard the data it has received and assume the next data byte received is the start of the next valid message.

The timeout must be considered carefully when developing your application. 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 EZT-430i within the proper time frame.

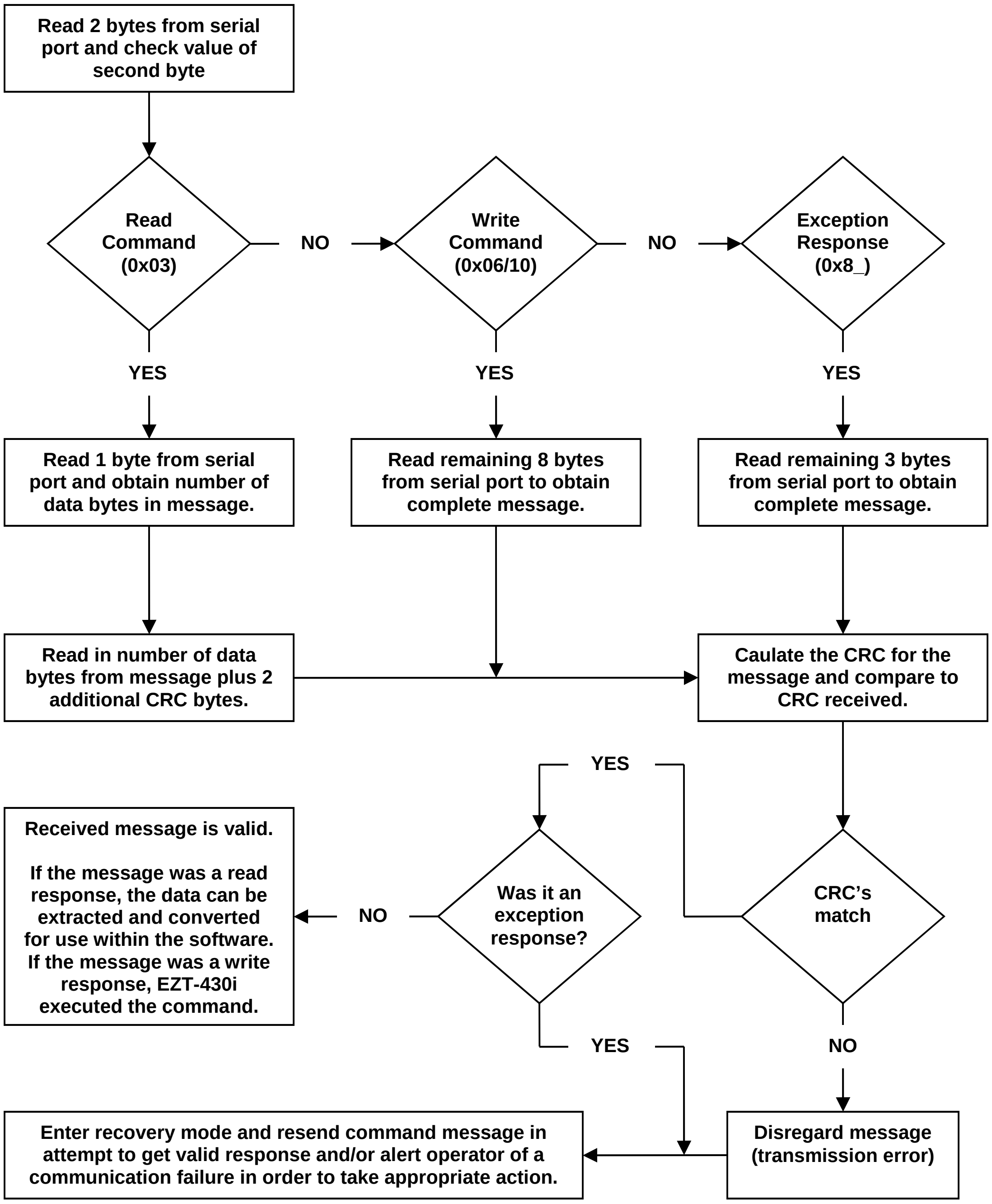
Receiving Messages

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 4.3.1, the length of the returned message will very 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 125 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 caulate 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 caulate 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.
Caulate 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, EZT-430i 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)

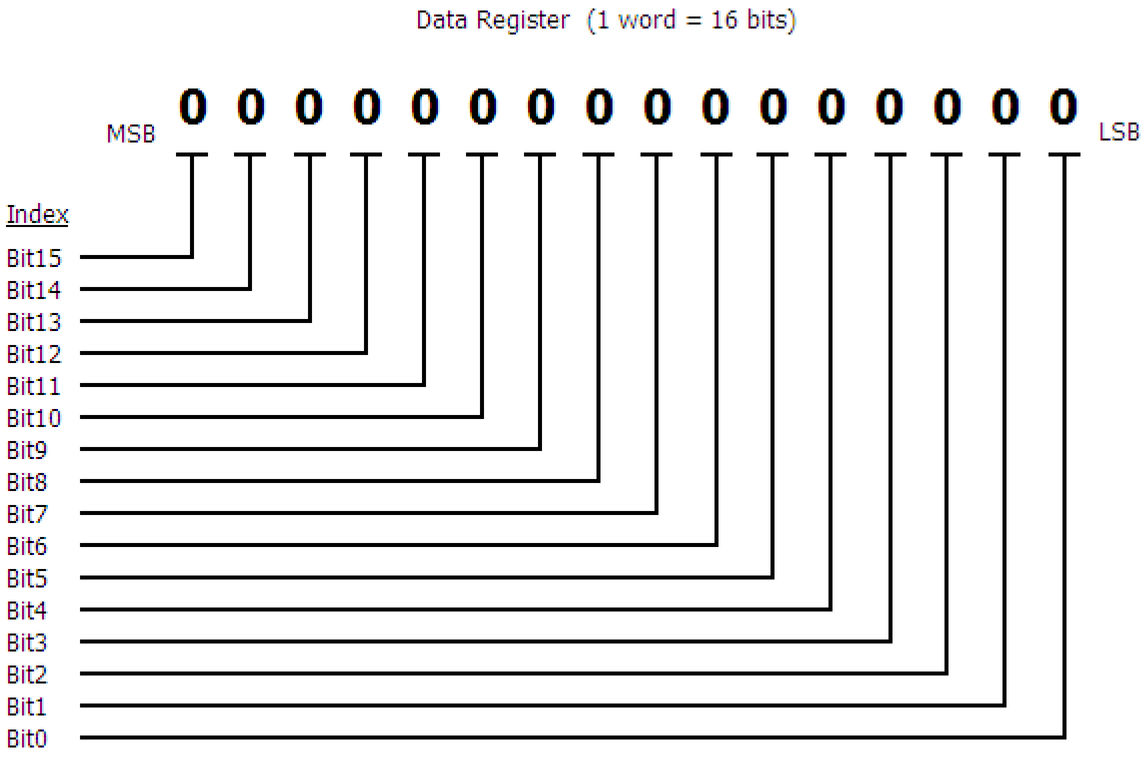
.

5 EZT-430i Data Registers (standard serial interface)

Some of the values contained in the EZT-430i 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 hexadecimal value that can be sent to EZT-430i in order to set a control register.

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 (LSB). 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 lists of registers and their allowable data ranges. DO NOT attempt to write to any other register number than those listed. DO NOT write to registers that are for options your controller does not have. Failure to adhere to this requirement can result in erratic control and/or damage to equipment.

<u>All register numbers are listed in relative values. To convert to absolute values, add 400001.</u>

5.1 Control Registers

Register Address		Parameter Description	Data *A Type	Range *B Low	High	*C Unit
0	**(0x0000)**	System Offline/Busy Status	R	*B1	*B1	-
1	**(0x0001)**	Alarm Reset	R/W	0	1	-
2	**(0x0002)**	Automatic Program Out of Sync Alarm	R	0	1	-
3	**(0x0003)**	Loop Communication Alarm Status	R	*B2	*B2	-
4	**(0x0004)**	Loop Control Error Status	R	*B3	*B3	-
5	**(0x0005)**	Input Alarm Status	R	*B4	*B4	-
6	**(0x0006)**					
7	**(0x0007)**					
8	**(0x0008)**					
9	**(0x0009)**	Control Loop Manual Override	R/W	*B5	*B5	-
10	**(0x000A)**	Control Loop Autotune Activation	R/W	*B6	*B6	-
11	**(0x000B)**	**RESERVED** – DO NOT write to register				
12	**(0x000C)**	System Events	R/W	*B7	*B7	-
13	**(0x000D)**					
14	**(0x000E)**	Automatic Program Start Step Number	R/W	1	64	-
15	**(0x000F)**	Automatic Program Operating Status	R/W	*B8	*B8	-
16	**(0x0010)**	Program Name Characters 1 & 2	R	*B9	*B9	-
17	**(0x0011)**	Program Name Characters 3 & 4	R	*B9	*B9	-
18	**(0x0012)**	Program Name Characters 5 & 6	R	*B9	*B9	-
19	**(0x0013)**	Program Name Characters 7 & 8	R	*B9	*B9	-
20	**(0x0014)**	Program Name Characters 9 & 10	R	*B9	*B9	-
21	**(0x0015)**	Program Name Characters 11 & 12	R	*B9	*B9	-
22	**(0x0016)**	Program Name Characters 13 & 14	R	*B9	*B9	-
23	**(0x0017)**	Current Program Step	R	1	64	-
24	**(0x0018)**	Total Number of Steps	R	1	64	-
25	**(0x0019)**	Total Time of Step	R	0	99.59	-
26	**(0x001A)**	Time Remaining in Step	R	0	99.59	-
27	**(0x001B)**	Cycles Remaining for Current Jump Loop	R	1	10000	-
28	**(0x001C)**					
29	**(0x001D)**					
30	**(0x001E)**					
31	**(0x001F)**	Loop 1 Target Setpoint for Current Step	R	-32768	32767	*C1
32	**(0x0020)**	Loop 2 Target Setpoint for Current Step	R	-32768	32767	*C1
33	**(0x0021)**					
34	**(0x0022)**					
35	**(0x0023)**	Loop 1 Process Variable (PV)	R	-32768	32767	*C1
36	**(0x0024)**	Loop 1 Setpoint (SP)	R/W	-32768	32767	*C1

Register Address		Parameter Description	Data *A Type	Range *B		*C
				Low	High	Unit
37	**(0x0025)**	Loop 1 Percent Output (%Out)	R/W	-100.00	100.00	%
38	**(0x0026)**	Loop 1 Mode and Operational Status	R	*B10	*B10	-
39	**(0x0027)**	Loop 1 Error Code	R	*B11	*B11	-
40	**(0x0028)**	Loop 2 Process Variable (PV)	R	-32768	32767	*C1
41	**(0x0029)**	Loop 2 Setpoint (SP)	R/W	-32768	32767	*C1
42	**(0x002A)**	Loop 2 Percent Output (%Out)	R/W	-100.00	100.00	%
43	**(0x002B)**	Loop 2 Mode and Operational Status	R	*B10	*B10	-
44	**(0x002C)**	Loop 2 Error Code	R	*B11	*B11	-
45	**(0x002D)**					
46	**(0x002E)**					
47	**(0x002F)**					
48	**(0x0030)**					
49	**(0x0031)**					
50	**(0x0032)**					
51	**(0x0033)**	Alarm 1 Setpoint	R/W	*B12	*B12	*C1
52	**(0x0034)**	Alarm 2 Setpoint	R/W	*B12	*B12	*C1
53	**(0x0035)**	Alarm 3 Setpoint	R/W	*B12	*B12	*C1
54	**(0x0036)**	Alarm 4 Setpoint	R/W	*B12	*B12	*C1
55	**(0x0037)**	Alarm 5 Setpoint	R/W	*B12	*B12	*C1
56	**(0x0038)**	Alarm 6 Setpoint	R/W	*B12	*B12	*C1
57	**(0x0039)**					
58	**(0x003A)**					
59	**(0x003B)**					

Notes:

*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 are 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
0	EZT-430i Online
1	Offline/Program Download in Progress

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	Description
Bit0	Loop 1 Communications Error
Bit1	Loop 2 Communications Error
Bit2 – Bit15	Not Assigned

*B3

Parameter Value	Description
Bit0	Loop 1 Control Error
Bit1	Loop 2 Control Error
Bit2 – Bit15	Not Assigned

Note: If either bit is on, check the corresponding loop mode and operational status register to determine the cause for the alarm.

*B4

Parameter Value	Description
Bit0	Alarm 1 Active
Bit1	Alarm 2 Active
Bit2	Alarm 3 Active
Bit3	Alarm 4 Active
Bit4	Alarm 5 Active
Bit5	Alarm 6 Active
Bit6 – Bit15	Not Assigned

*B5

Parameter Value	Description
Bit0	Loop 1 Manual Mode
Bit1	Loop 2 Manual Mode
Bit2 – Bit15	Not Assigned

Note: When loop is in manual mode, the loop percentage of output can be set by writing to the corresponding loop percent output register.

*B6

Parameter Value	Description
Bit0	Loop 1 in Auto Tune
Bit1	Loop 2 in Auto Tune
Bit2 – Bit15	Not Assigned

Note: Auto tune may not be available depending upon the loop configuration. If auto tune operation is not available, the bit for the loop will automatically turn off after being set.

When auto tune completes normally, the bit for the loop will automatically turn off indicating that tune is complete. To terminate an auto tune in progress, turn off the bit for the desired loop.

*B7

Parameter Value	Description
Bit0	Event 1
Bit1	Event 2
Bit2	Event 3
Bit3	Event 4
Bit4	Event 5
Bit5	Event 6
Bit6 – Bit15	Not Assigned

Note: Not all system events may be available on your system. Event names and functions are defined by system configuration. Consult your system documentation or contact your OEM for information on event use.

*B8

Parameter Value	Description
0	Automatic Program Start (On)
1	Automatic Program Hold
2	Automatic Program Halt (Off)

*B9

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 16 to 22 from EZT-430i returns the following values:

```
Register Values:   0x74 53    0x72 6F    0x20 65    0x65 54    0x74 73    0x20 20    0x20 20
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 16 in order to assemble the automatic 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 14 character name length if the program name is not 14 characters long.

*B10

Parameter Value	Description
Bit0	Automatic Program Run Mode
Bit1	Automatic Program Hold Mode
Bit2	Static Mode
Bit3	Auto Tune Mode
Bit4	Manual Mode
Bit5	Off Mode
Bit6	Failure Transfer
Bit7	Automatic Program Ramp Up
Bit8	Automatic Program Ramp Down
Bit9	Automatic Program Soak
Bit10	Alarm 1 Active
Bit11	Alarm 2 Active
Bit12	Alarm 3 Active
Bit13	Event 1 On
Bit14	Event 2 On
Bit15	Event 3 On

Note: The “alarm active” and “event on” status bits can not be used for determining alarm or event status. The alarm and event activation is based upon loop configuration. Do monitor alarm and event status, use registers 5 and 12 respectively.

*B11

Parameter Value	Description
0	No Error
4	Illegal Setup Values
10	Comm Error – Bad Function Code
11	Comm Error – Register Out of Range
14	Comm Error – Write Read Only Data
15	Comm Error – Out of Range Data
25	Holdback Time Out
26	Auto Tune Error
27	Input Type Requires Calibration
29	EEPROM Error
30	Cold Junction Failure
39	Sensor Break
40	A to D Failure

*B12

Parameter Value	Deviation Alarm	Process Alarm
Range Low	-18000	SP.Lo
Range High	18000	SP.Hi

*C1 The units of measure and range of a loop input is dependant upon the configuration of the input and/or the units of temperature selection (Celsius or Fahrenheit) of EZT-430i . The decimal point position for the loop or monitor input is an implied value based on the configuration of the input. Thus, a register value of 345 can represent an actual process value of 345, 34.5, 3.45 or 0.345 depending upon the decimal point configuration of the loop or monitor input.

5.2 Automatic Program Registers

The automatic program parameters are a separate group of registers that are used to load programs into EZT-430i . The manner in which the program steps are configured and sent to EZT-430i is specific and must be followed exactly.

Each program step consists of 14 data registers. The program must be written one step at a time, using a multiple write command (0x10) to write the data for all 14 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 file to EZT-430i .

The first 14 registers of the program contain specific settings related to the program. These include hold back bands, ramp units, soak units, the program name and the length of the program (number of steps).

Register Address		Parameter Description	Data *D Type	Range *E		*F
				Low	High	Unit
100	**(0x0064)**	Hold Back Band Loop 1	W	*E1	*E1	-
101	**(0x0065)**					
102	**(0x0066)**	Ramp Units	W	*E2	*E2	-
103	**(0x0067)**	Soak Units	W	*E3	*E3	-
104	**(0x0068)**	Hold Back Band Loop 2	W	*E1	*E1	-
105	**(0x0069)**					
106	**(0x006A)**	Total Number of Steps	W	1	64	-
107	**(0x006B)**	Program Name (Chars 1 & 2)	W	*E4	*E4	-
108	**(0x006C)**	Program Name (Chars 3 & 4)	W	*E4	*E4	-
109	**(0x006D)**	Program Name (Chars 5 & 6)	W	*E4	*E4	-
110	**(0x006E)**	Program Name (Chars 7 & 8)	W	*E4	*E4	-
111	**(0x006F)**	Program Name (Chars 9 & 10)	W	*E4	*E4	-
112	**(0x0070)**	Program Name (Chars 11 & 12)	W	*E4	*E4	-
113	**(0x0071)**	Program Name (Chars 13 & 14)	W	*E4	*E4	-

The following 14 registers of the automatic program contain the data for step 1 of the program.

Register Address		Parameter Description	Data D Type	Range *E		*F Unit
				Low	High	
114	**(0x0072)**	Step Number*	W	0	63	-
115	**(0x0073)**	Step Type	W	*E5	*E5	-
116	**(0x0074)**	Ramp Target Setpoint Loop 1	W	-32768	32767	PV
117	**(0x0075)**	Ramp Time or Ramp Rate	W	0	5999	-
118	**(0x0076)**	Event Selections 1, 2, 3	W	*E6	*E6	-
119	**(0x0077)**	Holdback Type Loop 1	W	*E7	*E7	-
120	**(0x0078)**	Dwell Time	W	0	9999	-
121	**(0x0079)**	Jump Step	W	0	63	-
122	**(0x007A)**	Jump Cycles	W	1	9999	-
123	**(0x007B)**	Final Setpoint Loop 1	W	-32768	32767	PV
124	**(0x007C)**	Ramp Target Setpoint Loop 2	W	-32768	32767	PV
125	**(0x007D)**	Event Selections 4, 5, 6	W	*E8	*E8	-
126	**(0x007E)**	Holdback Type Loop 2	W	*E7	*E7	-
127	**(0x007F)**	Final Setpoint Loop 2	W	-32768	32767	PV

***The **Step Number** must be offset by 1 when writing step data to EZT-430i . Steps 1-64 will be sent with the step number as a value of 0-63 (-1 offset).

All remaining steps of the program follow the same format and data structure as is represented for step one above. Up to the following 882 registers are used to contain the additional step data of the program as required for steps 2 through 64. Since few if any programs will contain the maximum of 64 steps, it is only necessary to write the step data for the number steps used in the automatic program.

128 (0x0080) – 141 (0x008D)	Program Step 2 Data Registers
142 (0x008E) – 155 (0x009B)	Program Step 3 Data Registers
156 (0x009C) – 169 (0x00A9)	Program Step 4 Data Registers
170 (0x00AA) – 183 (0x00B7)	Program Step 5 Data Registers
184 (0x00B8) – 197 (0x00C5)	Program Step 6 Data Registers
198 (0x00C6) – 211 (0x00D3)	Program Step 7 Data Registers
212 (0x00D4) – 225 (0x00E1)	Program Step 8 Data Registers
226 (0x00E2) – 239 (0x00EF)	Program Step 9 Data Registers

----
through
----

996 (0x03E4) – 1009 (0x03F1)	Program Step 99 Data Registers

Notes:*DW Specifies write only data.

*E1

Parameter Value	Temperature Units °C	Temperature Units °F	Process Units
Range Low	1	1	1
Range High	555	999	999

*E2

Parameter Value	Description
0	Hours and Minutes
1	Minutes and Seconds
2	Units/Minute
3	Units/Hour

Note: DO NOT use ramp units of units/minute or units/hour on a dual loop system. Since step time is defined by the ramp rate and the change in set point, the program will get out of sync between each loop causing an automated program run error.

*E3

Parameter Value	Description
0	Hours and Minutes
1	Minutes and Seconds

*E4 See note *B9 in Section 5.1 for information on the range of this parameters.

*E5

Parameter Value	Description
0	Ramp
1	Soak
2	Jump
3	End

Note: The last step of an automatic program must be an end step. If the last step is not an end step, the program will not run correctly and/or a program download error will occur and the program will not operate.

*E6

Parameter Value	Description
Bit0	Event - Output 2
Bit1	Event - Output 3
Bit2	Event - Output 4
Bit3 – Bit15	Not Assigned

Note: The available system events are based on the configuration of nCompass. Event 1 may correspond to loop 1 “event - output 4” and event 2 may correspond with loop 2 “event - output 3”. To properly set the events of your EZT-430i system, you must know the event output configuration of the loop controls and set the appropriate bit for the output the event is assigned to.

Contact your OEM for equipment related questions. Only they can answer questions regarding the configuration of your nCompass.

*E7

Parameter Value	Description
0	Holdback Disabled
1	Deviation Low Holdback
2	Deviation High Holdback
3	Deviation Band Holdback

Note: DO NOT enable holdback for a loop that does not have hold back enabled in the configuration (seen on the program Entry screen). If holdback is set but not enable din the configuration, only the loop with holdback set will enter hold mode and cause the program to get out of sync between the loops resulting in a automated program run error.

*F The unit PV means that the unit of the parameter is the same as the unit of PV (the loop configuration).

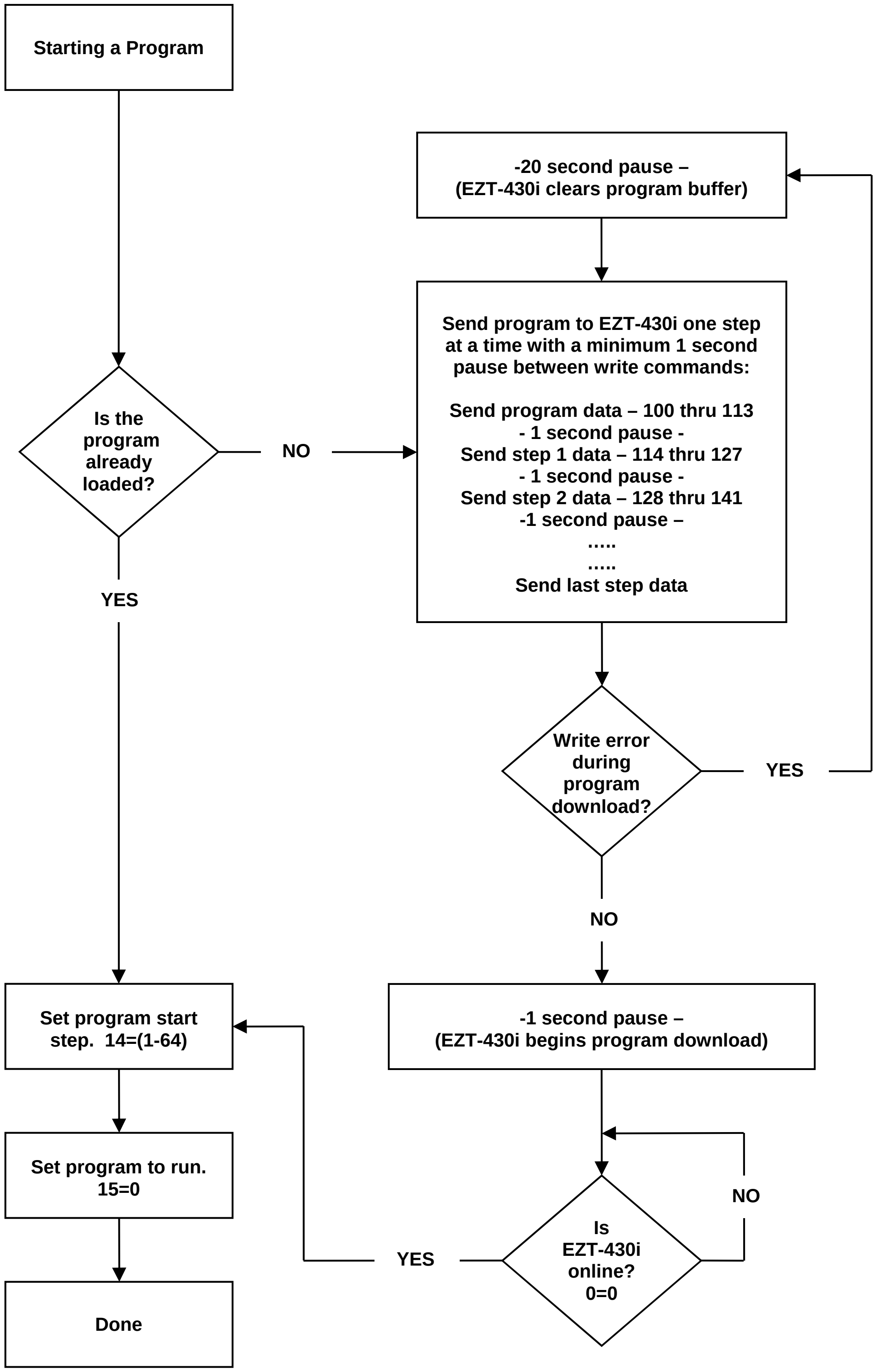
Use caution when loading an automatic program to EZT-430i . You must insure that the decimal point scaling and units of measurement in the program match the loop setting. Loading a program with a temperature setpoint of 80 will result in a control temperature of 80°F if the EZT-430i control is in degrees Fahrenheit. However, if the EZT-430i control is set for degrees Centigrade, it will result in a control temperature of 80°C (176°F).

5.2.1 Starting an Automatic Program on EZT-430i

Automatic programs are sent to EZT-430i in a step-by-step process. The download sequence must be followed in proper 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 EZT-430i (no response from EZT-430i or NACK returned), the program download must be aborted and restarted.

EZT-430i is automatically placed into program transfer mode when the first group of registers containing the program specific data is sent (registers 100-113). EZT-430i then begins looking for the number of steps of the program to be sent as was set in register 106. As each step is received, it increments the count. Once all steps have been received, EZT-430i downloads the program to the loop controller program memory. During this transfer, register 0 will be set to 1 to indicate that the process is taking place. Once the register value returns to zero, the automatic program is ready to be started.

If an error occurs during the transfer process from the PC to EZT-430i , the program transfer process should be stopped at the PC. The data sent to EZT-430i 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 EZT-430i properly. On the transmission error, EZT-430i 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.


Starting a Program
Is the program already loaded?
NO
YES
-20 second pause –
(EZT-430i clears program buffer)
Send program to EZT-430i one step at a time with a minimum 1 second pause between write commands:
Send program data – 100 thru 113
- 1 second pause -
Send step 1 data – 114 thru 127
- 1 second pause -
Send step 2 data – 128 thru 141
-1 second pause –
.....
.....
Send last step data
Write error during program download?
YES
NO
-1 second pause –
(EZT-430i begins program download)
Is EZT-430i online? 0=0
NO
YES
Set program start step. 14=(1-64)
Set program to run. 15=0
Done

6 EZT-430i Simulated F4S/D Interface Operation

The EZT-430i simulated Watlow F4S/D interface mode allows EZT-430i to appear as a Watlow F4 controller to Watlow's Watview software. This allows EZT-430i to be placed on an existing serial communication link with other F4 controllers already communicating with WatView or another software package. This eliminates the need for users already familiar with their existing software to learn another software package in order to communicate with EZT-430i . It also allows for the replacement or upgrade of older, individual F4 controllers without having to implement a complete changeover.

This section targets the functionality provided by the EZT-430i F4S/D interface, and not the specifics on how to interface with a Watlow F4S/D controller. It is assumed that users who choose to use this interface mode are already familiar with or have an existing F4 installation and wish to continue its use while integrating the EZT-430i into that installation base. For specifics on how to communicate with a Watlow F4 controller or to obtain the full list of communications registers, see the Series F4S/D User's Manual available from Watlow's website (http://www.watlow.com/downloads/en/manuals/Series%20F4SD_Rev%20H.pdf).

6.1 F4S/D Supported Control/Monitoring Data Registers

The following table of data registers represents active (functional) data registers for the F4 interface that are used by EZT-430i for control/monitoring. While EZT-430i provides the full register base of a Watlow F4S/D to insure compatibility with WatView, many of the registers are non-functional. The purpose of the interface is to allow control/monitoring and not configuration/setup of EZT-430i .

Register Address		Parameter Description	Data *A Type	Range *B		*C
				Low	High	Unit
100	**(0x0064)**	Input 1 Value, Status	R	-32768	32767	*C1
101	**(0x0065)**	Input 1 Error, Status	R	*B1	*B1	-
102	**(0x0066)**	Alarm 1, Status	R	*B2	*B2	-
103	**(0x0067)**	% Power Output 1A, Status	R	0	100.00	%
104	**(0x0068)**	Input 2 Value, Status	R	-32768	32767	*C1
105	**(0x0069)**	Input 2 Error, Status	R	*B1	*B1	-
106	**(0x006A)**	Alarm 2 , Status	R	*B2	*B2	-
107	**(0x006B)**	% Power Output 1B, Status	R	0	100.00	%
111	**(0x006F)**	% Power Output 2A, Status	R	0	100.00	%
115	**(0x0073)**	% Power Output 2B, Status	R	0	100.00	%
200	**(0x00C8)**	Operation Mode, Status	R	*B3	*B3	-
300	**(0x012C)**	Set Point 1, Value	R/W	-32768	32767	*C1
302	**(0x012F)**	Alarm Low Set Point and Deviation, Alarm 1	R/W	*B4	*B4	*C1
303	**(0x0130)**	Alarm High Set Point and Deviation, Alarm 1	R/W	*B4	*B4	*C1
305	**(0x0131)**	Autotune Channel 1	R/W	*B5	*B5	-

Register Address		Parameter Description	Data *A Type	Range *B Low	High	*C Unit
311	(0x0137)	Clear Error 1, Key Press Simulation	W	*B6	*B6	-
312	(0x0138)	Clear Alarm 1, Key Press Simulation	W	*B6	*B6	-
313	(0x0139)	Silence Alarm 1, Key Press Simulation	W	*B6	*B6	-
319	(0x013F)	Set Point 2, Value	R/W	-32768	32767	*C1
321	(0x0141)	Alarm Low Set Point and Deviation, Alarm 2	R/W	*B4	*B4	*C1
322	(0x0142)	Alarm High Set Point and Deviation, Alarm 2	R/W	*B4	*B4	*C1
324	(0x0144)	Autotune Channel 2	R/W	*B5	*B5	-
330	(0x014A)	Clear Error 2, Key Press Simulation	W	*B6	*B6	-
331	(0x014B)	Clear Alarm 2, Key Press Simulation	W	*B6	*B6	-
332	(0x014C)	Silence Alarm 2, Key Press Simulation	W	*B6	*B6	-
602	(0x025A)	Set Point Low Limit, Analog Input 1	R	-32768	32767	*C1
603	(0x025B)	Set Point High Limit, Analog Input 1	R	-32768	32767	*C1
606	(0x025E)	Decimal Point, Analog Input 1	R	0	3	-
608	(0x0260)	Process Units, Analog Input 1	R	*B7	*B7	-
612	(0x0264)	Set Point Low Limit, Analog Input 2	R	-32768	32767	*C1
613	(0x0265)	Set Point High Limit, Analog Input 2	R	-32768	32767	*C1
616	(0x0268)	Decimal Point, Analog Input 2	R	0	3	-
618	(0x026A)	Process Units, Analog Input 2	R	*B7	*B7	-
901	(0x006F)	°F or °C, System	R	0	1	-
1205	(0x04B5)	Guaranteed Soak Band, Channel 1	R/W	*B8	*B8	*C1
1209	(0x04B9)	Resume a Profile, Key Press Simulation	W	*B6	*B6	-
1210	(0x04BA)	Hold a Profile, Key Press Simulation	W	*B6	*B6	-
1212	(0x04BC)	Guaranteed Soak Band, Channel 2	R/W	*B8	*B8	*C1

Register Address		Parameter Description	Data *A Type	Range *B Low	High	*C Unit
1217	**(0x04C1)**	Terminate a Profile, Key Press Simulation	W	*B6	*B6	-
2000	**(0x07D0)**	Digital (Event) Output 1, Condition	R/W	0	1	-
2010	**(0x07DA)**	Digital (Event) Output 2, Condition	R/W	0	1	-
2020	**(0x07E4)**	Digital (Event) Output 3, Condition	R/W	0	1	-
2030	**(0x07EE)**	Digital (Event) Output 4, Condition	R/W	0	1	-
2040	**(0x07F8)**	Digital (Event) Output 5, Condition	R/W	0	1	-
2050	**(0x0802)**	Digital (Event) Output 6, Condition	R/W	0	1	-
3100 -thru- 3109	**(0x0C1C) -thru- (0x0C25)**	Name, Digital (Event) Output 1 (10 characters)	R	*B9	*B9	-
3110 -thru- 3119	**(0x0C26) -thru- (0x0C2F)**	Name, Digital (Event) Output 2 (10 characters)	R	*B9	*B9	-
3120 -thru- 3129	**(0x0C30) -thru- (0x0C39)**	Name, Digital (Event) Output 3 (10 characters)	R	*B9	*B9	-
3130 -thru- 3139	**(0x0C3A) -thru- (0x0C43)**	Name, Digital (Event) Output 4 (10 characters)	R	*B9	*B9	-
3140 -thru- 3149	**(0x0C44) -thru- (0x0C4D)**	Name, Digital (Event) Output 5 (10 characters)	R	*B9	*B9	-
3150 -thru- 3159	**(0x0C4E) -thru- (0x0C57)**	Name, Digital (Event) Output 6 (10 characters)	R	*B9	*B9	-
3200 -thru- 3209	**(0x0C80) -thru- (0x0C89)**	Name, Alarm 1 (10 characters)	R	*B9	*B9	-
3210 -thru- 3219	**(0x0C8A) -thru- (0x0C93)**	Name, Alarm 2 (10 characters)	R	*B9	*B9	-
3500 -thru- 3509	**(0x0DAC) -thru- (0x0DB5)**	Name, Profile 1 (10 characters)	R	*B9	*B9	-
4100	**(0x1004)**	Profile Number, Current Status	R	1	1	-
4101	**(0x1005)**	Profile Step Number, Current Status	R	1	64	-

Register Address		Parameter Description	Data *A Type	Range *B Low	High	*C Unit
4102	(0x1006)	Profile Step Type, Current Status	R	*B10	*B10	-
4111	(0x100F)	Digital (Event) Output 1, Current Status	R	0	1	-
4112	(0x1010)	Digital (Event) Output 2, Current Status	R	0	1	-
4113	(0x1011)	Digital (Event) Output 3, Current Status	R	0	1	-
4114	(0x1012)	Digital (Event) Output 4, Current Status	R	0	1	-
4115	(0x1013)	Digital (Event) Output 5, Current Status	R	0	1	-
4116	(0x1014)	Digital (Event) Output 6, Current Status	R	0	1	-
4119	(0x1017)	Hours Remaining, Ramp/Soak Step	R	0	99	-
4120	(0x1018)	Minutes Remaining, Ramp/Soak Step	R	0	59	-
4121	(0x1019)	Seconds Remaining, Ramp/Soak Step	R	0	59	-
4122	(0x101A)	Set Point Channel 1, Current Profile Status	R	-32768	32767	*C1
4123	(0x101B)	Set Point Channel 2, Current Profile Status	R	-32768	32767	*C1
4126	(0x101E)	Jump Count, Current Profile Status	R	1	10000	-

Notes:

Parameters descriptions with Watlow F4 designations of “channel 1”, “input 2”, etc. refer to EZT-430i data values for loop 1 or loop 2.

*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 are dependent upon system options. Consult the following range tables for information regarding the use of these parameters.

*B1

Parameter Value	Description
0	No error
5	Indicates loop input sensor trouble

*B2

Parameter Value	Description
0	No alarm
1	High alarm
2	Low alarm

Note: EZT-430i can provide up to 6 alarms depending upon configuration; however, the Watlow F4 is only equipped with two alarms. Thus, only EZT-430i alarms 1 and 2, if configured, are able to be monitored.

*B3

Parameter Value	Description
0	Profile off
1	Profile Pre-run
2	Profile Run
3	Profile Hold

Note: The profile “pre-run” value is used to indicate that EZT-430i is either offline or downloading a profile to the loop control boards. Therefore, do not attempt to write a profile to EZT-430i or perform control actions wile the profile mode is in “pre-run”.

*B4

Parameter Value	Deviation Alarm	Process Alarm
Range Low	-18000	SP Low
Range High	18000	SP High

*B5

Parameter Value	Description
0	Auto tune off
1	Auto tune on

Note: Auto tune may not be available depending upon the loop configuration. If auto tune operation is not available, auto tune will automatically turn off after being set. The Watlow F4 contains multiple PID sets which can be tuned while EZT-430i supports only one (PID Set 1).

*B6
Key press simulation values accept a write of any data value to perform their action.

*B7

Parameter Value	Description
0	Temperature
3	Process units

*B8

Parameter Value	Temperature Units °C	Temperature Units °F	Process Units
Range Low	1	1	1
Range High	555	999	999

*B9
Range limited to standard printable ASCII characters [0-9], [A –Z], etc.

*B10

Parameter Value	Description
1	Ramp Time
3	Soak
4	Jump
5	End

*C1 The units of measure and range of an input is dependant upon the configuration of the input and/or the units of temperature selection (Celsius or Fahrenheit) of EZT-430i . The decimal point position for the input is an implied value based on the decimal point selection for the input. Thus, a register value of 345 can represent an actual process value of 345, 34.5, 3.45 or 0.345 depending upon the decimal point configuration of the input.

6.2 F4S/D Supported Profile Step Data Registers

The following table of data registers provides the list of all data registers associated with downloading profiles to a Watlow F4S/D. Not all of the F4S/D functionality is provided by EZT-430i , so registers that are shown “grayed out” in the list represent those functions NOT supported by the EZT-430i automatic program operation.

EZT-430i is compatible with the profile editor of WatView or any other software that follows the proper sequence for downloading profiles to an F4. When a profile is downloaded to EZT-430i , it will automatically ignore the non supported functions so a special profile editor is not required. For the proper method of downloading profiles to an F4 controller, reference the Profile Programming Procedures in section 7, Communications, of the Watlow F4S/D User Manual.

Register Address		Parameter Description	Data *D Type	Range *E Low	High	*F Unit
4000	**(0x0FA0)**	Profile Number	R/W	1	1	-
4001	**(0x0FA1)**	Profile Step Number	W	1	64	-
4002	**(0x0FA2)**	Profile Edit Action	W	*E1	*E1	-
4003	**(0x0FA3)**	Profile Step Type	W	*E2	*E2	-
4004	**(0x0FA4)**	Autostart Profile Date or Day				
4005	**(0x0FA5)**	Autostart, Date (month)				
4006	**(0x0FA6)**	Autostart, Date (day)				
4007	**(0x0FA7)**	Autostart, Date (year)				
4008	**(0x0FA8)**	Autostart, Day (of week)				
4009	**(0x0FA9)**	Autostart/Ramp/Soak Time (hours)	W	0	99	*F1
4010	**(0x0FAA)**	Autostart/Ramp/Soak Time (minutes)	W	0	59	*F1
4011	**(0x0FAB)**	Autostart/Ramp/Soak Time (seconds)	W	0	59	*F1
4012	**(0x0FAC)**	Wait/Don’t Wait, Ramp/Soak Steps				
4013	**(0x0FAD)**	Wait For Event 1, Ramp/Soak Steps				
4014	**(0x0FAE)**	Wait For Event 2, Ramp/Soak Steps				
4015	**(0x0FAF)**	Wait For Event 3, Ramp/Soak Steps				
Register Address		Parameter Description	Data *D Type	Range *E Low	High	*F Unit
4016	**(0x0FB0)**	Wait For Event 4, Ramp/Soak Steps				
4021	**(0x0FB5)**	Wait For Analog 1, Ramp/Soak Steps				
4022	**(0x0FB5)**	Wait For Analog 1, Value, Ramp/Soak Steps				
4023	**(0x0FB6)**	Wait For Analog 2, Ramp/Soak Steps				
4024	**(0x0FB7)**	Wait For Analog 2, Value, Ramp/Soak Steps				

4025	**(0x0FB8)**	Wait For Analog 3, Ramp/Soak Steps				
4026	**(0x0FB9)**	Wait For Analog 3, Value, Ramp/Soak Steps				
4030	**(0x0FBE)**	Event Output 1, Ramp/Soak Steps	W	0	1	-
4031	**(0x0FBF)**	Event Output 2, Ramp/Soak Steps	W	0	1	-
4032	**(0x0FC0)**	Event Output 3, Ramp/Soak Steps	W	0	1	-
4033	**(0x0FC1)**	Event Output 4, Ramp/Soak Steps	W	0	1	-
4034	**(0x0FC2)**	Event Output 5, Ramp/Soak Steps	W	0	1	-
4035	**(0x0FC3)**	Event Output 6, Ramp/Soak Steps	W	0	1	-
4036	**(0x0FC4)**	Event Output 7, Ramp/Soak Steps				
4037	**(0x0FC5)**	Event Output 8, Ramp/Soak Steps				
4043	**(0x0FCB)**	Rate, Ramp Rate Step				
4044	**(0x0FCC)**	Ramp Set Point Channel 1, Ramp Time Step	W	-32768	32767	*F2
4045	**(0x0FCD)**	Ramp Set Point Channel 2, Ramp Time Step	W	-32768	32767	*F2
4046	**(0x0FCE)**	Channel 1 PID Set, Ramp/Soak Steps				
4047	**(0x0FCF)**	Channel 2 PID Set, Ramp/Soak Steps				
4048	**(0x0FD0)**	Guaranteed Soak Channel 1, Ramp/Soak Steps	W	0	1	-
4049	**(0x0FD1)**	Guaranteed Soak Channel 2, Ramp/Soak Steps	W	0	1	-
4050	**(0x0FD2)**	Jump to Profile, Jump Step				
4051	**(0x0FD3)**	Jump to Step, Jump Step	W	1	64	-
4052	**(0x0FD4)**	Jump Repeats, Jump Step	W	1	999	-
4060	**(0x0CSZ)**	End Action, End Step				
4061	**(0x0FDD)**	End Idle Setpoint Channel 1, End Step	W	-32768	32767	*F2
4062	**(0x0FDE)**	End Idle Setpoint Channel 2, End Step	W	-32768	32767	*F2

Notes:

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

*E1

Parameter Value	Description
1	Create profile
2	Insert step
3	Delete current profile
4	Delete step
5	Start profile
255	Delete all profiles

Note: EZT-430i only supports the create profile and start profile commands. When a value of 1 is written to the register, EZT-430i is placed in profile receive mode in which all following write commands are seen as profile data. Once the end step is detected, the profile receive mode will end.

When a value of 5 is written to the register, EZT-430i will start the currently loaded profile if it is not already running.

Any other non-supported command will be ignored.

*E2

Parameter Value	Description
1	Ramp time
2	Ramp rate
3	Soak
4	Jump
5	End

Note: EZT-430i does not support the ramp rate step type. DO NOT send profiles to EZT-430i with steps programmed for ramp rate or unexpected operation may occur. Use ramp time step types only for both local EZT-430i automatic programs and F4 programs downloaded remotely via a PC.

*F1 EZT-430i automatic program steps are entered locally using a time range of hours/minutes or minutes/seconds. EZT-430i cannot be programmed with step times of hours/minutes/seconds. Therefore, when programming remotely via an F4S/D profile editor, all ramp type steps and soak type steps must be programmed in either hours/minutes or minutes/seconds. EZT-430i will then select the proper range time range for ramp/soak steps when downloading to its loop control boards based on the maximum time programmed in any of the steps of that type in the profile.

For example, if all ramp steps have time entries of only minutes and seconds, EZT-430i will select time units of minutes/seconds for the profile ramp steps. However, if a ramp step is programmed with a non-zero entry for hours, all ramp steps will operate on a time base of hours and minutes. Any ramp steps that were then programmed with a minutes/seconds entry will ignore the seconds and only use the minutes. This may cause unexpected profile operation so be sure to program all profile ramp/soak steps accordingly.

*F2 The units of measure and range of an input is dependant upon the configuration of the input and/or the units of temperature selection (Celsius or Fahrenheit) of EZT-430i . The decimal point position for the input is an implied value based on the decimal point selection for the input. Thus, a register value of 345 can represent an actual process value of 345, 34.5, 3.45 or 0.345 depending upon the decimal point configuration of the input.

6.2.1 Limitations of Profile Download/Operation (IMPORTANT – Please Read!)

The section covers the rules of use when sending F4S/D based profiles to the EZT-430i controller. These rules must be followed in order to insure proper profile operation especially when using the Profile Editor tool of WatView software. WatView's operation can not be altered to comply with the manner in which EZT-430i operates, so it is up to the end user to insure that profiles are created and sent to EZT-430i properly.

6.2.1.1 <u>Sending Profiles to EZT-430i</u>

The Watlow F4S/D controller is capable of storing up to 40 profiles with up to 256 steps. EZT-430i stores profiles differently, therefore, only one profile (profile 1) can be sent to EZT-430i at a time. If another profile is sent, it overwrites the previous profile.

When using the WatView Profile Editor, DO NOT attempt to "read profile image from controller". Selecting this function will cause WatView to get stuck in an infinite loop and the program will have to be shut down manually and possibly require a reboot of the PC. EZT-430i does not support the read of a profile from its memory. Only profile download is supported.

Thus, when using WatView software or other profile editing software made for the F4, profiles must be stored on the PC as separate profile images with each containing a single profile. That single profile can then be downloaded when required. Note that profiles can not exceed 64 steps. If a profile exceeds 64 steps (including the end step), the download will fail without notice to the user and the previous profile will remain loaded in memory on EZT-430i.

Also, due to limitations of WatView software, the profile name shown on control and status screens in WatView, do not reflect the actual profile name currently running in EZT-430i. If a user were to select and run a profile locally at EZT-430i, the name field data registers of the serial interface are updated to show the correct profile name; however, WatView does not read them and update its display fields dynamically. It only uses the names of the profile image currently loaded in WatView for the controller. Thus, the user must be sure that when a profile is downloaded or started from WatView, that the local EZT-430i interface displays the correct profile name to insure the desired profile is running.

When a profile is sent to EZT-430i over the serial interface, upon receiving the last step, EZT-430i must then download the profile to its loop control boards. During this process, the operating mode register (200) will indicate a pre-run status (1) for the profile. When the profile state returns to profile off (0), the download is complete and the profile can be started.

6.2.1.2 Setting Proper Step Times

Ramp rate steps are not supported by EZT-430i, even if configured as a single loop controller (F4S). Thus, all profiles must be programmed with ramp time steps, either locally at EZT-430i or via the Profile Editor of WatView. In addition, EZT-430i ramp/soak steps must be configured in hours/minutes or minutes/seconds only. Steps can not be programmed with an entry of hours/minutes/seconds.

In order for EZT-430i to run the profile correctly, it automatically configures itself to use the best time format based on the programming for all ramp and soak steps. For example, if all ramp steps have only entries for minutes and seconds (all hour entries are 0), EZT-430i will select minutes/seconds for the ramp step time format. However, if a single ramp step has a non-zero entry for hours, it will use the hours/minutes status registers for operation. This causes any steps that were configured with only minutes and seconds, to discard the seconds entry and only use the minutes. This can cause unexpected profile operation.

It is then necessary to insure that when a profile is created using the Profile Editor of WatView, that all ramp steps use the same entry fields for time and all soak steps use the same entry fields for time. That means that if any step of your profile must operate for at least 1 hour, you will have to use hours/minutes for all other steps of that type. Any entries for seconds will be ignored. Note that a step time of 0 can be entered under any circumstance, so if you have to use the hours/minutes range and a step time of 1 minute is too long, you can enter 0.

6.2.1.3 Setting Final Setpoints

If EZT-430i is configured to end its automatic program with a final setpoint, you must set the F4 end step type to idle. This then allows you to set the idle setpoints of the end step in the WatView Profile Editor. If you do not do this, the setpoints default to their low limit value and the system will assume these setpoints upon the end step of the program.

6.2.1.4 Starting a Profile

When starting a profile, there is no need to set the profile number as EZT-430i has only one profile loaded and available for run at a time. It is then only necessary to send the step to start the profile on (4001 = 1 to 64)), and then set the profile to run (4002 = 5).