TENNEY ENVIRONMENTAL

DATA COMMUNICATIONS OPERATION MANUAL

VERSATENN III CONTROLLER

For Temperature / Humidity Chambers

LUNAIRE LIMITED
A SPX Company
4 Quality Street
Williamsport, PA   17701
Tele. 570 - 326 - 1770   Fax: 570 - 326 - 7304
www.lunaire.com

Rev. 05-29-01

TABLE OF CONTENTS

1.0 INTRODUCTION TO DATA COMMUNICATIONS

Preface:

The following instructions will be your guide to communicate with the VersaTenn Controller via a computer. A section is provided, which briefly describes the different communications options that are supported by Lunaire Limited. Most of this manual is devoted to the actual commands the VersaTenn responds to. This manual does not go into any details of how to use the VersaTenn Controller itself, or the actual function of a particular command. The commands that are described here are all features the user has through the front panel of the instrument. Before attempting to communicate with the VersaTenn using a computer, the user is urged to become familiar with the functions of the controller and the operation of the chamber it controls.

Introduction:

In spite of our so-called “Age of Information”, communication is as old as life itself. It had first chemical and biological, then linguistic, and finally electronic means for information exchange. The term “communication” comes from an old Latin root meaning “common”, or that which is shared.

Sharing information is a way for individuals or systems to cooperate and benefit to work together towards a common goal. However, the evolution of information sharing is in constant flux. In the overall scheme of things the rate of communications expansion is increasing tremendously.

The main reason for this exponential increase is the computer, a machine using a vast number of simple “on-off”, or 1 and 0 electronic circuits to answer complex questions and to perform extensive tasks rapidly. Just as we must communicate, we want our computers and controllers to converse as well. In fact, nearly everything electronic today can be linked to a computer.

Elements of Machine-to-Machine Communication:

In human communication there are basic ways and means to get a message across. This is also true with computers and controllers. They need a code to talk with, called a “character format, or character set”. They need a common data link, or interface to communicate through. They need rules, or protocol to govern their talk, to prevent confusion and errors. Let’s examine these elements of a data communication, plus some other.

Character Format:

The code or “character format” for data communication is shared by virtually everyone in the electronics business. This code defines a computer stream of 1’s and 0’s that are created by simply varying a voltage signal in a regular manner. The code is the American Standard Code for Information Interchange, called ASCII.

Bits and Bytes:

ASCII consists of bits and bytes. Bits are like the separate pen strokes we make for a single printed letter of the alphabet. A byte is like the whole letter.

“Bit” is simply the contraction of the words “binary digits”. A bit is the basic unit in ASCII. It is either a 1, or a 0. A byte is a string of seven or eight bits, which a computer treats as a single character. ASCII requires seven bits to represent each letter of the alphabet, each digit and each punctuation mark we use.

Interface:

An interface is a means for electronic systems to interact. It is a specific kind of electrical wiring configuration, if you will. As two computers talk, they must use a connecting interface to come together.

Serial Communication:

The interfaces we've chosen employ serial communication. Serial communication 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.

Protocol:

Up to this point we've looked at a code, a way to link codes together, and the sequence that we'll send each bit of information. Now we need a few rules to talk by. The who-gets-to-talk-when of data communications is the communication protocol, or formal etiquette. A protocol is a set of standards for formatting and timing information exchange between electronic systems.

Protocol assists computers in hand shaking and saying, "talk to you later". It also prevents two machines from attempting to talk at the same time. There are a number of different protocols, just as there are different human cultural protocols around the world.

Protocols are like manners. An Arab tribal sheik would have as much difficulty with the correct protocol at his first White House dinner as would a White House aide with his first ceremonial tribal feast in the Saharan desert. Anywhere in the world, the correct protocol is always the host's. That's just good manners. The same is true in data communications.

The protocol part of data communications is very important, because it gives a high quality of communication that you would not have if it were not employed. Protocol-driven communications are more accurate, because they are less prone to both operator and noise errors.

A Protocol Example:

Let's assume that we have a computer and several microprocessors linked together. They all use ASCII and are connected via a common interface and data cables. In process control, one device often has greater function and memory capability than the devices it is communicating with. This host computer, or master device, always initiates the talk between it and the connected slave devices.

Here's what happens:

Imagine PC-1, the host computer, sitting at the end of a long hallway with nine doors spaced along it. Every door has a slave device behind it; PC-1 has a telephone party line connecting itself to them. The slave devices are busy controlling heaters to specific points. PC-1 is in business to monitor and adjust the heaters, which are part of a large chamber.

By your request, PC-1 wants to talk with the slave device S-2 on the line, and inquires if S-2 has time to talk. This electronic knocking on S-2's door is the "Connection".

Three scenarios may occur at PC-1's ring:

1.) S-2 answers saying "This is S-2, go ahead". PC-1 then begins to talk.
2.) S-2 answers and says, "I'm too busy to talk now, please try later". PC-1 may try again immediately, or it may wait awhile.
3.) S-2 does not answer. PC-1 must then try again, or presume that S-2 is out to lunch.

Let’s take the best case scenario. Here’s a simple version of what happens:

- ♦ S-2 answers and hears PC-1 say: “Hello S-2. Do you have time to talk?”
- ♦ S-2 acknowledges PC-1 with “S-2 here, go ahead”.
- ♦ PC-1 then sends an ASCII-encoded message instructing S-2 to make a change in setpoint to 170 degrees C. (Message)
  When PC-1 is finished with its message, it says in effect, “That’s all, your turn”.
- ♦ S-2 replies “OK”, and carries out the instruction.
- ♦ S-2 then takes the protocol lead, and tells PC-1 “The new setpoint is 170 degrees C”. (Message)
- ♦ PC-1 says “OK”
- ♦ S-2 says “That’s all, your turn”.
- ♦ PC-1 then takes the Protocol lead and says “Thank you, that’s all”.
- ♦ S-2 hangs up. (Disconnect)

That’s basically how the Connect, Message, and Disconnect protocol works in Lunaire data communications. The hallway in this example is really a communication bus, a common connection among a number of separate devices. The term applied to a communications system with multiple devices on a common bus is **Multidrop**.

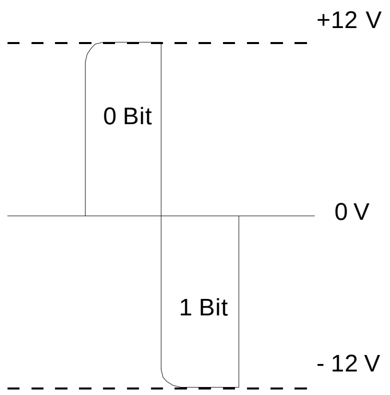
Why go through this exacting connect-message-disconnect procedure? It is done to insure that nothing is lost to noise, or operator error as the devices pass information back and forth. We must be sure that our system has integrity.

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 slip indicating that the receiver signed.

In Lunaire data communications, a dialogue 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 operator or the host 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.

2.0 RS-232, RS-423, RS-422, and RS-485 INTERFACES

RS-232



Bit signals on an RS-232 interface

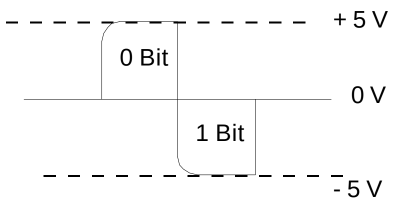
An RS-232 interface uses three wires; a single transmit wire, a single receive wire, and a ground. A signal of +/-12 volts establishes communication between two devices only.

A ‘1 Bit” through RS-232 is a -12 volt signal. Conversely, a ‘0 Bit’ is a +12 volt signal. The RS-232 signal is referenced to ground rather than to a separate wire as in RS-422. RS-232 is limited in its cable length ability and may be subjected to noise. Protocol with RS-232 is a simple X-ON / X-OFF, which are Transmitter ON / Transmitter OFF control characters that start or stop communication. With only two devices on-line there is no need for addressing, thus a simple protocol.

RS-423

The RS-423 interface is similar to RS-232. It uses a +/- 5 volt signal with one wire each for the transmit and receive signals.

The RS-423 interface is also referenced to ground with a third wire. A ‘1 Bit’ through RS-423 is a - 5 volt signal. Conversely, a ‘0 Bit’ is a + 5 volt signal. Almost without exception, an RS-423 interface is compatible with an RS-232 device. The RS-232 device will accept the similar signal with its lower voltage span. Like RS-232, RS-423 is also limited to one device at each end of the line, and addressing is not part of the protocol.

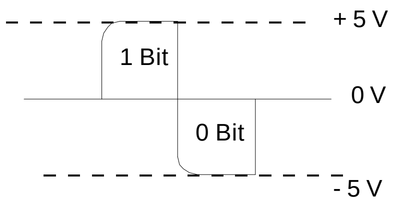


Bit signals on an RS-423 interface

RS-422

The RS-422 interface uses five wires; a talk pair, a listen pair, and a common mode or ground. It can handle up to ten connections in a “multidrop” network. (Remember the “hallway” and “doors” in the protocol examples.)

To pass a ‘1 Bit’, or a ‘0 Bit’ along it’s wires, RS-422 uses the difference in voltage between the two wires, always 0 to 5 volts. A 1 is a difference of +5 volts, while a 0 is a difference of -5 volts. Protocol for RS-422 includes an address for each device on the network.

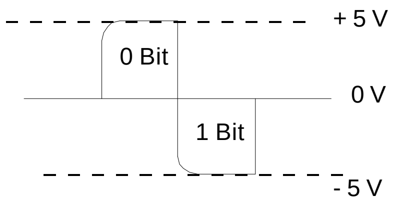


Bit signals on an RS-422 interface

RS-485

The RS-485 interface uses a two wire (half duplex) system. Both lines are used for transmitting and receiving.

Only one device, the computer or the controller, can be speaking at a time. There is a 1 millisecond delay between transmission and receipt of data. The RS-485 is a multidrop device network allowing up to 32 devices on the network, with a maximum of 4000 feet total network distance.



Bit signals on an RS-485 interface

3.0 IEEE - 488 INTERFACE

Up to now we have discussed four different interfaces, all of which use serial data transfer between the host and slave. Serial communications methods are the only ones that all Lunaire programmable controllers use for data transfer. Lunaire supports the IEEE-488 interface with any of these programmable controllers by using a “black box” between the Lunaire controller and the computer. The “black box” is a transparent IEEE-488 Bus-to-Serial Interface. When addressed as a listener, it inputs data bytes from the Bus, buffers them, and then outputs them via the serial interface. In the reverse direction, the unit receives serial data, converts the characters into eight-bit bytes, buffers the data, and then handshakes the data onto the Bus when addressed as a talker. Communication with the controller can be achieved using either the ANSI X3.28 (**Tenney default shown below**), or X-ON/X-OFF protocol.

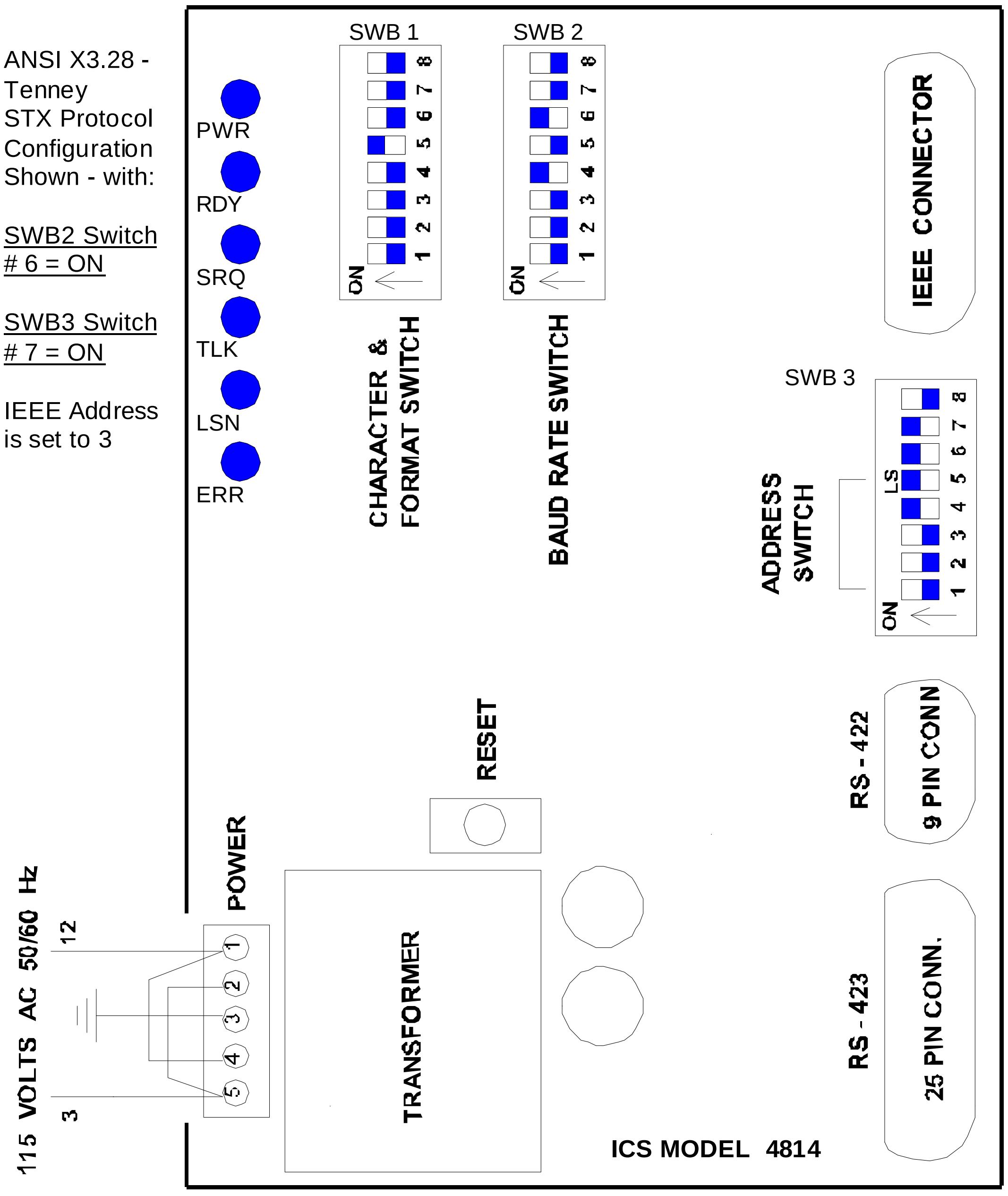
IEEE - 488 INTERFACE CARD

ANSI X3.28 - Tenney STX Protocol Configuration Shown - with:

SWB2 Switch # 6 = ON

SWB3 Switch # 7 = ON

IEEE Address is set to 3



Note: For the X-ON / X-OFF protocal, bit switch SWB2 Switch # 6 = OFF, and SWB3 Switch # 7 = OFF.

Data / Message Transfer:

The IEEE-488, or GPIB as it is commonly referred to, provides a means of transferring data and commands between groups of devices. The interface functions for each device are contained within that device itself, so only passive cabling is needed to interconnect the devices. The cables connect all instruments, controllers, and other components of the systems in parallel to the signal line. Eight of the lines (DI01-DI08) are reserved for the transfer of data and other messages in a byte-serial, bit-parallel manner. Data and message transfer is asynchronous, coordinated by the three handshake lines (DAV, NRFD, NDAC). The other five lines control Bus activity.

Two types of messages are transferred over the Bus:

- ♦ Interface Messages: For Bus management
- ♦ Device - Dependent Messages: For device control and data transfer

Devices connected to the Bus may act as talkers, listeners, controllers, or combinations of the three functions, depending upon their internal capability. The system controller is a controller that becomes active at power turn-on. It is the Bus manager and the initial controller-in-charge.

Device Functions:

Talker: Talkers send device-dependent messages, i.e., data, status.

Listener: Listeners accept interface messages, Bus commands, and device-dependent messages, i.e., setup commands, data.

Controller: Controller send interface messages to manage the other devices, address devices to talk or listen, and command specific actions within devices.

Bus systems can be as simple as two devices, a talker and a listener. Larger systems can have one or more controllers and many devices (the IEEE-488 driver specifications limit the total number of units on the Bus system to 15). Only one controller can be the controller-in-charge at any given time. Control originates with the system controller and is passed to the other controller(s) as required. Control can be passed back to the system controller or to another controller after the completion of a task. The system controller has the capability of taking control back at any time and resetting all addressed devices to their unaddressed state.

Each Bus device is identified by a five-bit binary address. There are 31 possible primary addresses, 0 through 30. Address 31 is reserved as the untalk or unlisten command. Some devices contain subfunctions, or the devices themselves may be addressed by a secondary five-bit binary address immediately following the primary address, i.e., 1703. This secondary address capability expands the Bus address range to 961 addresses. Most Bus addresses are set at the time the system is configured by rocker switches, which are typically located on rear panel of each device.

Information is transmitted on the data lines under sequential control of the three handshake lines. No step in the sequence can be initiated until the previous step is completed. Information transfer proceeds as fast as the devices respond (up to 1MB), but no faster than that allowed by the slowest addressed device. This permits several devices to receive the same message byte at the same time. Although several devices can be addressed to listen simultaneously, only one device at a time can be addressed as a talker. When a talk address is put on the data lines, all other talkers are normally unaddressed.

4.0 ASCII CHARACTER FORMAT

Format Description:

As you examine ASCII more closely, you'll learn that this binary (two-condition) code defines 128 separate 7-bit characters, one for each letter and digit that we use in speaking and writing, and one for a number of punctuation characters.

ASCII uses several control characters similar to those we find on a computer keyboard, like "backspace", "shift", and "return". It also has ten communications control characters for Identification, Enquire (inquire), Start of Text, End of Text, End of Transmission, Acknowledge, Negative Acknowledge, Escape, and so on. Because binary numbers are long and difficult to read, the ASCII code is usually interpreted in a 16 base number system called **Hexadecimal**, "**HEX**" for short. The first ten digits of this system are represented by the numbers 0 to 9, and the final six digits are represented by the letters A through F.

In the 10 base number system that we use everyday, the 1's column uses the digits 0-9 and the 10's column starts over with the 1, and so on. In hexadecimal, we need to think of a 16's column instead of a 10's column. It's not crucial to learn hexadecimal to understand data or data communications, but being aware of it will boost your ability to communicate with the engineers and technicians who use it every day. The chart on the next page represents the 128 ASCII character code with the decimal and hexadecimal equivalents. Notice how a computer keyboard "shift" key would move between adjacent or alternate columns to transmit a lower case "b" and an upper case "B", or between an "!" and the digit "1".

ASCII CHARACTER SET

Dec	Hex	Abbr.	Dec	Hex	Abbr.	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	16	10	DLE	32	20		48	30	0
1	01	SOH	17	11	DC1	33	21	!	49	31	1
2	02	STX	18	12	DC2	34	22	“	50	32	2
3	03	ETX	19	13	DC3	35	23	#	51	33	3
4	04	EOT	20	14	DC4	36	24	$	52	34	4
5	05	ENQ	21	15	NAK	37	25	%	53	35	5
6	06	ACK	22	16	SYN	38	26	&	54	36	6
7	07	BEL	23	17	ETB	39	27	‘	55	37	7
8	08	BS	24	18	CAN	40	28	(	56	38	8
9	09	HT	25	19	EM	41	29	)	57	39	9
10	0A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	0B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	0C	FF	28	1C	FS	44	2C	,	60	3C	<
13	0D	CR	29	1D	GS	45	2D	-	61	3D	>
14	0E	SO	30	1E	RS	46	2E	.	62	3E	=
15	0F	SI	31	1F	US	47	2F	/	63	3F	?

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	DEL

ASCII Control Characters:

There are several ways to transmit ASCII communication control characters. One way is to use the computer “Control” key. Look at the abbreviated ASCII table below. When you type the upper case letter while simultaneously depressing the Control key, you transmit an ASCII control character. Another way is to assign the hex or decimal equivalent in a program, and key the assigned characters into the computer. Or simpler yet, a single digit or letter in a program will also do the job. How ever the control characters are sent, they will always mean the same thing in the protocol.

ASCII Char	Ctrl Key Equiv.	Dec. Equiv.	Hex Equiv.	Definition
ENQ	CtrlE	05	05	Enquiry
ACK	CtrlF	06	06	Acknowledge
NAK	CtrlU	21	15	Neg. Acknowledge
STX	CtrlB	02	02	Start of Text
ETX	CtrlC	03	03	End of Text
EOT	CtrlD	04	04	End of Transmission
DLE	CtrlP	16	10	Data Link Escape
LF	CtrlJ	10	0A	Line Feed
CR	CtrlM	13	0D	Carriage Return
XON	CtrlQ	17	11	X-On
XOFF	CtrlS	19	13	X-Off

ASCII Control Character Definitions:

The ASCII control characters enable device transmission through their use with protocol.

ID# Identification Number: (Not an ASCII control character). The specific identity of each device on a multidrop network.

EOT End of Transmission: Concludes one or more messages. Precedes exchange of the protocol leadership.

ENQ Enquiry (inquiry): A request by the master for a response from a slave.

ACK Acknowledge: An affirmative response from the receiver.

NAK Negative Acknowledge: A negative response from the receiver.

STX Start of Text: Precedes any message from the sender.

ETX End of Text: Follows any message from the sender.

DLE Data Link Escape: Disconnect signal from the master to device on the network.

LF Line Feed: Advances to the same character on the next line, but does not affect its column.

CR Carriage Return: Same function as a carriage return on a typewriter.

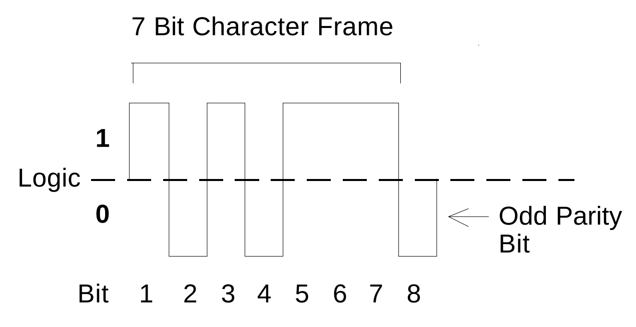
XON X-On: Sent by the receiver when it is ready for more data.

XOFF X-Off: Sent by the receiver when its buffer is full and it is not ready to receive data.

Parity Bit:

Remember that ASCII is a seven or eight bit code. What about that eighth bit? It's called the "parity" bit. A parity bit is added to the ASCII character to check the accuracy of the first seven bits. Here again we want system integrity. Here's how:

We are declaring that the number of 1's in the 8-bit character frame will be either always odd, or always even. To do that, about half the time we'll have to add another 1 to get an odd or even number of ones. The other half of the time we'll need to add a 0 so we don't change the total number of 1's. That way we can detect a single error in the seven-bit group.



Transmitted Upper Case "W" (Binary 1010111)

Take a look at the representation of the transmitted upper case "W" in the example above. In this case we have selected ODD parity. The number of 1's in the first seven bits, plus the parity bit, must always total an odd number. The total number of 1's in the binary character 1010111 (W) is 5, already an odd number. Thus, our parity bit in this case will be a 0. If we were transmitting the lower case "w" (binary 1110111), the parity bit would be a 1 because the total number of 1's in the character frame is 6, an even number. Adding the parity bit makes it odd, and consistent with the odd parity rule.

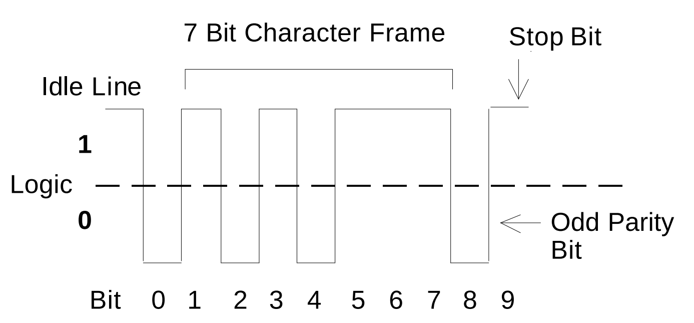
If a noise spike came onto the data line and changed the signal voltage level enough to reverse a 1 to a 0 in the character frame, the receiver would detect that error. The total number of 1's would be even and a violation of the odd parity rule.

There are other kinds of parity. The table below lists their rules. A "space" in data communication is a logical 0, while a "mark" is a logical 1. **At Tenney Environmental, we default to Odd Parity.**

PARITY TYPES	
Odd	Parity bit is logical zero if the total number of logical 1's in the first 7 data bits is odd.
Even	Parity bit is logical zero if the total number of logical 1's in the first 7 data bits is even.
Mark	Parity bit is always logical 1 (High or Mark Parity).
Space	Parity bit is always logical 0 (Low or Space Parity).
None / Off	Parity bit is always ignored.

Start and Stop Bits:

These are the first and last bits in a total character group. They inform the receiving device that a character is coming, or that one is complete. The “start bit” is always a 0, or space. The “stop bit” is always a 1, or mark. We’ve added the start and stop bits to the transmitted “W” example. The human speaking 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 assist the listener in comprehending the message.



Upper Case “W” with Start and Stop Bits

Baud Rate:

Baud rate is speed of data transmission. “Baud” is the unit of transmission speed. Baud rate is equal to bits per second (bps). Our standard baud rates are 1200 baud (default), 2400 baud, 4800 baud, and 9600 baud.

Computer Language:

A computer language is simply a set of symbols and rules for their use. Most computer languages use ASCII to make those symbols. Languages enable machines to function, and also to communicate. There are many computer languages, and a wide variety of applications for them. Programmers use languages to enable computers to do real work.

Syntax:

Syntax for the English language dictates how we can put words together to make phrases and sentences. Similarly, syntax in computer language is the actual phrase of code words and “delimiters” (spaces, commas, etc.) that tells the processor what to do. In data communications, syntax is a specific mnemonic (shortened) message for information or data entry.

For example, the VersaTenn parameter for setpoint information on Channel 1 is SP1. The VersaTenn control panel displays the setpoint information whenever you physically press the VersaTenn’s Mode key to reach SP1 in the parameter sequence. With a computer-linked VersaTenn, SP1 is part of the syntax for data communication.

If you just type “SP1” on the computer keyboard, the VersaTenn won’t respond to your computer with the current setpoint 1 data. The syntax requires spaces and “fields” of specific size to be complete. Plus, we need to add the protocol. It’s like putting the message in an envelope so you can mail it. The entire syntax of the SP1 command includes the message protocol’s STX (Start of Text) character, =, space, SP1, space, up to four decimal places of setpoint data, and a protocol ETX (End of Text).

The whole syntax phrase would look something like this: (RS-422 protocol)

STX= SP1 0500ETX

A Data Communications Conversation with ANSI X3.28 Protocol:

The example below uses ASCII control characters (represented here by abbreviations) to send a setpoint data command to the controller. This example uses the ANSI X3.28-1976, Subcategory 2.2-A3 protocol (Tenney STX protocol). Only the “protocol leader” may send a message. Every message requires a response. **Every message must include a start of transmission <STX> character and an end of transmission <ETX> character.**

	MASTER PC		SLAVE
A	**0ENQ** “#0, are you there?”	**B**	**0ACK** “I’m #0, I’m here”
C	**STX= SP1 500ETX** a) “Here comes a message” b) “Make SP1 = 50.0 deg. C” c) “I’m done with the message”	**D**	**ACK** “I understand”
E	**STX? SP1ETX** a) “Here comes a message” b) “What is SP1’s value?” c) “I’m done with the message”	**F**	**ACK** “I understand”
G	**EOT** “That’s all go ahead” Protocol lead goes to slave	**H**	**STX500ETX** a) “Here comes the answer” b) “The value is 50.0” c) “I’m done”
I	**ACK** “I understand”	**J**	**EOT** “That’s all, go ahead” Protocol lead goes to master
K	**DLEEOT** “Disconnect, that’s all”		

A Data Communications Conversation with X-ON / X-OFF Protocol:

This example basically follows the one above. **This protocol requires a Carriage Return <CR> character at the end of every message.**

	MASTER PC		SLAVE
A	**= SP1 500CR** a) “Here comes a message” b) “Make SP1 = 50.0 deg. C” c) “I’m done with the message”	**B**	**XOFFXON** a) “Don’t send anything” b) “OK, now ready”
C	**? SP1CR** a) “Here comes a message” b) “What is SP1’s value?” c) “I’m done with the message”	**D**	**XOFF500CRXON** a) “Don’t send anything” b) “The value is 50.0” c) “OK, now ready”

Remember there is no “addressing” in the X-ON / X-OFF protocol. Therefore, only one controller can be connected to the computer.

A Data Communications Conversation with IEEE-488:

On power-up or after reset, device clear, selected device clear, or interface clear, the user must first send a single ASCII number 0 – 9 (representing the VersaTenn's ID). When the VersaTenn receives the ID number it will respond with the ASCII - ID number followed by the ASCII string “ACK”, terminated with a CRLF.

To output a command to the VersaTenn, send the command string terminated with EOI, CR, or LF. If the command is a read request, the buffer of the IEEE circuitry will contain the data requested, otherwise the buffer will contain the ASCII “ACK” if a valid command, or an ASCII “NAK” if the VersaTenn did not understand the command. All data will be terminated by a CRLF.

On receiving an IFC, device clear, or selected device clear from the Bus, the ICS-4814 will send the 2 disconnect control characters DLE (10h) and EOT (04h) to the VersaTenn.

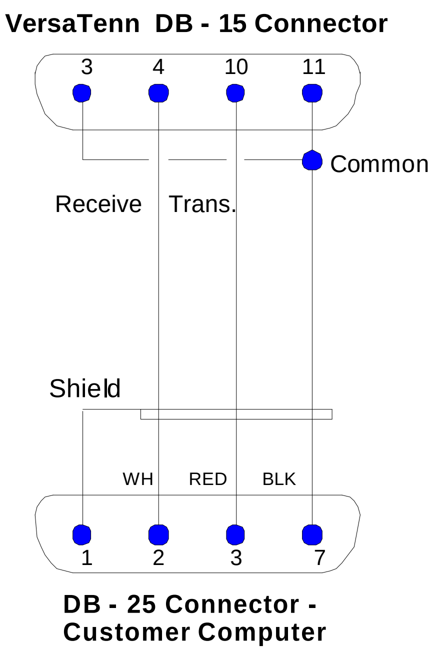
	<u>**MASTER PC**</u>		<u>**SLAVE**</u>
A	**“0”CRLF** “#0, are you there?”	**B**	**“0ACK”CRLF** “I'm #0, I'm here”
C	**“= SP1 500CRLF”** a) “Here comes a message” b) “Make SP1 = 50.0 deg. C” c) “I'm done with the message”	**D**	**“ACK”CRLF** “I understand”
E	**“? SP1”CRLF** a) “Here comes a message” b) “What is SP1's value” c) “I'm done with the message”	**F**	**“500”CRLF** “The value is 50.0”

5.0 INTERFACE DEFAULT SETTINGS

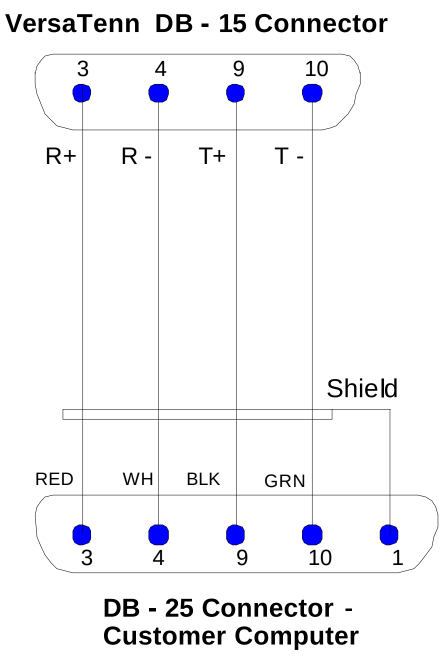
RS - 232C:	Protocol	ANSI X3.28
	ID	0
	Baud Rate	1200
		Odd Parity, Seven Bits, One Stop Bit
RS - 423:	Protocol	X-ON / X-OFF
	ID	not used
	Baud Rate	1200
		Odd Parity, Seven Bits, One Stop Bit
RS - 422:	Protocol	ANSI X3.28
	ID	0
	Baud Rate	1200
		Odd parity, Seven Bits, One Stop Bit
EIA - 485:	Protocol	ANSI X3.28
	ID	0
	Baud Rate	1200
		Odd Parity, Seven Bits, One Stop Bit
IEEE - 488:	Protocol	RS - 422
	Address	3
	ID	0
	Baud Rate	1200
		Odd Parity, Seven Bits, One Stop Bit

6.0 INTERFACE WIRING DIAGRAMS

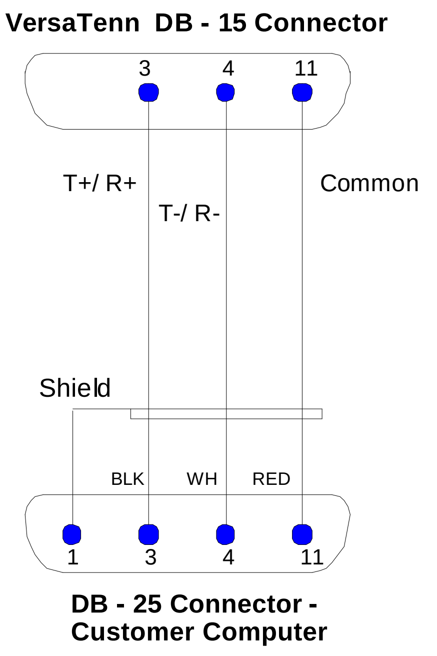
RS232 / 423 Interface Wiring Diagram



RS 422 Interface Wiring Diagram



RS - 485 Interface Wiring Diagram



7.0 VERSATENN " = " COMMAND INSTRUCTION SET

The ' = ' command is used to set a specific parameter to a specific value. The parameters available for use with this command are listed in the explanations that follow. This command has the following message syntax:

= <space> **ARG.1** <space> **ARG.2** <space> . . . **ARG.N**

Where:

< >	=	Indicates description of a character in non-literal form.
ARG.1	=	The parameter name
ARG.2 . . . ARG.N	=	A value to which the parameter given in ARG.1 is to be set equal to, or a modifier for ARG.1. ARG.2 . . . ARG.N should be a decimal or a hex number, the limits of which can be found in the following sections.

The following parameter names are available for use as ARG.1 . . . ARG.N:

1)	SP1	**19)**	GS	**37)**	L7	**55)**	RT1C
2)	SP2	**20)**	TI	**38)**	L8	**56)**	RT1H
3)	EV1	**21)**	RTD	**39)**	L9	**57)**	RT2C
4)	EV2	**22)**	4 - 20	**40)**	L11	**58)**	RT2H
5)	EV3	**23)**	LOCK	**41)**	L12	**59)**	RB1C
6)	EV4	**24)**	R1H	**42)**	L14	**60)**	RB1H
7)	EV5	**25)**	R1L	**43)**	L15	**61)**	RB2C
8)	EV6	**26)**	R2H	**44)**	OT11	**62)**	RB2H
9)	LEV1	**27)**	R2L	**45)**	OT18	**63)**	CT1C
10)	LEV2	**28)**	A1H	**46)**	AT1H	**64)**	CT1H
11)	ON	**29)**	A1L	**47)**	PB1C	**65)**	CT2C
12)	OFF	**30)**	A2H	**48)**	PB1H	**66)**	CT2H
13)	STP	**31)**	A2L	**49)**	PB2C	**67)**	DB1
14)	STRT	**32)**	CAL1	**50)**	PB2H	**68)**	DB2
15)	RSUM	**33)**	CAL2	**51)**	RS1C	**69)**	ALT
16)	HOLD	**34)**	L3	**52)**	RS1H	**70)**	VCMP
17)	CLRF	**35)**	L4	**53)**	RS2C	**71)**	CMS
18)	CF	**36)**	L6	**54)**	RS2H	**72)**	SYRS

8.0 VERSATENN " ? " COMMAND INSTRUCTION SET

The ' ? ' command is used to find the value of a specific parameter. The parameters available for use with this command are listed in the explanations that follow. This command has the following message syntax:

? <space> **ARG.1** <space> **ARG.2** <space> . . . **ARG.N**

Where:

< >	=	Indicates description of a character in non-literal form.
ARG.1	=	The parameter name
ARG.2 . . . ARG.N	=	A modifier of the parameter name

The following parameter names are available for use as ARG.1 . . . ARG.N:

1)	C1	**23)**	TI	**45)**	L3	**67)**	RS1H
2)	C2	**24)**	RTD	**46)**	L4	**68)**	RS2C
3)	SP1	**25)**	4 - 20	**47)**	L6	**69)**	RS2H
4)	SP2	**26)**	LOCK	**48)**	L7	**70)**	RT1C
5)	EV1	**27)**	DIP	**49)**	L8	**71)**	RT1H
6)	EV2	**28)**	ALM	**50)**	L9	**72)**	RT2C
7)	EV3	**29)**	ER1	**51)**	L11	**73)**	RT2H
8)	EV4	**30)**	ER2	**52)**	L12	**74)**	RB1C
9)	EV5	**31)**	1LO	**53)**	L14	**75)**	RB1H
10)	EV6	**32)**	1HI	**54)**	L15	**76)**	RB2C
11)	LEV1	**33)**	2LO	**55)**	OT11	**77)**	RB2H
12)	LEV2	**34)**	2HI	**56)**	OT18	**78)**	CT1C
13)	RUN	**35)**	R1H	**57)**	OT0	**79)**	CT1H
14)	MTR	**36)**	R1L	**58)**	OT1	**80)**	CT2C
15)	AFL	**37)**	R2H	**59)**	OT2	**81)**	CT2H
16)	FST	**38)**	R2L	**60)**	OT3	**82)**	DB1
17)	MDL	**39)**	A1H	**61)**	AT1H	**83)**	DB2
18)	STP	**40)**	A1L	**62)**	PB1C	**84)**	ALT
19)	RJ	**41)**	A2H	**63)**	PB1H	**85)**	VCMP
20)	EI	**42)**	A2L	**64)**	PB2C	**86)**	INP
21)	CF	**43)**	CAL1	**65)**	PB2H	**87)**	INP1
22)	GS	**44)**	CAL2	**66)**	RS1C		

9.0 COMMAND DESCRIPTIONS & EXAMPLES

The SP1 Command

This command is used to set or read the temperature setpoint of Channel 1. This is a steady state temperature that the chamber is to control at.

Range: R1H to R1L

Ex: To set Chan. 1 setpoint to 100.0 deg. C (Decimal point is implied - not a part of character string)

= SP1 1000

Ex: To read the current setpoint of Channel 1: **? SP1**

The SP2 Command

This command is used to set or read the relative humidity setpoint of Channel 2. This is a steady state humidity that the chamber is to control at.

Range: 0.0 to 100.0 % RH

Ex: To set Channel 2 setpoint to 94.0 % RH (Decimal point is implied - not a part of character string)

= SP2 940

Ex: To read the current setpoint of Channel 2: **? SP2**

The EV1 thru EV6 Command

This command is used to enable, disable, or get the current status of the six optional events.

Range: 0 to 1 Definition: 0 = Turn Event Off
1 = Turn Event On

Ex: To turn Event 1 On: **= EV1 1**

Ex: To check status of Event 2: **? EV2**

The LEV1 and LEV2 Commands

This command is used to set or read the current setting of Logic Event 1, or Logic Event 2 (used for special applications).

Range: 0 to 1 Definition: 0 = Set to zero 1 = Set to one

Ex: To set Logic Event 1 (LEV1) to one: **= LEV1 1**

Ex: To read the current setting of Logic Event 2 (LEV2): **? LEV2**

The ON and OFF Commands

These commands are used to turn the VersaTenn ON or OFF in the same sense that the ON / OFF key on the front panel is used to turn ON / OFF the VersaTenn's outputs.

Ex: To turn the VersaTenn's outputs ON: **= ON**

The STP Command

This command when used with the '=' command is used to enter a program step in a file with a total program space < 100 steps. The command can only be executed while in the HOLD mode. This command has the following message syntax:

= <space> **STP** <space> <**File #**> <space> <**Step #**> <space> <**Step type**> <space> <**ARG.4**> <space> <**ARG.5**> <space> <**ARG.6**> <space> . . . **ARG.N**

Where:

< >	=	Description of a character in non-literal form.
File #	=	The file number that this step will be programmed into.
Step #	=	The step number that is to be programmed.
Step type	=	The step-type to be programmed at the step.
ARG.4 . . . ARG.N	=	The data assigned to the appropriate parameters of this type of step.

File #: This argument specifies the file to which the arguments that follow are to be programmed.

Range: 1 to 10

Step #: This argument specifies the step to which the arguments that follow are to be programmed.

Range: 1 to 99

Step type: This argument specifies the type of step that is to be programmed at the given step. It consists of a one-character ASCII number, which can be any of the following:

0 = Setpoint	3 = Autostart
1 = Jumploop	4 = Stop
2 = Waitfor	5 = Link

ARG.4 thru ARG.N:

These arguments constitute the data that a given step will require based upon the step-type. Each of the step-types will require the following data: (Note that the parameter names are not meant to imply that this information is to be included. Each data item is to be a decimal number only.)

0 - Setpoint

ARG.4 = SP1 – Channel 1 Setpoint	ARG.11 = EV3 – Event 3
ARG.5 = SP2 – Channel 2 Setpoint	ARG.12 = EV4 – Event 4
ARG.6 = Hour – Ramp Time Hours	ARG.13 = EV5 – Event 5
ARG.7 = Min – Ramp Time Minutes	ARG.14 = EV6 – Event 6
ARG.8 = Sec – Ramp Time Seconds	ARG.15 = LEV1
ARG.9 = EV1 – Event 1	ARG.16 = LEV2
ARG.10 = EV2 – Event 2	

Note: If the VersaTenn you are using is not equipped with any optional events, or if you do not use LEV1 or LEV2, these parameters must still contain some data (ex: 0 to turn Off, 1 to turn On).

Example of a Setpoint step:

= STP 1 1 0 1000 -1 0 30 0 0 0 0 0 0 0 0 0

The above example programs File # 1, Step # 1 as a Setpoint step. Channel 1 setpoint is 100.0 degrees, Channel 2 is turned Off (-1 setpoint), time is no hours, 30 minutes, no seconds. The six events are turned Off, both LEV1 and LEV2 are also turned Off.

1 - Jumploop

ARG.4 = JS – Jump Step
ARG.5 = JC – Jump Count

Example of a Jumploop step:

= STP 1 4 1 1 255

The above example programs File # 1, Step # 4 as a jumploop step. The step to jump to is 1. Repeat the steps in between another 255 times.

2 - Waitfor

ARG.4 =	W1:	Wait for Channel 1 actual
ARG.5 =	W2:	Wait for Channel 2 actual
ARG.6 =	WHR:	Wait for hour (expired hour)
ARG.7 =	WMN:	Wait for minutes (expired minute)
ARG.8 =	WE:	Wait for external event (0 = open, 1 = closed)

Example of a Waitfor step:

= STP 1 1 2 1000 * * * 1

The above example programs File # 1, Step # 1 as a Waitfor step. The program is to wait for 100.0 degrees actual on Channel 1, the Waitfor arguments of Channel 2, and the hours and minutes are not set to any condition (this is done by using the ‘ * ‘ character as a filler). The program also waits for the external event to be closed. Before a program gets out of a Waitfor condition, **all** of the conditions that are set in the Waitfor **must** be satisfied.

3 - Autostart

ARG.4 = DAY – Autostart Day (0 to 13)
ARG.5 = HOUR – Autostart Hour (0 to 23)
ARG.6 = MIN – Autostart Minute (0 to 60)

Example of an Autostart step:

= STP 1 1 3 3 8 0

The above example programs File # 1, Step # 1 as an Autostart step. The step tells the VersaTenn to start at day 3 from the current day, at 8 a.m. When using the Autostart step, make sure the real time clock of the VersaTenn is set properly.

4 - Stop

ARG.4 = 0 – VersaTenn outputs are Off
1 – VersaTenn outputs are On

Example of a Stop step:

= STP 1 5 4 0

The above example programs File # 1, Step # 5 as a Stop step. When the program gets to this step it will shut the VersaTenn outputs Off.

5 - Link

ARG.4 = LF - File number to link to

Example of a Link step:

= 1 5 5 2

The above example programs File # 1, Step # 5 as a Link step. When the profile gets to this step it then links to File #2 and begins to execute this file.

The STRT Command

This command used to start a program in the memory of the VersaTenn software. If the program is already in a Run mode, an error is flagged. The syntax of this command is:

= <space> **STRT** <space> **ARG.1** <space> **ARG.2**

Where:

< >	=	Indicates description of a character in non-literal form.
ARG.1	=	File number to run (range 1 to 10)
ARG.2	=	Step number to start at

Ex: To start File # 1 at Step # 2: **= STRT 1 2**

The run status may be checked with the **? RUN** command.

The RSUM Command

This command is used to resume a profile from the point that the program was put into the 'HOLD' mode. The profile will continue from where it was left for the time remaining.

Ex: To resume a profile currently in Hold: **= RSUM**

The HOLD Command

This command is used to put the currently running profile into the Hold mode. Before sending this command, make sure the profile is in the Run mode using the "? RUN" command, otherwise an error will be generated.

Ex: To stop execution of the currently running profile: **= HOLD**

The CLRF Command

This command is used to clear the indicated file from the VersaTenn memory. The command syntax is:

= CLRF <space> <**File #**>

Where:

< >	=	Indicates description of a character in non-literal form.
File #	=	Specifies the file number to clear (Range: 1 – 100)

Ex: To clear File # 1: **= CLRF 1**

The CF Command

This command is used to set or read the current temperature setting (be sure to read the user manual regarding the use of this instruction and its' side-effects).

Range: 0 to 1 Definition: 0 = degrees Celsius
1 = degrees Fahrenheit

Ex: To set the VersaTenn to operate in Celsius: **= CF 0**

Ex: To read the current temperature setting: **? CF**

The GS Command

This command is used to set or read the Guaranteed Soak for temperature on Channel 1 (refer to user manual on description of Guaranteed Soak).

Range: 0 to 5.0 degrees C
0 to 9.0 degrees F

Note: A value of 0 disables Guaranteed Soak

Ex: To set Guaranteed Soak to 2.0 degrees C.

= GS 20 (Note: Decimal point is implied - not a part of character string)

Ex: To read the current setting of Guaranteed Soak: **? GS**

The TI Command

This command is used to set or read the real time clock hours and minutes. The syntax of the command is as follows:

= TI <space> <**hours**> <space> <**minutes**> -- OR -- **? TI**

Where:

< >	=	Non-literal description	
hours	=	Real time hours;	Range = 0 to 23
minutes	=	Real time minutes;	Range = 0 to 59

Ex: To set the real time clock for 8:30 am

= TI 8 30

Ex: To read the current setting of the real time clock: **? TI**

The command will return the following data: <**hours**> <space> <**minutes**>

The RTD Command

This command is used to set or read the current temperature resistance curve.

Range:	0 to 1	Definition:	0 = JIS
			1 = DIN

Ex: To set the temperature resistance curve to JIS: **= RTD 0**

Ex: To read the current temperature resistance curve: **? RTD**

The 4 -20 Command

This command is used to set or read the range of the optional 4 to 20 milliamp output.

Range:	0 = Heat / Cooling
	1 = Heat
	2 = Cooling

Ex: To set the 4 – 20 milliamp output to represent; 4 to 20 = 0 to 100% heat output

= 4-20 1

Ex: To read the current setting of the 4 - 20 milliamp output: **? 4-20**

The LOCK Command

This command is used to set or read the current operator keyboard LOCK access.

Range: 0 to 2

Definition:	0 = No LOCK
	1 = Data change / entry inhibited on all parameters except SYSTEM Group Parameters and the LOCK parameter, and Step # in PROG editor.
	2 = Data change / entry inhibited on all parameters except ER1, ER2, ALRM, the LOCK parameter and step # in PROG editor.

Ex: To set full access of the keyboard: **= LOCK 0**

Ex: To read the current operator LOCK access: **? LOCK**

The R1H Command

This command is used to set or read Channel 1 temperature Hi setpoint Range Limit. The temperature setpoint cannot be set higher than the value that the R1H parameter is set to.

Range:	-99.9 to 200.0 degrees C	Range – Displayed in Units:	-xxx to xxxx degrees C
	-99.9 to 392.0 degrees F		-xxx to xxx degrees F

Ex: To set the temperature Hi setpoint Range Limit of Channel 1 to 170.0 degrees C

= R1H 1700 (Note: Decimal point is implied - not a part of character string)

Ex: To read the current temperature Hi setpoint Range Limit of Channel 1: **? R1H**

The R1L Command

This command is used to set or read Channel 1 temperature Low setpoint Range Limit. The temperature setpoint cannot be set lower than the value that the R1L parameter is set to.

Range:	-99.9 to 200.0 degrees C	Range – Displayed in Units:	-xxx to xxxx degrees C
	-99.9 to 392.0 degrees F		-xxx to xxx degrees F

Ex: To set the temperature Low setpoint Range Limit of Channel 1 to -77.0 degrees C

= R1L -770 (Note: Decimal point is implied - not a part of character string)

Ex: To read the current temperature Low setpoint Range Limit of Channel 1: **R1L**

The R2H Command

This command is used to set or read Channel 2 temperature Hi setpoint Range Limit. The temperature setpoint cannot be set higher than the value that the R2H parameter is set to.

Range:	-99.9 to 200.0 degrees C	Range – Displayed in Units:	-xxx to xxxx degrees C
	-99.9 to 392.0 degrees F		-xxx to xxx degrees F

Ex: Refer to R1H example

The R2L Command

This command is used to set or read Channel 2 temperature Low setpoint Range Limit. The temperature setpoint cannot be set lower than the value that the R2L parameter is set to.

Range:	-99.9 to 200.0 degrees C	Range – Displayed in Units:	-xxx to xxxx degrees C
	-99.9 to 392.0 degrees F		-xxx to xxx degrees F

Ex: Refer to the R1L example

The A1H Command

This command is used to set or read the upper Alarm Limit of Channel 1. If the actual temperature exceeds this limit, the VersaTenn will generate an alarm message.

Range: Limited by the setting of the Range Hi (R1H) parameter

Ex: To set the Alarm Hi Limit to 200.0 degrees C

= A1H 2000 (Note: Decimal point is implied - not a part of character string)

Ex: To read the current Alarm Hi Limit: **? A1H**

The A1L Command

This command is used to set or read the lower Alarm Limit of Channel 1. If the actual temperature exceeds this limit, the VersaTenn will generate an alarm message.

Range: Limited by the setting of the Range Low (R1L) parameter

Ex: To set the Alarm Low Limit to -98.0 degrees C

= A1L -980 (Note: Decimal point is implied - not a part of character string)

Ex: To read the current Alarm Low Limit: **? A1L**

The A2H Command

This command is used to set or read the upper Alarm Limit of Channel 2. If the actual humidity exceeds this limit, the VersaTenn will generate an alarm message.

Range: Limited by the setting of the Range Hi (R2H) parameter

Note: The alarm output for humidity is not normally connected.

Ex: Refer to the A1H example

The A2L Command

This command is used to set or read the lower Alarm Limit of Channel 2. If the actual humidity exceeds this limit, the VersaTenn will generate an alarm message.

Range: Limited by the setting of the Range Low (R2L) parameter

Note: The alarm output for humidity is not normally connected.

Ex: Refer to the A1L example

The CAL1 Command

This command is used to set or read Channel 1 temperature Calibration Offset. The value of CAL1 is added to the actual measured temperature and then displayed as the current actual temperature. When a second display is used to monitor actual chamber temperature, the user can offset any errors between the two displays.

Range:	-5.0 to 5.0 degrees C	Range – Displayed in Units:	-50 to 50 degrees C
	-9.0 to 9.0 degrees F		-90 to 90 degrees F

Ex: To set the Calibration Offset to .2 degrees C

= CAL1 2 (Note: Decimal point is implied - not a part of character string)

Ex: To get the current Calibration Offset: **? CAL1**

The CAL2 Command

This command is used to set or read a humidity or altitude / pressure calibration offset. The value of CAL2 is added to the actual measured signal input and then displayed as the current humidity or altitude / pressure actual. When a second display is used to monitor actual humidity or altitude / pressure, the user can offset any errors between the two displays.

Range:	-9.0 to 9.0 % RH	Range – Displayed in Units:	-90 to 90 % RH
	-9.0 to 9.0 units	(**Altitude / Pressure**)	-90 to 90 units

Ex: Refer to the example of CAL1

The L - Value Commands

These commands are used to set or read the VersaTenn 'L' parameters. These parameters are used by the VersaTenn's control algorithm. Their definitions are given in the main controller manual.

Range:

L3	0 to 100%
L4	0 to 100%
L6	-99.9 to 212.0 F / -99.9 to 100.0 C
L7	0 to 100%
L8	0 to 100%
L9	-99.9 to 212.0 F / -99.9 to 100.0 C
L11	0 to 100%
L12	0 to 100%
L14	0.0 to 60.0 minutes
L15	0.0 to 2.0 minutes

Ex: To read the L3 parameter setting: **? L3**

The OT11 Command

This command is used to set or read the current definition of Output 11. (Used for special applications)

Range: 0 to 1	Definition:	0 = ON / OFF
		1 = Time Proportional

Ex: To select Output 11 as a time proportional output: **= OT11 1**

Ex: To read the current definition of Output 11: **? OT11**

The OT18 Command

This command is used to set or read the current definition of Output 18. (Used for special applications)

Range: 0 to 1	Definition:	0 = Vent
		1 = Boost Cooling

Ex: To select Output 18 as a Vent output: **= OT18 0**

Ex: To read the current definition of Output 18: **? OT18**

The OT0 Command

This command is used to read the data on Output Bank 0 of the controller. The information returned is a decimal encoded binary number describing the status of the represented outputs.

Range: 0 to 255

Ex: To read the status of Output Bank 0: **? OT0**

The returned value will represent the following outputs, where: 1 = ON 0 = OFF

Bit 0 = Output 3	Bit 4 = Output 14
Bit 1 = Output 7	Bit 5 = Output 15
Bit 2 = Output 9	Bit 6 = Output 17
Bit 3 = Output 11	Bit 7 = Output 18

The OT1 Command

This command is used to read the data on Output Bank 1 of the controller. The information returned is a decimal encoded binary number describing the status of the represented outputs.

Range: 0 to 255

Ex: To read the status of Output Bank 1: **? OT1**

The returned value will represent the following outputs, where: 1 = ON 0 = OFF

Bit 0 = Output 1	Bit 4 = Output 6
Bit 1 = Output 2	Bit 5 = Output 8
Bit 2 = Output 4	Bit 6 = Alarm #1
Bit 3 = Output 5	Bit 7 = Alarm #2

The OT2 Command

This command is used to read the data on Output Bank 2 of the controller. The information returned is a decimal encoded binary number describing the status of the represented outputs.

Range: 0 to 255

Ex: To read the status of Output Bank 2: **? OT2**

The returned value will represent the following outputs, where: 1 = ON 0 = OFF

Bit 0 = Output 10	Bit 4 = Not used
Bit 1 = Output 11	Bit 5 = Not used
Bit 2 = Output 13	Bit 6 = Not used
Bit 3 = Output 16	Bit 7 = Not used

The OT3 Command

This command is used to read the data on Output Bank 3 of the controller. The information returned is a decimal encoded binary number describing the status of the represented outputs.

Range: 0 to 255

Ex: To read the status of Output Bank 3: **? OT3**

The returned value will represent the following outputs, where: 1 = ON 0 = OFF

Bit 0 = Event 1	Bit 4 = Event 5
Bit 1 = Event 2	Bit 5 = Event 6
Bit 2 = Event 3	Bit 6 = Not used
Bit 3 = Event 4	Bit 7 = Not used

The AT1H Command

This command is used to set or read the Auto-tune for Channel 1.

Range: 0 to 3	Definition:	0 = OFF	2 = Medium
		1 = Slow	3 = Fast

Ex: To turn Auto-tune OFF: **= AT1H 0**

Ex: To read the current Auto-tune setting: **? AT1H**

The PB1C Command

This command is used to set or read the Cooling Output Proportional Band of Channel 1.

Range:	0.0 to 50.0 degrees C	Range – Displayed in Units:	0 to 500 degrees C
	0.0 to 90.0 degrees F		0 to 900 degrees F

Ex: Refer to the PB1H example

The PB1H Command

This command is used to set or read the Heat Output Proportional Band of Channel 1.

Range:	0.0 to 50.0 degrees C	Range – Displayed in Units:	0 to 500 degrees C
	0.0 to 90.0 degrees F		0 to 900 degrees F

Ex: To set the Proportional Band to 3.0 degrees C

= PB1H 30 (Note: Decimal point is implied - not a part of character string)

Ex: To read the current Heat Proportional Band: **? PB1H**

The PB2C Command

This command is used to set or read the Dehumid Output Proportional Band of Channel 2.

Range: 0.0 to 99.9% RH Ex: Refer to the PB1H example.

The PB2H Command

This command is used to set or read the Humidify Output Proportional Band of Channel 2.

Range: 0.0 to 99.9% RH Ex: Refer to the PB1H example.

The RS1C Command

This command is used to set or read the Cooling Output Reset Value of Channel 1.

Range: 0.00 to 9.99

Ex: To set Channel 1 Cooling Output Reset to 0.20

= RS1C 20 (Note: Decimal places are implied - not a part of character string)

Ex: To read the current Reset Value of Channel 1 Cooling: **? RS1C**

The RS1H Command

This command is used to set or read the Heat Output Reset Value of Channel 1.

Range: 0.00 to 9.99

Ex: To set Channel 1 Heat Output Reset to 0.50

= RS1H 50 (Note: Decimal places are implied - not a part of character string)

Ex: To read the current Reset Value of Channel 1 Heat: **? RS1H**

The RS2C Command

This command is used to set or read the Dehumid Output Reset Value of Channel 2.

Range: 0.00 to 9.99 Ex: Refer to the RS1C example.

The RS2H Command

This command is used to set or read the Humidify Output Reset Value of Channel 2.

Range: 0.00 to 9.99 Ex: Refer to the RS1H example.

The RT1C Command

This command is used to set or read the Cooling Output Rate Value of Channel 1.

Range: 0.00 to 9.99

Ex: To set Channel 1 Cooling Output Rate to 0.01

= RT1C 1 (Note: Decimal places are implied - not a part of character string)

To read the current Rate Value of Channel 1 Cooling: **? RS1C**

The RT1H Command

This command is used to set or read the Heat Output Rate Value of Channel 1.

Range: 0.00 to 9.99

Ex: To set Channel 1 Heat Output Rate to 0.01

= RT1H 1 (Note: Decimal places are implied - not a part of character string)

Ex: To read the current Rate Value of Channel 1 Heat: **? RS1H**

The RT2C Command

This command is used to set or read the Dehumid Output Rate Value of Channel 2.

Range: 0.00 to 9.99 Ex: Refer to the RT1C example

The RT2H Command

This command is used to set or read the Humidify Output Rate Value of Channel 2.

Range: 0.00 to 9.99 Ex: Refer to the RT1H example

The RB1C Command

This command is used to set or read the Cooling Output Rate Band of Channel 1.

Range: 0 to 7

Ex: To set the Rate Band of Channel 1 Cooling Output to 3: **= RB1C 3**

Ex: To read the current Cooling Rate Band of Channel 1: **? RB1C**

The RB1H Command

This command is used to set or read the Heat Output Rate Band of Channel 1.

Range: 0 to 7

Ex: To set the Rate Band of Channel 1 Heat Output to 1: **= RB1H 1**

Ex: To read the current Heat Rate Band of Channel 1: **? RB1H**

The RB2C Command

This command is used to set or read the Dehumid Output Rate Band of Channel 2.

Range: 0 to 7 Ex: Refer to the RB1C example

The RB2H Command

This command is used to set or read the Humidify Output Rate Band of Channel 2.

Range: 0 to 7 Ex: Refer to the RB1H example

The CT1C Command

This command is used to set or read the Cooling Output Cycle Time of Channel 1.

Range: 7 to 60

Ex: To set the Cycle Time of Channel 1 Cooling Output to 7: **= CT1C 7**

Ex: To read the current Cooling Cycle Time of Channel 1: **? CT1C**

The CT1H Command

This command is used to set or read the Channel 1 Heat Output Cycle Time.

Range: 1 to 60

Ex: To set the Cycle Time of Channel 1 Heat Output to 5 seconds: **= CT1H 5**

Ex: To read the current Heat Cycle Time of Channel 1: **? CT1H**

The CT2C Command

This command is used to set or read the Channel 2 Dehumid Output Cycle Time.

Range: 7 to 60 Ex: Refer to CT1C example

The CT2H Command

This command is used to set or read the Channel 2 Humidify Output Cycle Time.

Range: 7 to 60 Ex: Refer to CT1H example

The DB1 Command

This command is used to set or read the Channel 1 Dead Band.

Range:	-25.0 to 25.0 degrees C	Range – Displayed in Units:	-250 to 250 degrees C
	-45.0 to 45.0 degrees F		-450 to 450 degrees F

Ex: To set the Dead Band of Channel 1 to 5.0 degrees: **= DB1 50**

Ex: To read the current Dead Band of Channel 1: **? DB1**

The DB2 Command

This command is used to set or read the Channel 2 Dead Band.

Range: -25.0 to 25.0 % RH Ex: Refer to the DB1 example

The ALT Command

This command is used to set or read the current Altitude Correction Factor.

Range: 0 to 2

Definition:	
0	= 0 ft
1	= 2500 ft
2	= 5000 ft

Ex: To set the Altitude Correction Factor to 0 feet: **= ALT 0**

Ex: To read the current Altitude Correction Factor: **? ALT**

The VCMP Command

This command is used to select or read the current setting of the Vaisala humidity sensor - Temperature Compensation Curve. This is used only for a specific humidity sensor. Refer to the VersaTenn's Setup Parameter Sheet for the factory setting. This sheet is in the Test Report of the chamber manual.

Range: 0 to 1

Definition:	
0	= ON
1	= OFF

Ex: To select the Temperature Compensation Curve - ON

= VCMP 0

To read the current setting of the Temperature Compensation Curve: **? VCMP**

The CMS Command

This command is used to invoke a Communications Shutdown, which is a system shutdown controlled via serial communications.

Range: 0 to 1	Definition:	0 = No shutdown requested. (Note, however, there are other sources of a request for shutdown.
		1 = Shutdown requested

Ex: To request a shutdown: **= CMS 1**

The SYRS Command

This command is used to cause a complete System Reset. The type of reset that will occur will depend on the configuration of the Cold / Warm / Pro start in the system at the time the command is executed.

The syntax of this command is:

= SYRS 1

When this command is used, there are system operation items to consider.

1) Under the Xon-Xoff protocol, the last character transmitted by the VersaTenn will be an Xoff character, and will have to be dealt with by the host.
2) Under the ANSI 2.2 – A3 protocol, the VersaTenn will default to a disconnected state.

The C1 Command

This command is used to read the Channel 1 actual temperature.

Ex: To read the current chamber actual temperature of Channel 1: **? C1**

The C2 Command

This command is used to read the Channel 2 actual humidity.

Ex: To read the current chamber actual humidity of Channel 2: **? C2**

The RUN Command

This command is used to check the VersaTenn current mode of operation.

Ex: **? RUN**

Returned Value: 0 = Hold mode 1 = Run mode

The MTR Command

This command is used to monitor the current File or Step that is being executed while control is in the Run mode, or the File or Step that will execute from the Hold mode in the event that a RESUME is commanded.

The syntax of the command is as follows:

? <space> **MTR**

There are no arguments to the command. The command will return the following data:

<**File #**> <space> <**Step #**> <space> <**Step type**> <space> <**ARG.4**> <space> <**ARG.5**> <space>……<**ARG.n**>

Where:

< >	=	Description of a character in non-literal form.
File #	=	The file number that is currently executing.
Step #	=	The step number that is currently executing.
Step type	=	The step-type of the current step.
ARG.4 . . . ARG.N	=	The appropriate data based upon the given step-type.

Step type: The Step-type of the Step number returned will be a one-character ASCII number representing one of the following Step-types.

0 = Setpoint	3 = Autostart
1 = Jumploop	4 = Stop
2 = Waitfor	5 = Link

ARG.4 thru ARG.N:

These arguments constitute the data that a given step includes, based upon the step-type. This data is parallel to the data that can be found via the front panel when the unit is in the Run mode. Each of the step-types will include the following data: (Note that the parameter names are not meant to imply that this information will be returned. Each data item will return a decimal number only.)

0 - Setpoint

ARG.4 = SP1 – Channel 1 Ramping Setpoint
ARG.5 = SP2 – Channel 2 Ramping Setpoint
ARG.6 = Hour – Ramp Time Hours Remaining
ARG.7 = Min – Ramp Time Minutes Remaining
ARG.8 = Sec – Ramp Time Seconds Remaining
ARG.9 = EV1 – Event 1 Status
ARG.10 = EV2 – Event 2 Status
ARG.11 = EV3 – Event 3 Status
ARG.12 = EV4 – Event 4 Status
ARG.13 = EV5 – Event 5 Status
ARG.14 = EV6 – Event 6 Status
ARG.15 = LEV1 Status
ARG.16 = LEV2 Status

Note: If the VersaTenn you are using is not equipped with any optional events, or if you do not use LEV1 or LEV2, these parameters will still be a part of the returned data (ex: 0 to turn Off, 1 to turn On).

See an example of a Setpoint on the next page.

Example of a Setpoint step:

1 1 0 1000 -1 0 30 0 0 0 0 0 0 0 0 0

The above example returns the current data: File # 1, Step # 1 - is a Setpoint step. Channel 1 is ramping to a setpoint of 100.0 degrees, Channel 2 is turned Off (-1 setpoint), time remaining is no hours, 30 minutes, no seconds. The six events are turned Off, and both LEV1 and LEV2 are also turned Off.

1 - Jumploop

ARG.4 = JS – Jump Step
ARG.5 = JC – Jump Count

Example of a Jumploop step:

1 4 1 1 255

The above example returns the current data: File # 1, Step # 4 - is a Jumploop step. The step to jump to is 1, and there remains 255 repeats.

2 - Waitfor

ARG.4	=	W1:	Wait for Channel 1 actual
ARG.5	=	W2:	Wait for Channel 2 actual
ARG.6	=	WHR:	Wait for hours programmed
ARG.7	=	WMN:	Wait for minutes programmed
ARG.8	=	WE:	Wait for external event (0 = open, 1 = closed)
ARG.9	=	WRH:	Wait for remaining hours
ARG.10	=	WRM:	Wait for remaining minutes
ARG.11	=	WES:	Wait for event status

Example of a Waitfor step:

1 1 2 1000 * * * 1 0 0 0

The above example returns the current data: File # 1, Step # 1 - is a Waitfor step. The program is to remain in Hold until the actual temperature for Channel 1 reaches 100.0 degrees, and the external Event is closed. The Waitfor Arguments of Channel 2, and the hours and minutes are not set to any condition because the returned value for each of these parameters is a ‘ * ‘ character, which is used as a filler. The hours and minutes remaining is 0 and the status of the external event is open.

3 - Autostart

ARG.4	=	DAY:	Autostart Day (0 to 13)
ARG.5	=	HOUR:	Autostart Hour (0 to 23)
ARG.6	=	MIN:	Autostart Minute (0 to 60)
ARG.7	=	ADA:	Autostart Days Accumulated
ARG.8	=	RH:	Real Time Hours
ARG.9	=	RM:	Real Time Minutes

Example is on next page.

Example of an Autostart step:

1 1 3 3 8 0 1 18 0

The above example returns the current data: File # 1, Step # 1 - is an Autostart step. The step tells the VersaTenn to start at day 3 from the current day, at 8 a.m. So far, 1 day has gone by, and the current time is 6 p.m.

<u>4 - Stop</u>

ARG.4 = 0 – VersaTenn outputs are Off
1 – VersaTenn outputs are On

Example of a Stop step:

1 5 4 0

The above example returns the current data: File # 1, Step # 5 - is a Stop step. The outputs are Off.

<u>5 - Link</u>

ARG.4 = LF - File number to link to

Example of a Link step:

= 1 5 5 2

The above example returns the current data: File # 1, Step # 5 - is a Link step. When the profile gets to this step it then links to File #2 and begins to execute this file.

The AFL Command

This command asks the VersaTenn for the file numbers of files that are programmed.

Ex: To read the programmed file numbers: **? AFL**

The command will return the following data:

1 <space> **ARG.2** <space> **ARG.3** <space>……**ARG.N**

<u>Where:</u>

< >	=	Description of a character in non-literal form
ARG.2 . . . ARG.N	=	File numbers of programmed files

Let’s say that we entered a program into the memory of the VersaTenn as File #1, and another program as File #4. When the AFL command is sent to the VersaTenn, the following data will be returned.

1 <space> **1** <space> **4**

Indicates File #1 and File #4 are programmed.

The FST Command

This command asks the VersaTenn for the number of steps programmed in the specified file.

The syntax of the command is:

? FST <space> <**File #**>

Where:

< >	=	Description of a character in non-literal form
File #	=	Specifies the file number to check; Range: 1 to 10

The command returns the # of steps programmed in the file.

Where: # of steps is a 1 or 2 character ASCII decimal number.

The MDL Command

This command is used to read the VersaTenn software version.

Ex: To read the software version: **? MDL**

The command will return the current software version.

The STP Command

This command when used with the ‘ **?** ‘ command is used to read a program step from the memory of the VersaTenn controller. The total steps will not be greater than 99. The syntax of the command follows.

? <space> <**STP**> <space> <**File #**> <space> <**Step #**>

Where:

< >	=	Description of a character in non-literal form.
File #	=	The file number that is to be queried. Range = 1 to 10
Step #	=	The step number that is to be queried. Range = 1 to 99

The command will return the following data:

<**Step type**> <space> <**ARG.2**> <space> <**ARG.3**>…….<**ARG.N**>

Where:

< >	=	Description of a character in non-literal form.
Step type	=	The step-type of the queried step.
ARG.2 . . . ARG.N	=	The data assigned to the appropriate parameters of this type of step.

Step type: This argument specifies the type of step that is programmed at the given step. It consists of a one-character ASCII number, which can be any of the following:

0 = Setpoint	3 = Autostart
1 = Jumploop	4 = Stop
2 = Waitfor	5 = Link

ARG.2 thru ARG.N:

These arguments constitute the data that a given step has been programmed with, based upon the step-type. (Note that the parameter names are not meant to imply that this information will be returned. Each data item will return a decimal number only.)

Ex: The following example will read the information of File # 1, Step # 1 (**or for any applicable step #**). The data returned depends on the step-type that is programmed.

? STP 1 1 (If the step-type is a setpoint step, the data returned will be in the following format.)

0 - Setpoint

ARG.2 = SP1 – Channel 1 Setpoint	ARG.9 = EV3 – Event 3
ARG.3 = SP2 – Channel 2 Setpoint	ARG.10 = EV4 – Event 4
ARG.4 = Hour – Ramp Time Hours	ARG.11 = EV5 – Event 5
ARG.5 = Min – Ramp Time Minutes	ARG.12 = EV6 – Event 6
ARG.6 = Sec – Ramp Time Seconds	ARG.13 = LEV1 Status
ARG.7 = EV1 – Event 1	ARG.14 = LEV2 Status
ARG.8 = EV2 – Event 2	

Note: If the VersaTenn you are using is not equipped with any optional events, or if you do not use LEV1 or LEV2, these parameters will still be a part of the returned data (ex: 0 to turn Off, 1 to turn On).

Example of a Setpoint step:

0 1000 -1 0 30 0 0 0 0 0 0 0 0 0

The above example returns: (File # 1, Step # 1) as a Setpoint step. Channel 1 setpoint is 100.0 degrees, Channel 2 is turned Off (-1 setpoint), time is no hours, 30 minutes, no seconds. The six events are turned Off, and both LEV1 and LEV2 are also turned Off.

1 - Jumploop

ARG.2 = JS – Jump Step
ARG.3 = JC – Jump Count

Example of a Jumploop step:

1 1 255

The above example returns: (File # 1, Step # 4) as a Jumploop step. The step to jump to is 1, and the number of repeats is 255.

2 - Waitfor

ARG.2	=	W1:	Wait for Channel 1 actual
ARG.3	=	W2:	Wait for Channel 2 actual
ARG.4	=	WHR:	Wait for real-time clock hour
ARG.5	=	WMN:	Wait for real-time clock minutes
ARG.6	=	WE:	Wait for external event (0 = open, 1 = closed)

Example of a Waitfor step:

2 1000 * * * 1

The above example returns: (File # 1, Step # 1) as a Waitfor step. The program is to wait for 100.0 degrees actual on Channel 1. The Waitfor arguments of Channel 2, and the hours and minutes are not set to any condition because the returned value for each of these parameters is a ‘ * ‘ character, which is used as a filler. The program also waits for the external event to be closed.

3 - Autostart

ARG.2	=	DAY:	Autostart Day (0 to 13)
ARG.3	=	HOUR:	Autostart Hour (0 to 23)
ARG.4	=	MIN:	Autostart Minute (0 to 60)

Example of an Autostart step:

3 3 8 0

The above example returns: (File # 1, Step # 1) as an Autostart step. The step tells the VersaTenn to start at day 3 from the current day, at 8 a.m.

4 - Stop

ARG.2 = 0 – VersaTenn outputs are Off
1 – VersaTenn outputs are On

Example of a Stop step:

4 0

The above example returns the current data: (File # 1, Step # 5) as a Stop step. When the program gets to this step it will shut the VersaTenn outputs Off.

5 - Link

ARG.2 = LF - File number to link to

Example of a Link step:

5 2

The above example returns the current data: (File # 1, Step # 5) - is a Link step. When the profile gets to this step it then links to File #2 and begins to execute this file.

The RJ Command

This command is used to return the number of jumps remaining and the last jump loop step executed so that nested jumps loops remaining can be determined.

Ex: To read the remaining jumps: **? RJ**

The data returned will be:

<**step**> <space> <**jump remaining**>

Where: <**step**> will be a step number of the last executed jump loop. If this value is 0, no jump loops have been executed in this file.

<**jumps remaining**> will be a number from 0 to 255 that will tell us the number of loops remaining for the last executed jump loop.

The EI Command

This command is used to read the current status of the logic event input.

The returned value will be:

0 or **1** Definition: 0 = open, 1 = closed

The DIP Command

This command is used to read the status of the Dip Switch. The data returned represents the position of the switches.

Ex: **? DIP**

7F	=	All Dip switches On
5F	=	Dip switch 1 Off, rest On
3F	=	Dip switch 1, 2 Off, rest On
1F	=	Dip switch 1 - 3 Off, 4 - 8 On
F	=	Dip switch 1 - 4 Off, 5 - 8 On
7	=	Dip switch 1 - 5 Off, 6 - 8 On
3	=	Dip switch 1 - 6 Off, 7 - 8 On
1	=	Dip switch 1 - 7 Off, 8 On
0	=	Dip switch 1 - 8 Off

The ALM Command

This command is used to read the alarm status of the VersaTenn. A code will be returned describing the alarm condition. The alarm message is cleared on a read.

0	=	No alarm occurred
1	=	A1H occurring (Alarm Chan. 1 hi)
2	=	A1L occurring (Alarm Chan. 1 lo)
4	=	A2H occurring (Alarm Chan. 2 hi)
8	=	A2L occurring (Alarm Chan. 2 lo)
16	=	A3H occurring (Alarm Chan. 2 pressure / altitude hi)
32	=	A3L occurring (Alarm Chan. 2 pressure / altitude lo)

The ER1 Command

This command is used to read the ER1 fatal error messages. The returned value is an error code.

Ex: **? ER1**

The ER2 Command

This command is used to read the ER2 non-fatal error messages. The returned value is an error code.

Ex: **? ER2**

The 1LO Command

This command is used to read the percent of cooling output of Channel 1. (The outputs must be On.)

Ex: To read the current % of cooling output by the controller.

? 1LO

The 1HI Command

This command is used to read the percent of heat output of Channel 1. (The outputs must be On.)

Ex: To read the current % of heat output by the controller.

? 1HI

The 2LO Command

This command is used to read the percent dehumid output of Channel 2. (The outputs must be On.)

Ex: Refer to the 1HI example

The 2HI Command

This command is used to read the percent humidify output of Channel 2. (The outputs must be On).

Ex: Refer to the 1HI example

The INP Command

This command is used to read the status of the Logic Inputs. The information returned is a decimal encoded binary number describing the status of the represented inputs.

Range: 0 to 255

Ex: **? INP**

Bit 0 = Input 1	Bit 4 = Input 5
Bit 1 = Input 2	Bit 5 = Input 6
Bit 2 = Input 3	Bit 6 = Input 7
Bit 3 = Input 4	Bit 7 = Input 8

Where: 1 = On, 0 = Off

The INP1 Command

This command is used to read the status of the Logic inputs. The information returned is a decimal encoded binary number describing the status of the represented inputs. “Board-type” represents VersaTenn hardware options.

Range: 0 to 255

Ex: **? INP1**

The returned value will represent the status of the following inputs.

Bit 0 = Input 1	1 = On,	0 = Off
Bit 1 = Event Input	1 = Open,	0 = Closed
Bit 2 = Remote Hold	1 = Open,	0 = Closed
Bit 3 = Keylock	1 = Open,	0 = Closed
Bit 4 = Comms Detect	1 = No Comms,	0 = Comms
Bit 5 = Board Type	1 = -1xxx,	0 = -0xxx
Bit 6 = Not Used		
Bit 7 = Not Used		

10.0 ER1 ERROR CODES

Code No.	Error Type
1	Processor RAM error
2	EPROM checksum error
3	Hardware configuration error
4	Low RAM battery
5	Battery back-up external RAM failure
6	EE checksum error
7	Stack overflow error
8	Input 1 interpolation error
9	Input 2 interpolation error
10	Ground overrange error
11	Ground underrange error
12	Input 1 overrange error
13	Input 1 underrange error
14	Input 2 overrange error
15	Input 2 underrange error
16	Process input overrange error
17	Process input underrange error

11.0 ER2 ERROR CODES

Code No.	Error Type
1	Transmit buffer overflow. - - Clear error; retransmit.
2	Receiver buffer overflow. - - Protocol or syntax violation; retransmit.
3	Framing error. - - Check baud rate, parity, stop bit.
4	Overrun error. - - Check baud rate, parity, stop bit.
5	Parity error. - - Check baud rate, parity, stop bit.
6	Talking out of turn. - - ANSI X3.28 (STX) protocol violation; retransmit.
7	Invalid reply error. - - Verify communications; retransmit.
8	Noise error. - - Check wiring.
20	Command not found. - - Check your program.
21	Equal / question parameter not found. - - Check your program.
22	Incomplete command line. - - Syntax error; retransmit.
23	Invalid character. - - Syntax error; retransmit.
24	Number of characters overflow. - - Numeric syntax error; retransmit.
25	Input out of limit. - - Transmitted value too large or small; retransmit.
26	Read only command. - - Cannot input a value for that parameter; retransmit.
27	No Channel 2 available error.
28	Write only error. - - A read was attempted on a parameter that can only be written to.
30	Request to run invalid. - - Verify a run condition.
31	Request to hold invalid. - - Verify a hold condition.
32	Command invalid in run mode. - - Cannot enter values in run mode; retransmit.
33	Self test mode not active.
35	Number of steps stored is > 99. - - Enter steps only to 99.
36	No file found. - - Check your program; retransmit.
37	No step found. - - Check your program; retransmit.
39	Infinite loop error. - - Check the number of consecutive loops; retransmit.
40	File change error. - - An attempt to resume a changed file occurred; check the program.

12.0 HOW TO CLEAR AN ERROR CODE ON THE VERSATENN

An Error Code / Message results from a problem within the VersaTenn Controller, the configuration of the controller, or with communications if applicable.

With an Error Message, the display will alternately flash the error message with the existing prompt. An Error Message displaying **ER1 XX** is a fatal error and all events and outputs will be turned off. An Error Message displaying **ER2 XX** is not fatal and if in the RUN mode, the controller should continue running.

Note: Before clearing an **ER2** message, you must first put the controller in the HOLD mode.

To clear either Error Message, press MODE until the SYSTEM prompt appears which will flash alternately with the Error Message. Continue with the following:

♦ Begin at SYSTEM. Press ENTER.

♦ Press MODE until either **ER1 XX** or **ER2 XX** appears. Press ENTER. The error will be cleared and the error code will now be zero (0) unless the error is a reoccurring type and it has not been resolved.

Note: Error codes will be masked for one minute in any menu after the ENTER key is pressed. Once the ER1 or ER2 message has been cleared, this prompt will not appear in the normal use of the SYSTEM parameter group.

13.0 PROGRAM EXAMPLES

Example # 1: IBM PC; Using the ANSI X3.28 Protocol

In the ANSI X3.28 protocol, the computer must first “address” the VersaTenn by sending the device ID number to the controller. Once a valid ID number has been sent and a valid response has been received from the controller, communication can begin and continue as long as the **data link escape** control character is not sent to the controller.

All commands begin and end with special control characters, and every command sent either by the computer or controller will have a response. The response may be only a control character or data, which will be terminated by a control character.

When sending data to the VersaTenn, the computer must not send a **carriage return** (**CR**) control character as part of its data string. If the VersaTenn receives a carriage return, an error message will be generated on the VersaTenn. As part of the response from the VersaTenn, a carriage return may be part of the message, <u>but</u> should not be considered a terminating character.

Some instructions require <u>spaces</u>, which separate parameters. These spaces must be part of the string sent to the VersaTenn.

The following assumptions have been made:

1) COM1 of the computer is being used
2) The protocol of the VersaTenn is ANSI X3.28 (STX)
3) The ID number of the VersaTenn is 0.
4) The communication parameters of the VersaTenn are set to 1200 baud, seven bits, odd parity

Program:

```
10      CLS:KEY OFF
20      OPEN “COM1:1200,0,7,1,RS,DS”AS#1
100     ‘check if controller is present
110     Z$ = “0”+CHR$(5)
120     GOSUB 1000
130     GOSUB 2000
140     IF A$<>”0”+CHR$(6) THEN GOTO 3000
150     ‘get temperature setpoint
160     PRINT “Enter temperature setpoint ( -99.9  to  199.9 ) “;
170     INPUT SETPOINT
180     ‘get rid of decimal point
190     SETPOINT = INT(SETPOINT*10)
200     ‘set up setpoint command, include control characters
210     Z$ = CHR$(2)+”= SP1 “+STR$(SETPOINT)+CHR$(3)
220     GOSUB 1000
230     GOSUB 2000
240     ‘check if response is ACK (acknowledge)
250     IF A$<>CHR$(6) THEN GOTO 3500
280     GOSUB 1000
290     GOSUB 2000
300     IF A$<>CHR$(6) THEN GOTO 3500
310     ‘send EOT (end of transmission) to controller
320     Z$ = CHR$(4)
330     GOSUB 1000
```

```
340     'get setpoint from controller
350     GOSUB 2000
360     'must be terminated with ETX (end of text)
370     IF RIGHT$(A$,1)<>CHR$(3) THEN GOTO 3600
380     L = LEN(A$)
390     'get rid of control characters and convert to number
400     A = (VAL(MID$(A$,2,L-2)))/10
410     PRINT:PRINT "Temperature Setpoint = ";A
420     PRINT:PRINT
430     'send acknowledge to controller
440     Z$ = CHR$(6)
450     GOSUB 1000
460     'get EOT (end of transmission) from controller
470     GOSUB 2000
480     GOTO 150
490     END

1000    'output commands to controller but do not send CR
1010    'end output with semi-colon
1020    PRINT #1,Z$;
1030    RETURN

2000    'get data from controller
2010    FOR X = 1 TO 500:NEXT X     'this timer loop may need to be varied on faster computers
2020    A$ = ""
2030    WHILE NOT EOF(1)
2040    A$ = INPUT$(LOC(1),#1)
2050    WEND
2060    RETURN

3000    'if unable to link print error
3010    CLS
3020    PRINT "Error Unable to Address Controller"
3030    END
3500    'if no ACK then print error
3510    CLS
3520    PRINT "No Acknowledge - - Error !!"
3530    END
3600    'if no ETX then print error
3610    CLS
3620    PRINT "No End of Text  - - Error !!"
3630    END
```

END OF PROGRAM # 1