1. Configuration:

Configuration is performed through the USB interface; this may be performed using the Windows application or by communicating with the CT425 using Modbus commands. Configuration parameters may be changed at any time during operation, even when actively controlling an application. All settings may be stored to non-volatile memory so removal of power does not cause loss of configuration.

All settings and control parameters may be written to or read from. Current temperature sensor readings are only available as read-only registers. A full listing of the registers that may be written or read can be found below.

The following settings and registers are available.

a. General settings:

- **Device Name –** A string of 20 ASCII characters used to identify the unit with a familiar name.

- **Temperature Scale –** Determines if the CT425 will interact with the Windows application in Celsius or Fahrenheit scale. All internal calculations and direct USB interactions utilize the Celsius scale.

- **Type –** Set to either Pt100, Pt1000, or Disabled for platinum 100 ohm, platinum 1000 ohm, or no sensor connected, respectively.

- **Offset –** Applies an offset to the respective sensor input. This could be used for wire lead compensation.

- **Normal Process Value Range –** The range of temperatures that causes the LED indicators to display green - indicating normal operation.

b. Output setting (applicable to each output independently):

- **Source –** Determines which sensor input the respective output will use. Multiple outputs may share the same sensor input.

- **Function –** Disabled, PID, On/Off, or Alarm. On/Off and alarm function similarly with the only difference being how the Hysteresis is used. See the Hysteresis section below for detail. When set to disabled, the output is de-energized regardless of the Reverse Acting setting. For the relay output, PID is disabled.

- **Reverse Acting –** Setting this to True causes the respective output to behave in the opposite manner, i.e. normally closed instead of normally open. This may be useful for cooling applications.

- **Over/Under –** Determines whether the output engages when the sensor temperature is over or under the Setpoint. This applies to the Alarm Function only.

- **Setpoint –** Determines the temperature that the selected sensor input is maintained to for PID and On/Off Functions. In the case of the Alarm Function, when the sensor is above or below this temperature the output will activate depending upon if Over or Under is selected, respectively.

- **Hysteresis –** Determines the band around the Setpoint where the output engages and disengages. When using the On/Off function, the Hysteresis value is evenly divided around the Setpoint. For example, if the Setpoint is 55.0°C and the Hysteresis is 0.1°C, the output will engage at 49.95°C and disengage at 50.05°C.

  When using the Alarm function, the Hysteresis is placed on the side of the Setpoint that does not engage the output. For example, if the Setpoint is 55C, Over/Under is Over, and Hysteresis is 0.1C, the output will engage when the sensor exceeds 55.0C, but will disengage when the sensor is less than 54.9C.

  The hysteresis value is meaningless to the PID Function and therefore is greyed out if that Function is used.

- **Minimum Duty Cycle –** Determines the minimum duty cycle, in percent, that the output will reach. This is the duty cycle at the output terminals with respect to the Reverse Acting setting. Normally this value is set to zero, however there may be some applications that require a minimum of more than zero. When the Function is set to Disabled, this setting is irrelevant. Only valid for SSR and logic outputs.

- **Maximum Duty Cycle –** Determines the maximum duty cycle, in percent, that the output will reach. This is the duty cycle at the output terminals with respect to the Reverse Acting setting. Normally this value is set to 100; however there may be some applications that require a maximum of less than 100. When the Function is set to Disabled, this setting is irrelevant. Only valid for SSR and logic outputs.

c. PID settings (applicable to SSR and logic outputs):

- **Method –** Determines if the PID coeffecients (Kp, Ki, Kd) will be entered manually by the user, or if they will be automatically generated from Autotune data by one of several algorithms.

  The information gained from an Autotune procedure is saved so the Method may be changed later by the user. As with all other settings, this is preserved through a power cycle only if written to nonvolatile memory.

  It should be noted that the various options for the Method are the common names given to the various algorithms. For example, Minimal Overshoot may or may not result in the least overshoot among the other available methods due to variability in application, and accuracy of the Autotune performed. It's recommended to try different Methods to find the best performer, and if desired, manually tweak from there.

  An easy, though generic option is to simply set Kp, Ki, and Kd to 100, 2, and 0, respectively. These are the default values the CT425 is shipped with, and work reasonably well in many applications. Kd could instead be set to 1000 for full PID. Again, these values are generic and are by no means intended to work in every application.

- **Kp –** The proportional coefficient.

- **Ki –** The integral coefficient.

- **Kd –** The derivative coefficient.

- **Loop Time –** The loop time for the PID Function, in milliseconds. This is the amount of time between each PID Function calculation, i.e. how often output duty cycle is recalculated based on current conditions. A lower number will theoretically result in faster response time. However, a Loop Time that is too low may result in an unstable system. In general, slow moving processes should have a higher Loop Time. Minimum loop time is 40ms. Maximum is 1000 seconds.

d. Autotune:

- **Autotune Band –** Determines the number of degrees above and below the Intended Setpoint the output is toggled during Autotuning, respectively. Generally this should be set as high as tolerable to improve Autotune results, although 0.5°C is frequently sufficient.

- **Autotune Output Step –** Determines the output duty cycle while the output is engaged. The Autotune algorithm will cycle the output between fully de-energized ("off") and the Output Step. The purpose of the Autotune Output Step is to limit heater power in applications where 100% duty cycle would cause very rapid heating, making autotuning difficult. In general, keep this value as low as possible while allowing the heater to receive enough power to reach the intended temperatures.

- **Intended Setpoint –** Determines the temperature around which the Autotune algorithm will operate. The purpose of this value is to first bring the heater to the approximate temperature at which it would normally be operated before Autotuning, which may improve Autotune results.

- **Autotune Ku –** This is the gain factor determined during the Autotuning process. The value is stored in the CT425 to be used in calculating the PID parameters. See Table 1 below for these calculations.

- **Autotune Tu –** This is the oscillation period determined during the Autotuning process. The value is stored in the CT425 to be used in calculating the PID parameters. See Table 1 below for these calculations.

- **Autotune Type –** This register is used by the software interface what PID Method was last used. This should not be over-written, but will not cause performance issues if it is modified.

e. How Autotune works:

The Autotune feature automatically generates PID coefficients based on measurements performed using the actual sensor and load. Sometimes the values generated by Autotune are sufficient as the final operating values, but for better control, these may have to be manually tweaked.

The CT425 Autotune uses the Ziegler-Nichols method. Autotune operates by first engaging the output to the duty cycle defined by Autotune Output Step until the temperature reaches the Intended Setpoint. The output is left in the engaged state until the temperature exceeds the sum of Intended Setpoint and Autotune Band. The output is disengaged, and the temperature will begin to fall and the output is engaged again once the temperature is less than the difference between Intended Setpoint and Autotune Band. The output then continues to cycle in the same manner for a period of time dependent on the application.

Essentially, the output is toggled to cause the temperature to vary above and below the Intended Setpoint by approximately the Autotune Band. This process is repeated several times until the results are consistent.

The time between temperature peaks, the difference between temperature peaks and valleys, and the output drive duty cycle are used to determine the Ku and Tu values of the Ziegler-Nichols method. Below in Table 1 are the calculations for the Kp, Ki, and Kd parameters for various PID control Methods based on the Ku and Tu values determined during the Autotune process.

Once Autotuning is complete, control is immediately returned to the Control Type selected for that output.

Ziegler-Nichols Method			
Control Type	K_p	K_i	K_d
P	$0.5K_u$	-	-
PI	$0.45K_u$	$1.2K_p/T_u$	-
PD	$0.8K_u$	-	$K_pT_u/8$
Classic PID	$0.6K_u$	$2K_p/T_u$	$K_pT_u/8$
Pessen Integral Rule	$0.7K_u$	$0.4K_p/T_u$	$0.15K_pT_u$
Medium Overshoot	$0.33K_u$	$2K_p/T_u$	$K_pT_u/3$
Minimum Overshoot	$0.2K_u$	$2K_p/T_u$	$K_pT_u/3$

Table 1 - PID Parameter Calculations

There are two main methods that can be implemented in order to Autotune a given application:

- Autotuning can be performed by using the CT425 in the application by writing to the Autotune Band, Autotune Step, and Intended Setpoint registers followed by initiating the Autotuning process by setting the desired Autotune Start register(s) to 0x0001. The Autotuning Status register(s) can then be monitored to determine the progress, and once complete, the CT425 will set the Autotune Start register(s) to 0x0000.

- Autotuning can also be performed by using the CT425, as well as the heater(s) and sensor(s) used in the Windows application. The software interface can then be used to easily initiate and monitor the Autotuning process as well as calculate and store the PID parameters. This method is recommended, if possible.

f. Submitting settings changes:

It is important to note that changes made to the settings of the CT425 are stored in RAM until written to the non-volatile memory. See the Communications section below for instructions to write the current settings to the non-volatile memory.

Communications:

a. USB:

The CT425 uses a standard USB interface operated at 115.2kbps with no flow control using 8 data bits, no parity, and 1 stop bit.

b. Modbus:

The MODBUS interface on the CT425 allow access to all configuration values, sensor readings, and output status. The configuration values may be read or written, while the remaining values are read-only. Configuration values are considered to be in Holding Registers, while the read-only status values are considered to be in Input Registers.

If a written value is not within the valid range, the CT425 will disregard the change to that register. Register limitations and default values can be found in Table 2.

A Function Code is a designator in the Modbus command that specifies the type of command being sent. Three function codes are supported by the CT425: 0x03, 0x04, and 0x10. These are discussed in greater detail in the following sections.

Typically Modbus addresses are allowed in the range from 1 to 247. However, the CT425 is only addressable on Modbus address 1.

c. Holding registers:

The holding registers contain the configuration values. These may be read or written by using function codes 0x03 or 0x10, respectively.

Changes made to any holding register via Modbus are not automatically saved to non-volatile memory, but rather exist only in RAM. In order to save the configuration to non-volatile memory after a change or multiple changes, the following command must be issued, which is essentially writing the value 0x1234 to the non-existent address of 0x0100:

0x01 0x10 0x01 0x00 0x00 0x01 0x02 0x12 0x34 (2 byte CRC)

If multiple changes are to be made, it's best to write those changes first, and then issue the above command to save all changes at once.

The format of the command to read values using Function Code 0x03 is as follows:

0x01	Modbus ID of the CT425
0x03	Function Code
X	16 bit starting register address, MSB first
Y	Number of 16 bit values to return, MSB first
CRC	2 byte cyclic redundancy check value

Example
Read two values from Modbus address 0x01, starting at address 0x0001:

0x01 0x03 0x00 0x01 0x00 0x02 0x95 0xCB

The format of the command to write values is as follows:

0x01	Modbus ID of the CT425
0x10	Function Code
X	16 bit starting register address
Y	16 bit value representing the number of 16 bit registers to write
Z	8 bit value representing the number of 8 bit data values to follow. This value equals (2 * Y)
Value 1	The first 16 bit value to be written
Value n	Additional 16 bit values to be written
CRC	2 byte cyclic redundancy check

d. Input/output registers:

The input/output registers contain read-only values, such as sensor input values and current output states. These are read-only by using Function Code 0x04.

The format of the command to read values using Function Code 0x04 is as follows:

0x01	Modbus ID of the CT425
0x04	Function Code
X	16 bit starting register address, MSB first
Y	Number of 16 bit values to return, MSB first
CRC	2 byte cyclic redundancy check value

2. Modbus Register Listing:

a. Holding registers:

Register Address	Description	Value Format	Valid Range
0x0000	Input 1 RTD Type	C	See Value Format Key
0x0002	Input 1 Sensor Offset	A	-10 to 10°C
0x0004	Input 1 Normal Process Range Lower Temperature	A	-70 to 650°C
0x0006	Input 1 Normal Process Range Upper Temperature	A	-70 to 650°C
0x0008	Input 2 RTD Type	C	See Value Format Key
0x000A	Input 2 Sensor Offset	A	-10 to 10°C
0x000C	Input 2 Normal Process Range Lower Temperature	A	-70 to 650°C
0x000E	Input 2 Normal Process Range Upper Temperature	A	-70 to 650°C
0x0010	SSR Output Source	D	See Value Format Key
0x0012	SSR Output Control Type	E	See Value Format Key
0x0014	SSR Output Setpoint	A	-70 to 650°C
0x0016	SSR Output Hysteresis	A	0 to 100°C
0x0018	SSR Output Reverse Acting	F	See Value Format Key
0x001A	SSR Output PID Kp	A	-1,000,000 to 1,000,000
0x001C	SSR Output PID Ki	A	-1,000,000 to 1,000,000
0x001E	SSR Output PID Kd	A	-1,000,000 to 1,000,000
0x0020	SSR Output Alarm Over/Under	G	See Value Format Key
0x0022	SSR Output Autotune Start	H	See Value Format Key
0x0024	SSR Output Autotune Type	I	See Value Format Key
0x0026	SSR Output Autotune Ku	A	-10,000 to 10,000
0x0028	SSR Output Autotune Tu	A	-10,000 to 10,000
0x002A	SSR Output Autotune Band	A	0 to 720
0x002C	SSR Output Autotune Temperature	A	-70 to 650°C
0x002E	SSR Output Autotune Step	A	0 to 1000 (in tenths of a percent)
0x0030	SSR Output Minimum Duty Cycle	B	0 to 1000 (in tenths of a percent)
0x0032	SSR Output Maximum Duty Cycle	B	0 to 1000 (in tenths of a percent)
0x0034	SSR Output PID Loop Time	B	40 to 10,000 (in ms)
0x0036	Logic Output Source	D	See Value Format Key
0x0038	Logic Output Control Type	E	See Value Format Key
0x003A	Logic Output Setpoint	A	-70 to 650°C
0x003C	Logic Output Hysteresis	A	0 to 100°C
0x003E	Logic Output Reverse Acting	F	See Value Format Key
0x0040	Logic Output PID Kp	A	-1,000,000 to 1,000,000
0x0042	Logic Output PID Ki	A	-1,000,000 to 1,000,000
0x0044	Logic Output PID Kd	A	-1,000,000 to 1,000,000

0x0046	Logic Output Alarm Over/Under	G	See Value Format Key
0x0048	Logic Output Autotune Start	H	See Value Format Key
0x004A	Logic Output Autotune Type	I	See Value Format Key
0x004C	Logic Output Autotune Ku	A	-10,000 to 10,000
0x004E	Logic Output Autotune Tu	A	-10,000 to 10,000
0x0050	Logic Output Autotune Band	A	0 to 720
0x0052	Logic Output Autotune Temperature	A	-70 to 650°C
0x0054	Logic Output Autotune Step	A	0 to 1000 (in tenths of a percent)
0x0056	Logic Output Minimum Duty Cycle	B	0 to 1000 (in tenths of a percent)
0x0058	Logic Output Maximum Duty Cycle	B	0 to 1000 (in tenths of a percent)
0x005A	Logic Output PID Loop Time	B	40 to 10,000 (in ms)
0x005C	Relay Output Source	D	See Value Format Key
0x005E	Relay Output Control Type	E	See Value Format Key
0x0060	Relay Output Setpoint	A	-70 to 650°C
0x0062	Relay Output Hysteresis	A	0 to 100°C
0x0064	Relay Output Reverse Acting	F	See Value Format Key
0x0066	Not Used		
0x0068	Not Used		
0x006A	Not Used		
0x006C	Relay Output Alarm Over/Under	G	See Value Format Key
0x0082	Temperature Scale (only used by Windows Application)	K	See Value Format Key
0x0088	ID String (20 characters, 10 Modbus addresses)		ASCII

Table 2 - Modbus Holding Register Listing

b. Input registers:

Register Address	Description	Value Format
0x0000	Reserved	
0x0002	Firmware Version	N
0x0003	Hardware Version	N
0x0004	Reserved	
0x0006	Reserved	
0x0008	Reserved	
0x000A	Input 1 Temperature	A
0x000C	Input 1 Autotuning Status	J
0x000E	Input 2 Temperature	A
0x0010	Input 2 Autotuning Status	J
0x0012	SSR Duty Cycle	B
0x0014	Logic Duty Cycle	B
0x0016	Relay Duty Cycle	B
0x0018	Detected Power Frequency	B

Table 3 - Modbus Input Register Listing

c. Value Format Key:

<table>
<tr><th>Value Format</th><th>Definition</th></tr>
<tr><td>A</td><td>32-bit IEEE 754 floating point value:
<table><tr><td>S</td><td>e = Exponent</td><td>m = Mantissa</td></tr></table>1 bit   8 bits   23 bits

Value = $(-1)^{S} \times (1.m) \times 2^{e-127}$</td></tr>
<tr><td>B</td><td>Signed 32-bit integer, LSB first.

PID loop time is in units of milliseconds.
Minimum and Maximum Duty Cycle are in units of tenths of a percent.
ie. 500 = 50%</td></tr>
<tr><td>C</td><td>This register holds the Sensor Type setting for the respective input. The LSB is first.
<table><tr><th>Value</th><th>Type</th></tr><tr><td>0x0000</td><td>Off</td></tr><tr><td>0x0001</td><td>Platinum RTD 100Ω</td></tr><tr><td>0x0002</td><td>Platinum RTD 1000Ω</td></tr></table></td></tr>
<tr><td>D</td><td>This register holds the Output Source setting for the respective output. The LSB is first.
<table><tr><th>Value</th><th>Source</th></tr><tr><td>0x0000</td><td>Input 1</td></tr><tr><td>0x0001</td><td>Input 2</td></tr></table></td></tr>
</table>

<table>
<tr><td>E</td><td>This register holds the Output Control Type setting for the respective output. The LSB is first.

<table><tr><th>Value</th><th>Control Type</th></tr><tr><td>0x0000</td><td>Off (Disabled)</td></tr><tr><td>0x0001</td><td>PID</td></tr><tr><td>0x0002</td><td>On/Off</td></tr><tr><td>0x0003</td><td>Alarm</td></tr></table></td></tr>
<tr><td>F</td><td>This register holds the Reverse Acting setting for the respective output. The LSB is first.

<table><tr><th>Value</th><th>Reverse Acting</th></tr><tr><td>0x0000</td><td>Disabled</td></tr><tr><td>0x0001</td><td>Enabled</td></tr></table></td></tr>
<tr><td>G</td><td>This register holds the Alarm Over/Under setting when the respective output is set to the Alarm output type. The LSB is first.

<table><tr><th>Value</th><th>Alarm Over/Under</th></tr><tr><td>0x0000</td><td>Under</td></tr><tr><td>0x0001</td><td>Over</td></tr></table></td></tr>
<tr><td>H</td><td>This register interacts with the controller to begin an Autotune cycle. By setting this register to 0x0001, this triggers the controller to begin an Autotune cycle with the parameters current in the settings. When the Autotune cycle is complete, the CT425 will write 0x0000 to this register. The LSB is first.

<table><tr><th>Value</th><th>Start Autotune Cycle</th></tr><tr><td>0x0000</td><td>False</td></tr><tr><td>0x0001</td><td>True</td></tr></table></td></tr>
<tr><td>I</td><td>This register holds the Autotune Type setting. This register is ONLY used by the software interface when using the Evaulation Board to store what PID Method was last used. This should not be over-written, but will not cause performance issues if it is modified. The LSB is first.

<table><tr><th>Value</th><th>Autotune Type</th></tr><tr><td>0x0000</td><td>Manual Method</td></tr><tr><td>0x0001</td><td>P</td></tr><tr><td>0x0002</td><td>PI</td></tr><tr><td>0x0003</td><td>PD</td></tr><tr><td>0x0004</td><td>Classis PID</td></tr><tr><td>0x0005</td><td>Pessen Integral Rule</td></tr><tr><td>0x0006</td><td>Medium Overshoot</td></tr><tr><td>0x0007</td><td>Minimum Overshoot</td></tr></table></td></tr>
<tr><td>J</td><td>This register holds the status of the Autotuning cycle that is currently in process. This is applicable for each input channel. The LSB is first.

<table><tr><th>Value</th><th>Autotune Status</th></tr><tr><td>0x0000</td><td>Not Autotuning</td></tr><tr><td>0x0001</td><td>Ramp-up</td></tr><tr><td>0x0002</td><td>First cycle</td></tr><tr><td>0x0003</td><td>Second cycle</td></tr></table></td></tr>
<tr><td>K</td><td>This register holds the Temperature Scale setting for the software interface. The LSB is first.

<table><tr><th>Value</th><th>Reverse Acting</th></tr><tr><td>0x0000</td><td>Celsius</td></tr><tr><td>0x0001</td><td>Fahrenheit</td></tr></table></td></tr>
<tr><td>L</td><td>This register holds the current number of times the settings have been written to the on-board flash memory. Even though this block of memory is wear-leveled, it is advised to only write to the non-volatile memory when there is a change.</td></tr>
</table>

	Unsigned 16-bit integer, LSB first.
M	This register holds the Modbus Address of the CT4235. It is an unsigned 16 bit integer used to identify a given unit during Modbus communications. Only 1 is a valid address.
N	This register holds the current firmware or hardware version. Unsigned 16-bit integer, LSB first.

Table 4 - Value Format Key