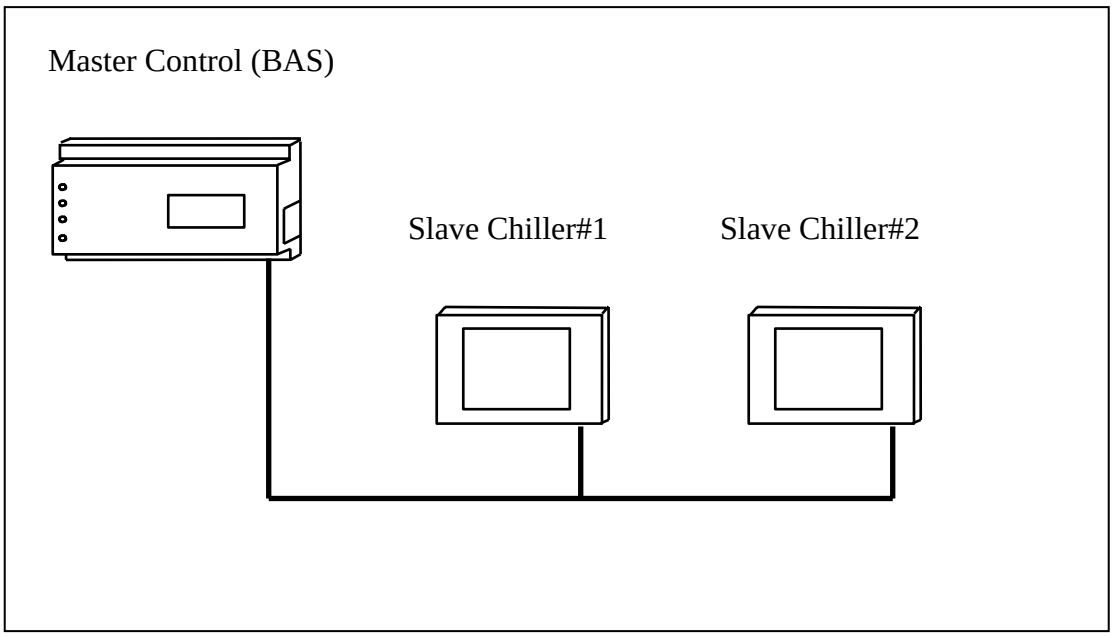
TABLE OF CONTENTS

Introduction

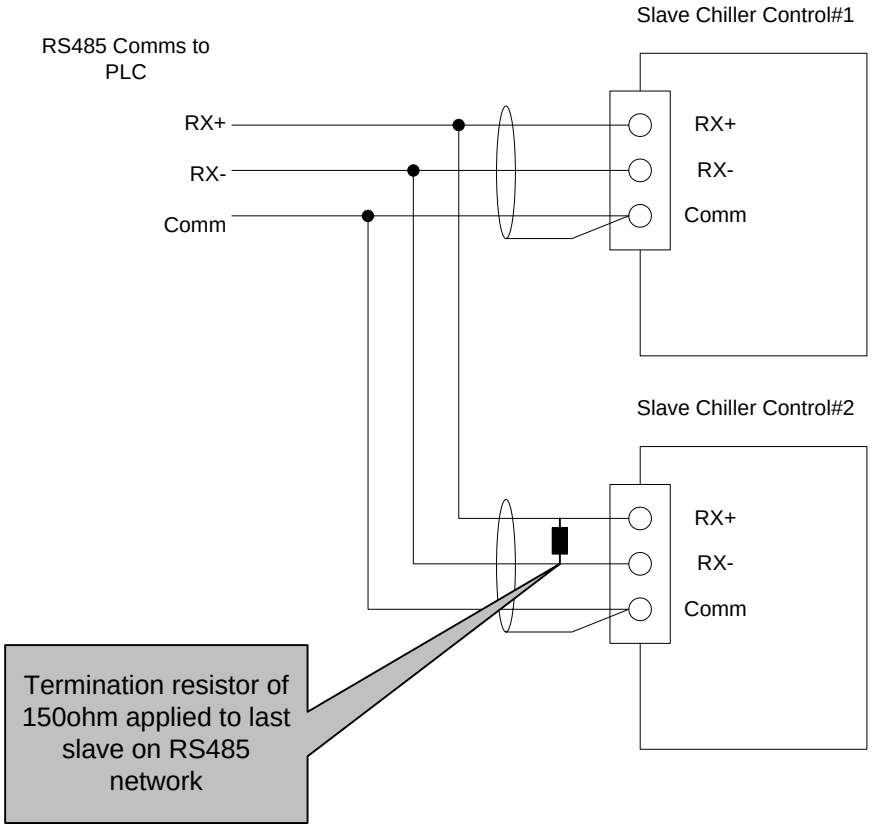
The Kiltech Controller Series uses the Modbus RTU protocol; a protocol widely used in the HVAC and industrial automation industries. This manual explains how the Modbus communication functionality works. For information on how the chiller controller operates, please refer to the complete Kiltech chiller control operating manual.

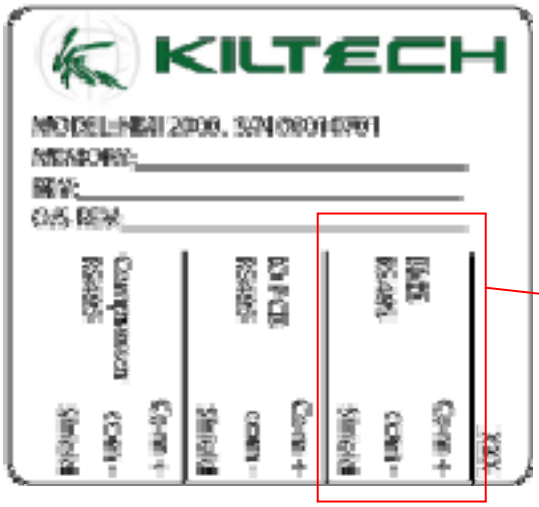
With Modbus communications, data transfer is possible between a single master (PLC) and up to 64 Kiltech Chiller Controllers (the slave). As the master (the BAS) transfers data simultaneously between single slave chiller controllers, the address for each slave must first be set. The slave chiller controller receiving data from the master will execute the instructed function, and then respond to the master (BAS).



Connection Diagram

Interconnection Diagram during RS-485 Transfer





Above: Sample connection diagram between multiple controllers and master plc.

Left: Connection points found on rear of touch panel.

Communication related parameters

Before Kiltech Chiller controller can communicate with a master controller the serial communication parameters must be setup via the touch panel. Communication parameters are found in the “Chiller Commissioning Screen”, a service password is required to gain access to this page – See Kiltech Chiller Control Manual.

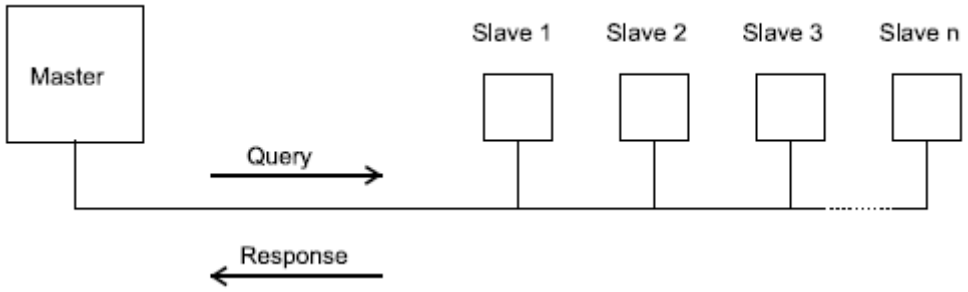
Baud Rate - possible settings =	9600, 19200 & 38400
Stop Bits - possible settings =	1 or 2
Modbus Slave Address =	1 to 64

The modbus RS485 parity is fixed at none.

The inverter uses RTS signal when operating with RS-485 transfer, switching the transfer direction for sending and receiving.

Modbus Messages & the Modbus Protocol

Communication on a MODBUS Network is initiated (started) by a “Master” (BAS) with a “query” to a “Slave”(Chiller Controller). The “Slave “ which is constantly monitoring the network for “Queries” will recognize only the “Queries” addressed to it and will respond either by performing an action (setting a value for example) or by returning a “response”. Only the Master can initiate a query.



In the MODBUS protocol the master can address individual slaves, or, using a special “Broadcast” address, can initiate a broadcast message to all slaves. The SPR and Integra products do not support the broadcast address.

For extra information please see http://www.modbus.org/ on the web.

Modbus Message Format

The MODBUS protocol defines the format for the master's query and the slave's response.
The query contains the device (or broadcast) address, a function code defining the requested action, any data to be sent, and an error-checking field.
The response contains fields confirming the action taken, any data to be returned, and an error-checking field. If an error occurred in receipt of the message, or if the slave is unable to perform the requested action, the slave will construct an error message and send it as its response.

Query
The example illustrates a request for a single 16-bit Modbus Register.

Slave Address	Function Code	Start Address (Hi)	Start Address (Lo)	Number of Points (Hi)	Number of Points (Lo)	Error Check (Lo)	Error Check (Hi)

Slave Address: 8-bit value representing the slave being addressed (1 to 247), 0 is reserved for the broadcast address. The SPR and Integra products do not support the broadcast address.

Function Code: 8-bit value telling the addressed slave what action is to be performed. (3, 4, or 16 are valid for Integra)

Start Address (Hi): The top (most significant) eight bits of a 16-bit number specifying the start address of the data being requested.

Start Address (Lo): The bottom (least significant) eight bits of a 16-bit number specifying the start address of the data being requested.

Number of Points (Hi): The top (most significant) eight bits of a 16-bit number specifying the number of registers being requested.

Number of Points (Lo): The bottom (least significant) eight bits of a 16-bit number specifying the number of registers being requested.

Error Check (Lo): The bottom (least significant) eight bits of a 16-bit number representing the error check value.

Error Check (Hi): The top (most significant) eight bits of a 16-bit number representing the error check value.

Response
The example illustrates the normal response to a request for a single 16-bit Register.

Slave Address	Function Code	Byte Count	Data (Hi)	Data (Lo)	Error Check (Lo)	Error Check (Hi)

Slave Address: 8-bit value representing the address of slave, which has just responded.

Function Code: 8-bit value which, when a copy of the function code in the query, indicates that the slave recognized the query and has responded. (See also Exception Response).

Byte Count: 8-bit value indicating the number of data bytes contained within this response

Data (Hi): The top (most significant) eight bits of a 16-bit number representing the register(s) requested in the query.

Data (Lo): The bottom (least significant) eight bits of a 16-bit number representing the register(s) requested in the query.

Error Check (Lo): The bottom (least significant) eight bits of a 16-bit number representing the error check value.

Error Check (Hi): The top (most significant) eight bits of a 16-bit number representing the error check value.

Exception Response
If an error is detected in the content of the query (excluding parity errors and Error Check mismatch), the function code will be modified to indicate that the response is an error response (called an exception response), and the data bytes will contain a code that describes the error. The exception response is identified by the function code being a copy of the query function code but with the most-significant bit set to logic ‘1’.

Slave Address	Function Code’	Error Code	Error Check (Lo)	Error Check (Hi)

Slave Address: 8-bit value representing the address of slave, which has just responded.

Function Code: 8 bit value which is the function code in the query OR'ed with Hex (80), indicating the slave either does not recognize the query or could not carry out the action requested.

Error Code: 8-bit value indicating the nature of the exception detected. (See “Exception Codes“ in the section “Product Information for a list of SPR and Integra supported codes).

Error Check (Lo): The bottom (least significant) eight bits of a 16-bit number representing the error check value.

Error Check (Hi): The top (most significant) eight bits of a 16-bit number representing the error check value.

Modbus RTU Mode Transmission

In RTU (Remote Terminal Unit) mode, each 8-bit byte in a message contains two 4-bit hexadecimal characters. The main advantage of this mode is that it's greater character density allows better data throughput than ASCII for the same baud rate, however each message must be transmitted in a continuous stream.

The format for each byte in RTU mode is:

Coding System:	8-bit binary, hexadecimal 0-9, A-F Two hexadecimal characters contained in each 8-bit field of the message
Bits per Byte:	1 start bit, 8 data bits, least significant bit sent first 1 parity bit for even/odd parity; no parity bit for no parity 1 stop bit if parity is used; 2 stop bits if no parity
Error Check Field:	Cyclical Redundancy Check (CRC)

MODBUS Message Timing (RTU Mode)

A MODBUS message has defined beginning and ending points. The receiving devices recognize the start of the message, read the "Slave Address" to determine if they are being addressed and know when the message is completed so that they can use the Error Check bytes to confirm the integrity of the query.
Partial messages can be detected and discarded:

In RTU mode, messages start with a silent interval of at least 3.5 character times.

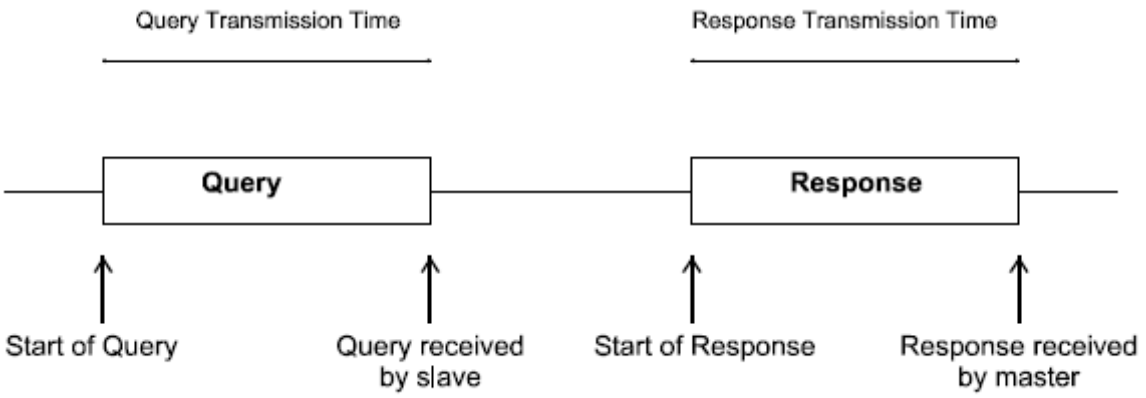
The first field then transmitted is the device address.

The allowable characters transmitted for all fields are hexadecimal 0-9, A-F. Devices monitor the network bus continuously, including during the 'silent' intervals. When the first field (the address field) is received, each device decodes it to find out if it is the addressed device. If the device determines that it is the one being addressed it decodes the whole message and acts accordingly, if it is not being addressed it continues monitoring for the next message.
Following the last transmitted character, a silent interval of at least 3.5 character times marks the end of the message. A new message can begin after this interval.

In the Integra 1000 and 2000, a silent interval of 60msec minimum is required in order to guarantee successful reception of the next request.
The entire message frame must be transmitted as a continuous stream. If a silent interval of more than 1.5 character times occurs before completion of the frame, the receiving device flushes the incomplete message and assumes that the next byte will be the address field of a new message.
Similarly, if a new message begins earlier than 3.5 character times following a previous message, the receiving device will consider it a continuation of the previous message. This will result in an error, as the value in the final CRC field will not be valid for the combined messages.

Modbus Error Checking Methods

Standard MODBUS serial networks use two error checking processes, the error check bytes mentioned above check message integrity whilst Parity checking (even or odd) can be applied to each character in the message. The master is configured by the user to wait for a predetermined timeout interval. The master will wait for this period of time before deciding that the slave is not going to respond and that the transaction should be aborted. Care must be taken when determining the timeout period from both the master and the slaves' specifications. The slave may define the 'response time' as being the period from the receipt of the last bit of the query to the transmission of the first bit of the response. The master may define the 'response time' as period between transmitting the first bit of the query to the receipt of the last bit of the response. It can be seen that message transmission time, which is a function of the baud rate, must be included in the timeout calculation.



Parity Checking

If parity checking is enabled - either Even or Odd Parity is specified - the quantity of "1's" will be counted in the data portion of each of the eight bits in the character. The parity bit will then be set to a 0 or 1 to result in an Even or Odd total of "1's".
Note that parity checking can only detect an error if an odd number of bits are picked up or dropped in a character frame during transmission, if for example two 1's are corrupted to 0's the parity check will not find the error.
If No Parity checking is specified, no parity bit is transmitted and no parity check can be made. An additional stop bit is transmitted to fill out the character frame when 2 stop bits are selected. If No Parity checking is specified and one stop bit is selected the character is effectively shortened by one bit.

CRC Checking

The error check bytes of the MODBUS messages contain a Cyclical Redundancy Check (CRC) value that is used to check the content of the entire message. The error check bytes must always be present to comply with the MODBUS protocol; there is no option to disable it. The error check bytes represent a 16-bit binary value, calculated by the transmitting device. The receiving device must recalculate the CRC during receipt of the message and compare the calculated value to the value received in the error check bytes. If the two values are not equal, the message should be discarded.
The error check calculation is started by first pre-loading a 16-bit register to all 1's (i.e. Hex (FFFF)) each successive 8-bit byte of the message is applied to the current contents of the register. Note: only the eight bits of data in each character are used for generating the CRC, start bits, stop bits and the parity bit, if one is used, are not included in the error check bytes.
During generation of the error check bytes, each 8-bit character is exclusive OR'ed with the register contents. The result is shifted in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position. The LSB prior to the shift is extracted and examined. If the LSB was a 1, the register is then exclusive OR'ed with a pre-set, fixed value. If the LSB was a 0, no exclusive OR takes place.
This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next 8-bit byte is exclusive OR'ed with the register's current value, and the process repeated. The final contents of the register, after all the bytes of the message have been applied, is the error check value.
In the following pseudo code "ErrorWord" is a 16-bit value representing the error check values.

```
BEGIN
        ErrorWord = Hex (FFFF)
                FOR Each byte in message
                        ErrorWord = ErrorWord XOR byte in message
                                FOR Each bit in byte
                                        LSB = ErrorWord AND Hex (0001)
                                        IF LSB = 1 THEN ErrorWord = ErrorWord – 1
                                        ErrorWord = ErrorWord / 2
                        IF LSB = 1 THEN ErrorWord = ErrorWord XOR Hex (A001)
                NEXT bit in byte
        NEXT Byte in message
END
```

Function Codes

The function code part of a MODBUS message defines the action to be taken by the slave. The Kiltech Chiller Control products support the following function codes:

Function 3:	Read Holding Registers.
Function 16:	Pre-Set Multiple Registers
Function 8:	Diagnostics

Modbus Commands Supported

Read Holding Registers

MODBUS code 03 reads the contents of the 4X registers (40,000 Range).

Example

The following query will request the prevailing 'Enable Status' and the 'Cooling set point' from slave one:

Field Name	Example (Hex)
Slave Address	01
Function	03
Starting Address High	00
Starting Address Low	00
Number of Points High	00
Number of Points Low	02
Error Check Low	C4
Error Check High	0B

The slave unit may respond as follows with two data points:

Field Name	Example (Hex)
Slave Address	01
Function	03
Byte Count	04
Data, High Word, High Byte	3F
Data, High Word, Low Byte	80
Data, Low Word, High Byte	00
Data, Low Word, Low Byte	00
Error Check Low	F7
Error Check High	CF

Write Holding Registers

MODBUS code 16 decimal (10 Hex) writes the contents of the 4X registers.

Example

The following query will set the Enable to off:

Field Name	Example (Hex)
Slave Address	01
Function	10
Starting Address High	00
Starting Address Low	00
Number of Registers High	00
Number of Registers Low	01
Byte Count	02
Data High	00
Data Low	00
Error Check High	3D
Error Check Low	45

The following response from the slave would indicate a successful write

Field Name	Example (Hex)
Slave Address	01
Function	10
Starting Address High	00
Starting Address Low	00
Number of Registers High	00
Number of Registers Low	01
Error Check High	41
Error Check Low	C6

Exception Codes

Whenever a Kiltech Chiller Control product receives a MODBUS message with valid parity and error check but which contains some other error (e.g. a request to set a register to an illegal value or a request for part of a floating point variable), an Exception code will be generated. (The message format is shown in the message Formats section) Exceptions are indicated by a value in the function code field of the response greater than Hex (80), obtained by OR'ing the original function code in the query with Hex (80). For example, if a function code of Hex (84) and exception code 2 were present in an exception response this indicates that a function 4 query (Read Holding Registers) has resulted in an illegal data address error.

The error codes and the corresponding types of error returned by the models covered in this guide are given in the following table:

Exception Code	MODBUS name	Description	Reported by
01	Illegal Function	The function code is not supported by the product	1000, 2000, 1540, 1560, 1580
02	Illegal Data Address	Attempt to access an invalid address or an attempt to read or write part of a floating point value	1000, 2000, 1540, 1560, 1580
03	Illegal Data Value	Attempt to set a floating point variable to an invalid value	1000, 2000, 1540, 1560, 1580
05	Slave Device Failure	An error occurred when the instrument attempted to store an update to it's configuration	1540, 1560, 1580

Product Specific Information

Modbus Register List

Variable Name	Data Type	Modbus Register
nviChillerEnable	// snvt_switch - 0/1	40001
nviCoolSetpt	// snvt_temp_p -	40002
nvoOnOff	// snvt_switch - 0/1	40003
nvoActiveSetpt	// snvt_temp_p	40004
nviCapacityLim	// snvt_lev_percent	40005
nviEntChWTemp	// snvt_temp_p	40006
nviMode	// snvt_hvac_mode	40007
nviHeatSetpt	// snvt_temp_p	40008
nvoActualCapacity	// snvt_lev_percent	40009
nvoCapacityLim	// snvt_lev_percent	40010
nvoLvgCHWTemp	// snvt_temp_p	40011
nvoEntCHWTemp	// snvt_temp_p	40012
nvoEntCndWTemp	// snvt_temp_p	40013
nvoLvgCndWTemp	// snvt_temp_p	40014
nvoLiqRefTemp	// snvt_temp_p	40015
nvoAlarmDescr	// snvt_chlr_type	40016
nvoChillerStat	// snvt_chlr_status	40017
nviOutdoorTemp	// snvt_temp_p	40018
nvoOutdoorTemp	// snvt_temp_p	40019
nviOutdoorRH	// snvt_lev_percent	40020
nvoOutdoorRH	// snvt_lev_percent	40021
nvoChlrPwr	// snvt_power_kilo - total power	40022
nviChlrPwrLim	// snvt_power_kilo - total power limit	40023
nvoChlrPwrLim	// snvt_power_kilo	40024
nvoEntCndWSetpt	// snvt_temp_p	40025
nvoActiveEntCndWSetpt	// snvt_temp_p	40026
nvoChlrState	// snvt_state_16	40027
nvoRpm_1	// snvt_rpm	40028
nvoRpm_2	// snvt_rpm	40029
nvoRpm_3	// snvt_rpm	40030
nvoRpm_4	// snvt_rpm	40031
nvoRpm_5	// snvt_rpm	40032
nvoRpm_6	// snvt_rpm	40033
nvoDrvPwr_1	// snvt_power_kilo	40034
nvoDrvPwr_2	// snvt_power_kilo	40035
nvoDrvPwr_3	// snvt_power_kilo	40036
nvoDrvPwr_4	// snvt_power_kilo	40037
nvoDrvPwr_5	// snvt_power_kilo	40038
nvoDrvPwr_6	// snvt_power_kilo	40039
nvoDrvRunHours_1	// snvt_time_hour	40040
nvoDrvRunHours_2	// snvt_time_hour	40041
nvoDrvRunHours_3	// snvt_time_hour	40042
nvoDrvRunHours_4	// snvt_time_hour	40043

nvoDrvRunHours_5	// snvt_time_hour	40044
nvoDrvRunHours_6	// snvt_time_hour	40045
nvoIgvPosition_1	// snvt_lev_percent	40046
nvoIgvPosition_2	// snvt_lev_percent	40047
nvoIgvPosition_3	// snvt_lev_percent	40048
nvoIgvPosition_4	// snvt_lev_percent	40049
nvoIgvPosition_5	// snvt_lev_percent	40050
nvoIgvPosition_6	// snvt_lev_percent	40051
nvoPressure_Suction_1	// snvt_press	40052
nvoPressure_Suction_2	// snvt_press	40053
nvoPressure_Suction_3	// snvt_press	40054
nvoPressure_Suction_4	// snvt_press	40055
nvoPressure_Suction_5	// snvt_press	40056
nvoPressure_Suction_6	// snvt_press	40057
nvoPressure_Discharge_1	// snvt_press	40058
nvoPressure_Discharge_2	// snvt_press	40059
nvoPressure_Discharge_3	// snvt_press	40060
nvoPressure_Discharge_4	// snvt_press	40061
nvoPressure_Discharge_5	// snvt_press	40062
nvoPressure_Discharge_6	// snvt_press	40063
nvoComp_State_1	// snvt_state_16	40064
nvoComp_State_2	// snvt_state_16	40065
nvoComp_State_3	// snvt_state_16	40066
nvoComp_State_4	// snvt_state_16	40067
nvoComp_State_5	// snvt_state_16	40068
nvoComp_State_6	// snvt_state_16	40069
nvoCCAlarm_State_1	// snvt_state_16	40070
nvoCCAlarm_State_2	// snvt_state_16	40071
nvoCCAlarm_State_3	// snvt_state_16	40072
nvoCCAlarm_State_4	// snvt_state_16	40073
nvoCCAlarm_State_5	// snvt_state_16	40074
nvoCCAlarm_State_6	// snvt_state_16	40075
nvoCCFault_State_1	// snvt_state_16	40076
nvoCCFault_State_2	// snvt_state_16	40077
nvoCCFault_State_3	// snvt_state_16	40078
nvoCCFault_State_4	// snvt_state_16	40079
nvoCCFault_State_5	// snvt_state_16	40080
nvoCCFault_State_6	// snvt_state_16	40081
nvoBRG_State_1	// snvt_state_16	40082
nvoBRG_State_2	// snvt_state_16	40083
nvoBRG_State_3	// snvt_state_16	40084
nvoBRG_State_4	// snvt_state_16	40085
nvoBRG_State_5	// snvt_state_16	40086
nvoBRG_State_6	// snvt_state_16	40087
nvoMTR_State_1	// snvt_state_16	40088
nvoMTR_State_2	// snvt_state_16	40089
nvoMTR_State_3	// snvt_state_16	40090
nvoMTR_State_4	// snvt_state_16	40091
nvoMTR_State_5	// snvt_state_16	40092

nvoMTR_State_6	// snvt_state_16	40093
nvoCoolingTowerSpd	// snvt_lev_percent	40094
nvoCondPumpSpd	// snvt_lev_percent	40095
nvoChillerModel	// snvt_count	40096
nvoDigitalInputs	// Binary representation of di status	40097
nvoDigitalOutputs	// Binary representation of relay status	40098
nvoNextComprStarting	// Binary representation of relay status	40099

Modbus Data Description

Modbus data in the Kiltech controller consists of the following types of data:

- **Switch points**. Switch data have only to possible values 0 or 1 and are used to represent things like enable/ disable function status.

- **Temperature points**. Temperature data is represented as ˚Cx10 or ˚Fx10; temperature data may be read only such as a leaving air/ water measurement or may be read/write such as a temperature set point.

- **Percentage points**. Percentage points represent data such as percent chiller loading or percentage motor speed, these points may be read and write.

- **Electrical consumption in kilowatts**, these points represent consumed electrical power from compressors or fans in kWx10 and may be read or write (write applies total kW input limit setting).

- **State points,** these points are integer values that represent a product defined state such as “Chiller Running” or “Chiller Faulted”.

- **RPM points,** these points represent actual motor shaft speeds in revolutions per minute scaled 1:1.

- **Pressure points,** these points represent system pressures scaled psia x10 or kPa x10.

- **Binary or Bit points,** these points are 16bit integer data where each represents a different piece of information; for example each bit may represent a different type of error on a compressor.

Modbus Data Scaling In PLC/ BAS

Temperature Conversion

Temperatures points exported from the Kiltech Chiller controller are represented as degrees multiplied by a factor of 10.

Example:

67.7˚F is transported across the modbus protocol as 667.

Power Input Conversion

Power in kilowatts is represented on the modbus protocol as kW multiplied by a factor of 10.

Example:

133.7kW is transported across the modbus protocol as 1337.

Pressure Conversion

Pressure points in kPa and psi is represented on the modbus protocol as absolute pressure multiplied by a factor of 10.

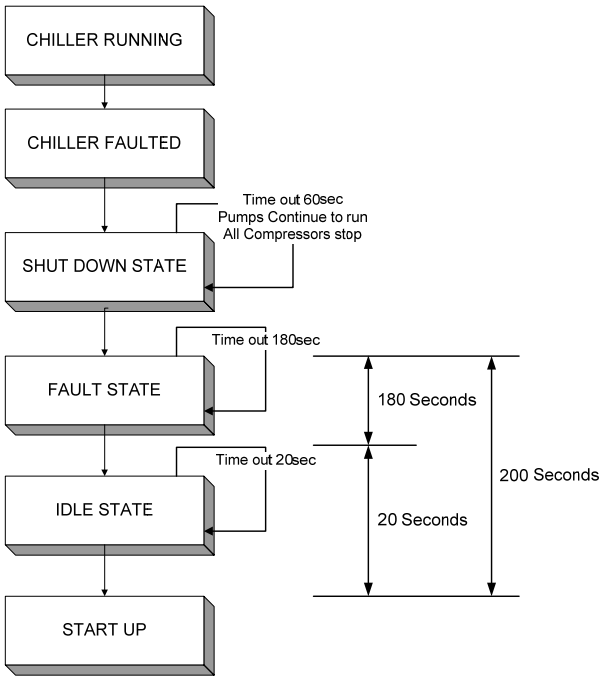
Example:

45.5psi is transported across the modbus protocol as 455.

To convert to gauge pressures subtract 14.7psi from scaled number when controller is set to imperial units and 100kPa from scaled number in metric.

Chiller Fault Codes

Active chiller faults are read from register 40017. Faults are represented in bit format each bit representing a different chiller fault. The chiller controller will attempt to reset each fault automatically approx four minutes after the chiller is shutdown due to fault condition except in the case where the maximum number of starts per day has been exceeded.
A chiller that has stopped on fault goes through the following sequence before been able to restart:



Bit Value	Fault Description	Integer Value	Hex Value
1	Low Chilled Water Temp	1	0x0001
2	Low suction pressure fault	2	0x0002
3	High discharge pressure fault	4	0x0004
4	Over current fault.	8	0x0008
5	High evaporator delta temp fault	16	0x0010
6	No chilled water flow fault	32	0x0020
7	No condenser water flow fault	64	0x0040
8	Chiller failed to start fault	128	0x0080
9	External HP/LP pressure cutout safety fault	256	0x0100
10	Emergency stop button activated fault	512	0x0200
11	Gas leakage input fault	1024	0x0400
12	Maximum starts per day exceeded fault	2048	0x0800
13	Loss of I/O module communications fault	4096	0x1000
14	No Compressors Available to Run	16348	0x2000
15	Spare	32768	0x4000
16	Spare	65536	0x8000

*For more details on bit description please see section - Program Examples.

Chiller Operating Mode or State

At any one time the chiller is operating in one of eight defined states each state is represented by an integer value, they are:

State Description	Integer Value
Idle State	0
Pull down State	1
Run State	2
Compressor Stage Up State	3
Compressor Stage Down State	4
Alarm Avoidance State	5
Fault State	6
Shutting Down State	7

Modbus register 40027 holds the chiller operating state the state.

Compressor Operating State

At any one time each compressor is operating in one of thirteen kiltech defined states, each state is represented by an integer value, and they are:

Compressor State Description	Integer Value
Offline State, No power.	0
Resetting IGV & Drive	1
Ready to run state	2
Ramping to min operating speed	3
Running normally	4
Running in alarm avoidance mode “capacity limited”	5
Resetting with a fault present	6
Idle with a fault present	7
Clearing fault	8
Compressor Locked out, discharge pressure fault or over current fault generated inside compressor. Compressor requires power down to reset fault	9
Compressor Locked out by chiller. Requires manual reset via chiller controller.	10
Compressor Exceeded Maximum number of starts per hour.	11
Compressor Exceeded Maximum number failed starts per hour.	12
Reserved	13

Modbus registers 40064 to 40069 hold the compressor state for compressors 1 to 6.

Compressor Alarms

The Turbocor compressor has 9 alarms defined. An alarm in the compressor is NOT a condition that stops the compressor it is a condition where the compressor operates at reduced capacity in order to avoid a fault. Compressor alarms are resented in bit format – see table below:

Bit Value	Fault Description	Integer Value	Hex Value
1	High Inverter Temperature Alarm	1	0x0001
2	High Discharge Temperature Alarm	2	0x0002
3	Low Suction Pressure Alarm	4	0x0004
4	High Discharge Pressure Alarm	8	0x0008
5	High 3ph Current Alarm	16	0x0010
6	High Rotor Temperature Alarm	32	0x0020
7	Low Leaving Temperature Alarm	64	0x0040
8	High Pressure ratio Alarm	128	0x0080
9	High SCR Temperature Alarm	256	0x0100

*For more details on bit description please see section - Program Examples.

Compressor alarm registers may be found in locations 40070 to 40075. If the register is equal to zero then compressor has no active alarms.

Compressor Faults

The Turbocor compressor has thirteen general faults defined; each fault is represented as a different bit in a 16bit register.

Bit Value	Fault Description	Integer Value	Hex Value
1	High Inverter Temperature Fault	1	0x0001
2	High Discharge Temperature Fault	2	0x0002
3	Low Suction Pressure Fault	4	0x0004
4	High Discharge Pressure Fault	8	0x0008
5	High 3ph Current Fault	16	0x0010
6	High Rotor Temperature Fault	32	0x0020
7	Low Leaving Temperature Fault	64	0x0040
8	High Pressure Ratio Fault	128	0x0080
9	Generic Motor or Bearing Fault	256	0x0100
10	Faulty Compressor Sensor Fault	512	0x0200
11	High SCR Temperature Fault	1024	0x0400
12	Compressor Locked Out	2048	0x0800
13	Motor Winding Over Temperature Fault	4096	0x1000
14	Not Used	16348	0x2000
15	Not Used	32768	0x4000
16	Not Used	65536	0x8000

*Compressor fault registers are located in registers 40076 to 40081. If the fault register is equal to zero then the compressor has no faults.

Compressor Magnetic Bearing Faults

The Turbocor compressor has eight defined magnetic bearing faults each fault is represented as a different bit in a 16bit register.

Bit Value	Fault Description	Integer Value	Hex Value
1	Bearing Calibration Failed	1	0x0001
2	Startup Check Failed	2	0x0002
3	Thrust Bearing Displacement Over Limit	4	0x0004
4	Thrust Bearing Over Current Fault	8	0x0008
5	Front Bearing Displacement Over Limit	16	0x0010
6	Front Bearing Over Current Fault	32	0x0020
7	Rear Bearing Displacement Over Limit	64	0x0040
8	Rear Bearing Over Current Fault	128	0x0080

*Compressor bearing fault registers are located in registers 40082 to 40087. If the bearing fault register is equal to zero then the compressor has no faults.

Compressor Motor Faults

The Turbocor compressor has sixteen defined motor faults each fault is represented as a different bit in a 16bit register.

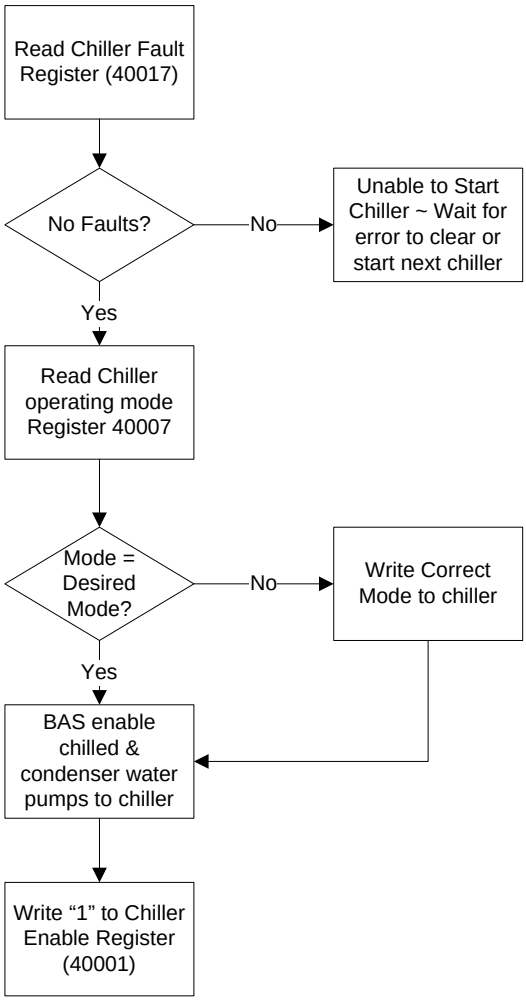
Bit Value	Fault Description	Integer Value	Hex Value
1	Motor Single phase over current	1	0x0001
2	DC Bus Over Voltage	2	0x0002
3	Motor High Current Warning	4	0x0004
4	Motor High Current Fault	8	0x0008
5	Inverter Error Signal Active	16	0x0010
6	Over Current during Startup – Rotor May Be Locked	32	0x0020
7	Bearing Error Active	64	0x0040
8	Bearing Warning Active	128	0x0080
9	Output voltage on motor generates no current	256	0x0100
10	Soft start error detected	512	0x0200
11	24VDC Out of bounds error	1024	0x0400
12	Motor back emf low	2048	0x0800
13	Eeprom Checksum error	4096	0x1000
14	Generator mode active	16348	0x2000
15	SCR Phase loss	32768	0x4000
16	System in startup mode	65536	0x8000

*Compressor motor fault registers are located in registers 40088 to 40093. If the bearing fault register is equal to zero then the compressor has no faults.

Example Program Usage

Starting and Stopping the Chiller

To start the chiller via the modbus communication protocol the following sequence is applicable:



To stop the chiller write '0" to enable register 40001.

Changing the Chiller Set point

To change the chiller set point via the modbus communication protocol the set point temperature must be multiplied by a factor of ten and then sent to register 40002.

Example:

Desired BAS set point = 45.0˚F,
Value to write to chiller controller = 45 x10 = 450

Setting a Chiller Demand Limit

On occasion it may be necessary to demand limit the chiller such as in a case where loading shedding is required to avoid peak demand charges. A demand limit may be applied to the chiller by writing a value between 25 and 100 to register 40005.
Setting the capacity limit register to a value to 50 will cause the chiller capacity to limited at 50%.

*Note this value is initialized on boot to 100%, if this value is changed via BAS it should be set back to 100 once the limit period required has expired – failing to do will cause the chiller to be locked into a limit state.

Changing the Chiller Operational Mode

The Kiltech controller is programmed to operate the chiller in one of three modes:

- HVAC Heat Mode = 0
- HVAC Cool Mode = 1
- Saturated Suction Temperature Mode = 2

To set the chiller to a different operating mode the chiller must be stopped then a new mode may be written to register 40007.

Setting the outside air Temperature and Relative Humidity

If the Kiltech chiller controller is setup to control the cooling tower fan and or variable speed condenser water pump the controller requires measurement of the outside air temperature and relative humidity.

The outside air temperature and relative humidity may be hard wired to the controllers I/O or supplied via the communication bus via the BAS.

To set the outside air temperature via the modbus network write the outside air temperature multiplied by ten to register 40018. To set the outside air relative humidity writes a value with no scaling between 10 and 100 to register 40020.

Detecting Next Compressor Starting

To detect which compressor is to be started before it is actually enabled a BAS system may read register 40099. Register 40099 sets a different bit for each compressor before it starts. The bit remains set until the compressor is stopped.

Bit Value	Description	Integer Value	Hex Value
1	Compressor #1 START REQUEST	1	0x0001
2	Compressor #2 START REQUEST	2	0x0002
3	Compressor #3 START REQUEST	4	0x0004
4	Compressor #4 START REQUEST	8	0x0008
5	Compressor #5 START REQUEST	16	0x0010
6	Compressor #6 START REQUEST	32	0x0020

Sample Software Routines

Bit Detection in Basic

```
Public Function ExamineBit(ByVal Byte_ As Long, ByVal Bit As Long) As Boolean
' The ExamineBit function will return True or False depending on
' the value of the nth bit (Bit%) of an integer (Byte%).
    Dim mask As Double

    ' Create a bitmask with the 2 to the nth power bit set:
    mask = 2 ^ Bit
    ' Return the truth state of the 2 to the nth power bit:
    ExamineBit = ((Byte_ And mask) > 0)
End Function
```