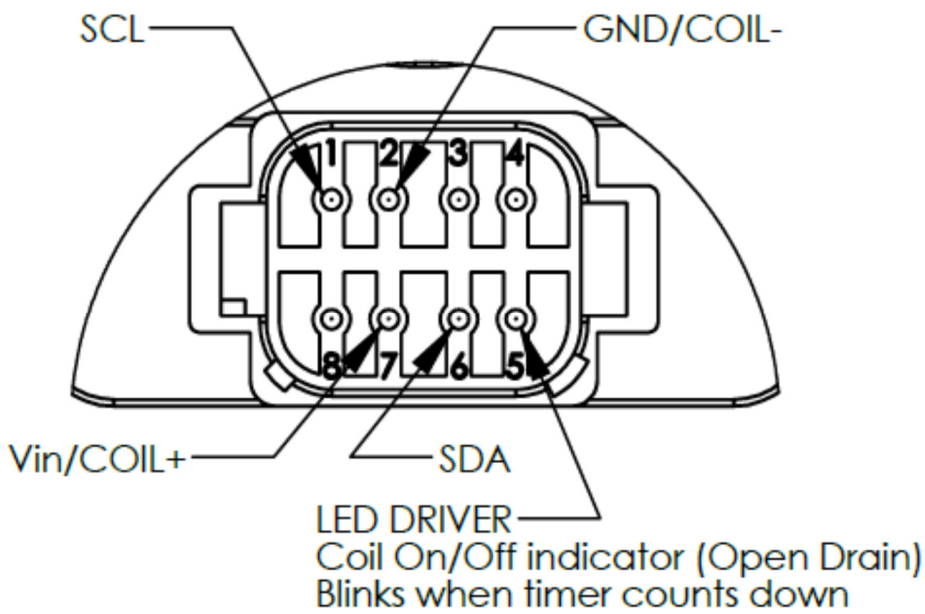
GIGAVAC Contactors I²C Communication

Document Revision: 3

Product models: MXST15/16-mm-ss, delay on break contactors.

Attention: Read this instruction entirely for a top-level-feel of what you prefer to focus on the details to accomplish your goal.

Hardware Connection



Each contactor needs 4 wires hookup to the connector in order to communicate in I²C mode: pin1 to SCL, pin 2 to GND, pin 6 to SDA and pin 7 to power. Two resistors 10K each internally pull up to 5V on the SCL and SDA lines. If a need for disconnecting the internal pull-ups resistors from the users, future hardware revision can implement pull-up (enable/disable) option. For now, multiple units on the network, combine internal pull-up total resistance of about 4.7K will work. At low speed, 10K pull-up alone will also fine. These MXST15/16 products always communicate in slave interface, and the users are in master role on the bus.

GIGAVAC®
ADVANCED SWITCHING SOLUTIONS

Operating modes

The relays have two operating modes: Voltage-control-mode and I²C-control-mode. When the products leave the factory, they are in the voltage-control mode (default). To commutate with the relay while it is in voltage control-mode follow the "Power-Up Procedure".

The 10K pull-up resistors permanently installed on the relays is for one to one master and slave communication at start-up. Some "USB to I²C" converts (master side) need to remove the pull-up resistors for the contactor to start its I²C mode. After I²C mode is activated, the removal of master pull-up is not necessary. At first power-apply to the unit, it waits for 20ms to see if any request from the master. If no signal on the bus, it will start its operation. The mode of operation (voltage control or I²C control) depends on the I²C enable byte at the contactor's memory. This I²C enable byte is default to disable state when the customers first receive the units; the following steps will bring the unit into I²C mode:

Power-Up Procedure:

1. Make sure the relay power is off.
2. Master and slave are properly connected.
3. Some I²C masters need to remove pull-up resistors in order not to receive its reflection.
4. Master send out an I²C read to address 0x7F and read-back from the same address.
5. The master keeps repeating step 4 for less than 20ms typically every 10ms to 15ms.
6. Turn relay power on.
7. When the relay is fired up and receives a read-command from the master, it will send out an 'I' character (0x49) and stay in the loop to listen.
8. When the master receives an 'I', the relay is ready for commands from the master.

The above steps bring the relay into I²C control-mode in a configure-state. At this state you can send command and receive data to and from the contactor (see command list for details).

The 2 states of I²C mode: configure-state and operate-state. Configure-state is activated by the above start-up procedure. The operate-state is enabled by sending an I²C enable command. After the I²C is enabled, cycle power and the unit will start in I²C mode with operate-state.

The starting of configure-state and operate-state is different, but they are no different in functionality. The configure-state always uses I²C address 0x7F and the operate-state uses I²C address from the command setting. No matter the unit will start in voltage-control mode or I²C control-mode, the start-up procedure will be available every time.

Important note: firmware version A cannot control the contactors in I²C configure-state.

In case you forget what address had been set to the relay, use power-up procedure to bring the relay into the configure-state and read the address setting.

Note: remove pull-up resistors on the master side is only for startup procedure. After the I^2C is activated, the mater can have the pull-up resistors stay to talk to the slave. For startup, some I^2C masters may not need to remove its pullup, and some others probably need no pullup. It depends on the type of I^2C module or I^2C controller is being used.

I^2C Version

This I^2C controller chip supports clock stretching and 10 bit address which is up to version 2.1, and data rate is up to 400Kbits/s. Because little data to exchange in this application, it is better to operate at lower speed for clear messages typically 100Kbits/s. These products use 7 bit address for simple and quick. Therefore, 10 bit address is not implemented. If you want you use lower bit-rate at power-up configuration make sure all the bits (start, address, ACK, read-back…) can fit in less than 20ms time frame with the chosen speed because the relay only wait for 20ms at power-up. After establishing communication with the relay, lower bitrates will have no restrictions. If the I^2C enable-bit is set, it will enter I^2C mode automatically at power-up.

Communication Sequence & Data Format

For complete I^2C protocol information, refer to I^2C specification, data-definitions and bit shifting for details. In general, this is what happens: When relay receives a complete byte from the master, it will send out an ACK. When the relay transmits data, it expects the master to send back an ACK after each byte. The last byte will not need an ACK neither from the master nor from the slave.

Data Definition of the MXST15/16

To command the relay, the master sends out the first byte containing an address to the relay with R/W bit low to signal a write sequence. The second byte is data send out from the master; this time the data write to the relay is a command to tell the relay what to do. This second byte of data also called **register number**. The following bytes after the register-number depends on what register had been used. Some registers need only one byte of data, some registers need two or more bytes and some registers need a read command. See relay registers (commands) for explanations.

The most significant bit of the command byte (register number) is defined as a read/write bit. If this bit is low, it means the master wants to read from the relay (read command). If this bit is high, the master wants to write to the relay (write command).

GIGAVAC®
ADVANCED SWITCHING SOLUTIONS

Read command sequence:

Read commands is dome with 2 repetitions: write register number and read back data.

1. Send START condition.
2. Send relay address byte with R/W bit **LOW**(signal a write).
3. Send command byte (register number) with most significant bit **LOW**.
4. Send STOP condition to end sending. (the relay is going to send data out)
5. Send START condition again.
6. Send relay address byte with R/W bit **HIGH**(signal a read).
7. Clock out each byte from the relay. (# of byte(s) depends on register number)
8. Send stop condition to end reading.

Write command sequence:

Write commands can be done in only 1 swing.

1. Send START condition.
2. Send relay address byte with R/W bit **LOW**(signal a write).
3. Send command byte (register number) with most significant bit **HIGH**.
4. Continue to send more byte. (# of byte(s) depends on register number)
5. Send STOP condition to end writing.

Relay Registers (Commands)

Register #	Type	# of bytes	Register Name Descriptions
0	N/A	N/A	Reserved unused for now
1	R	1	Product part-number string length
2	R	See reg#1	Product part-number
3	R	2	Firmware version
4	R/W	2	Delay time on break: byte1 minutes, byte2 seconds.
5	W	1	Reset CPU (restart): 0x52‘R’
6	R/W	1	I^2C mode enable: 0x45‘E’, disable 0x44 ‘D’
7	R/W	1	(1*) I^2C address: 0x01 to 0x7E (0x00 & 0x7F reserved)
8	R/W	1	(2*) Contactor Control: 0x43 (Close), 0x4F (Open) Write: C/O. Read: C/O & D (countdown to open)
9	R/W	1	(3*) Calibration State: 0x42 (Blank), 0x46 (Filled)
10	R/W	1	Automatic power-up contactor state: open/close: 0x43 (Close) and 0x4F (Open).
11	R/W	2	(4)* Delay countdown time remains: byte1 minutes, byte2 seconds.

Note:

(1) * - I²C address is 7-bit (exclude LSB). The LSB is for R/W bit. The address takes effects immediately after the write. Power-up procedure always use address 0x7F.
(2) * - Contactor Control: Firmware version A, configure-state can't control contactor. Firmware version B has no different between configure-state and operate-state for this control command.
(3) * - This command has no effect on MXST15/16. Calibration state can only be written 0x42 to blank (clear), can't write 0x46. This byte is an indicator of calibration status, not the calibration data.
(4) * - While in delay countdown to open the contact, write 2 numbers (0-99:0-59), the delay time will use this 2 numbers to countdown. Write both numbers to zero will cancel counting and open contactor immediately. The value of this command will not save to EEP.

Different of register 4 and register 11 is that the register 4 is a value stored in the EEP and register 11 stays in RAM. When countdown starts, the value of register 4 is loaded into the countdown memory. Register 11 shows how much time left to release the contact. If you write a value to register 11 while countdown to trip, it will change the countdown time. You can read/write to register 11, but next time to countdown register 4 will always be used.

The part number (MM:SS) and the delay time (register 4) are the same when the customers first receive the products. If the register 4 (delay time on break) is changed, the part number stays the same and the delay time will reflect the register 4 contents.

Examples:

A. **Write command:** Change contactor delay-on-break-time to 5 minutes and 30 seconds using address 0x61 (0b1100001x).
   1. Send START condition.
   2. Send relay address byte with R/W bit **LOW**. 0xC2(0b11000010).Note: address 0x61 now turns into address 0xC2 because the address 0x61 is shifted left 1 bit to make room for R/W bit.
   3. Send command byte (register #4) with most significant bit **HIGH**. 0x84(0b10000100).
   4. Continue to send 1 byte value of 5 minutes 0x05 (0b00000101).
   5. Continue to send 1 byte value of 30 seconds 0x1E (0b00011110).
   6. Send STOP condition to end writing.

B. Read command: Read back the time of delay-on-break using address 0x61 (0b1100001x).

1. Send START condition.
2. Send relay address byte with R/W bit **LOW**.0xC2 (0b11000010).
3. Send command byte (register #4) with most significant bit **LOW**. 0x03 (0b00000100).
4. If you decide not to read-back, send STOP condition to end. If you don't send STOP, it will wait for the step 5 and mess-up communication sequences.
5. Send RESTART, or START condition again.
6. Send relay address byte with R/W bit **HIGH**.0xC3 (0b11000011).
7. Clock out 2 bytes from the relay for minutes and seconds.
8. Send stop condition to end reading.

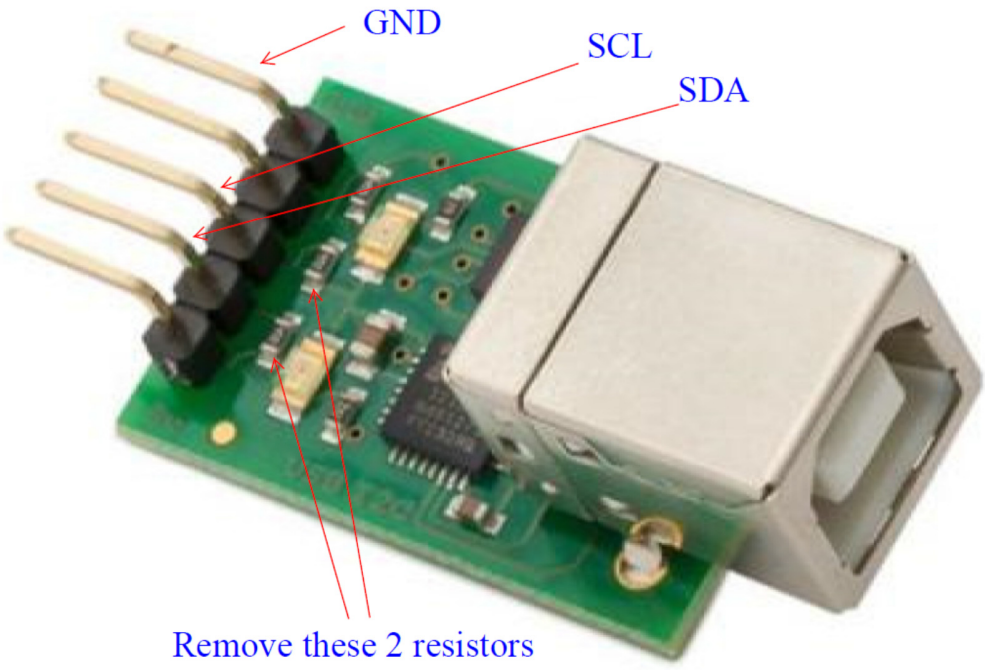
The other way to communicate with the relays is to download the application software "**MXST15/16 Control Panel**" from the GIGAVAC website.

"MXST15/16 Control Panel" Software User's Guide

To run this application program, a USB to I^2C Adapter from DEVANTECH is needed to serve as a bridge between the contactor and the computer. Here is one of the vendors has this converter:

https://acroname.com/products/DEVANTECH-USB-I2C-INTERFACE-ADAPTER?sku=R286-USB-I2C

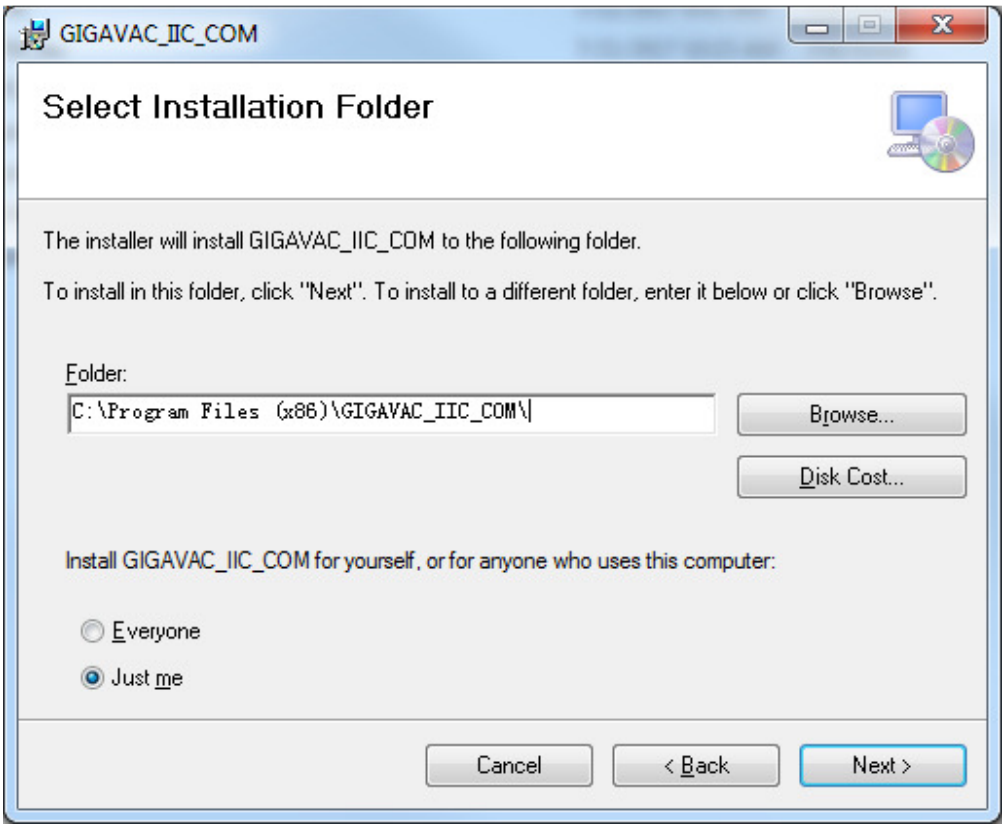
Prepare the I^2C converter to use: remove the 2 pull-up resistors:



After the converter is ready, download the converter-driver and install on your PC.

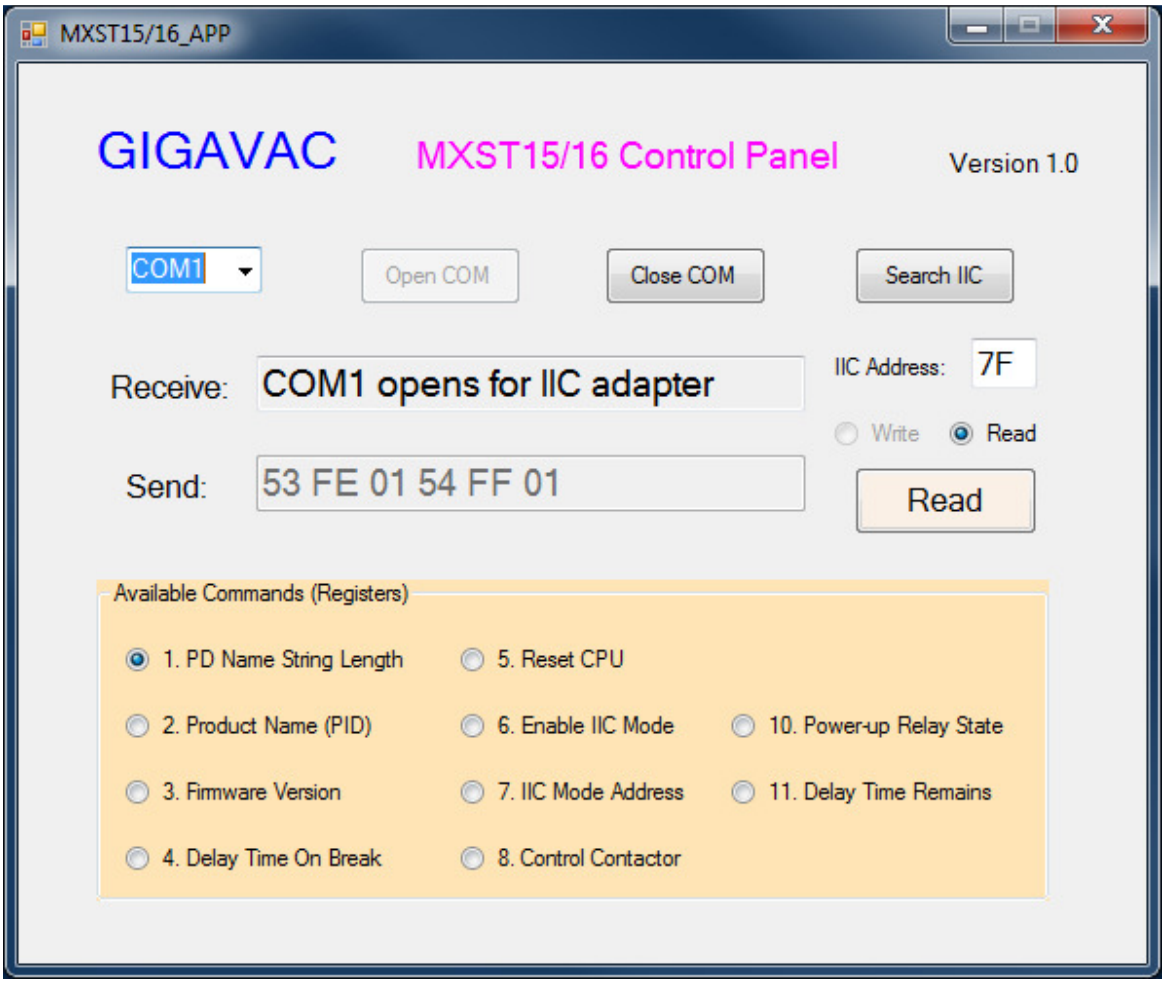
Note: remove the pull-up resistors from the adapter only for the startup procedure. If the contactor is already in I^2C mode, the resistors on the adapter will not affect communication.

After downloading the "**MXST15/16 Control Panel**" from GIGAVAC website, Install the program and choose the executable-file folder option (see picture below):



The above “Folder” textbox contents point to where the executable file will be located.

The application program user interface:



Start-up Procedure:

1. Select COM port to which the contactor is connected.
2. Click the [Search IIC] button.
3. Turn on the contactor.
4. After IIC is detected, the unit is ready to read/write.

All the buttons are self-explanatory. Try them out for yourself.

Thank you for using GIGAVAC products.

Revision History:

Rev.1: initial completion.

Rev.2: fixed the link to the adapter-vender's website.

Rev. 3: more setup and install details.