Subject: **Modbus/RTU from Pro-face HMI with Q Programming**
Applies to: ST5/10-Q, STM17/23/24Q, SWM24Q, TXM24Q, TSM11/17/23Q StepSERVO™ models (Serial versions), and SV200 series servo drives (RS-422/485 versions)
Date: October 13, 2015 (updated to include StepSERVO and SV200 servo products) (original version published July 2014)
Authors: Matt Cole, Jeff Kordik

Introduction

This exercise demonstrates the connection and control of an Applied Motion Products STM24QF-3AE integrated stepper drive by a Pro-face GP-4201T HMI. The STM24 integrated motor is coupled to a 12-inch long linear actuator. The HMI will be programmed to command simple actuator moves and to monitor the STM24 using Modbus/RTU protocol and RS-232 communication. In addition, the integrated stepper will be programmed with the Applied Motion software, Q Programmer, so that short routines including "hard-stop" homing can also be executed from the Pro-face HMI. These examples can be adapted to similar linear actuator applications that require an operator interface.

Your STM24 must have DSP firmware version 1.06 or later to support Modbus/RTU.

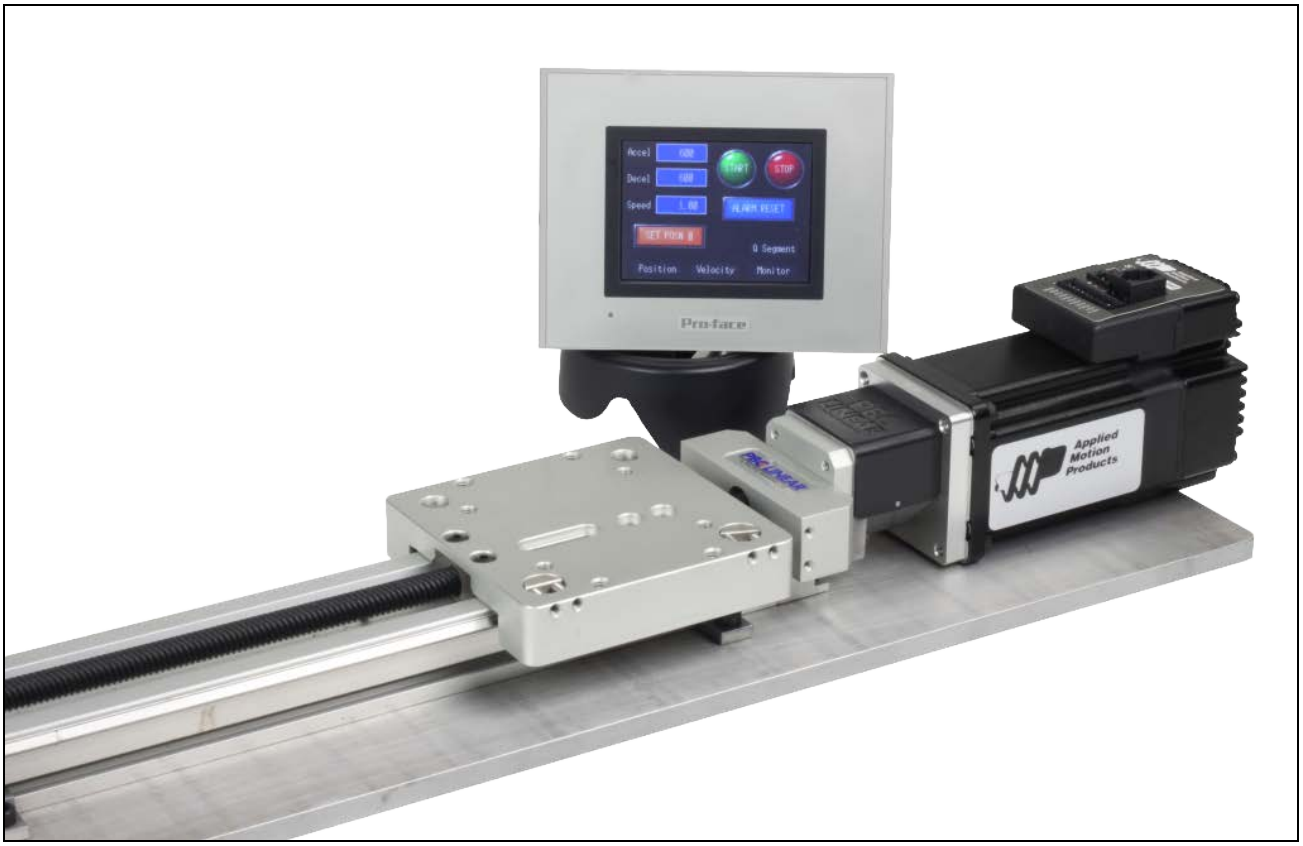


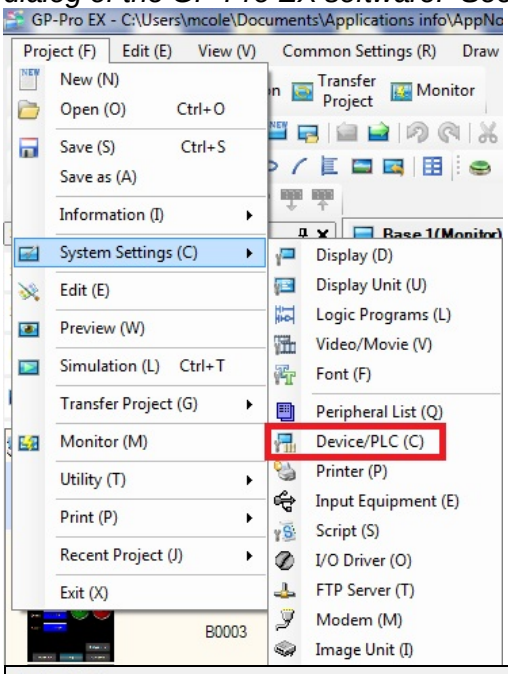
Figure 1: Linear actuator system driven by STM24QF-3AE integrated stepper using Pro-face HMI

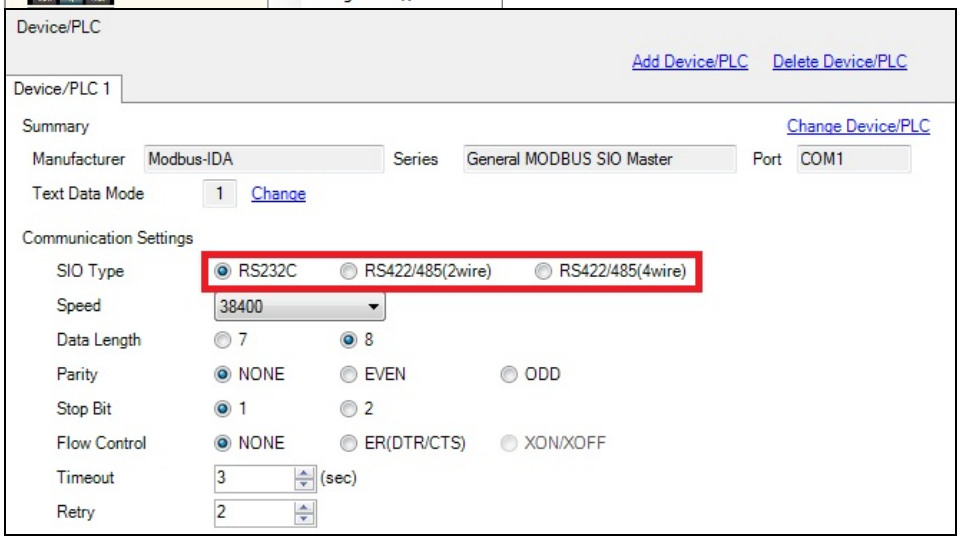
Serial Connection

Modbus/RTU can use RS-232, RS-422 or RS-485 as a physical layer. It can use any bit rate and any choice of parity and stop bits. It is the job of the user to make sure both sides are set the same and properly connected. SV200 series servo drives support Modbus RTU over RS-422 or RS-485.

This exercise uses an RS-232, three wire connection (RX, TX, GND), 38400 bps and no parity. The GP-4201T includes an RS-232 communication port with a DB-9 male connector that couples directly to the standard Applied Motion 3004-189 programming and configuration cable that ships with the STM24.

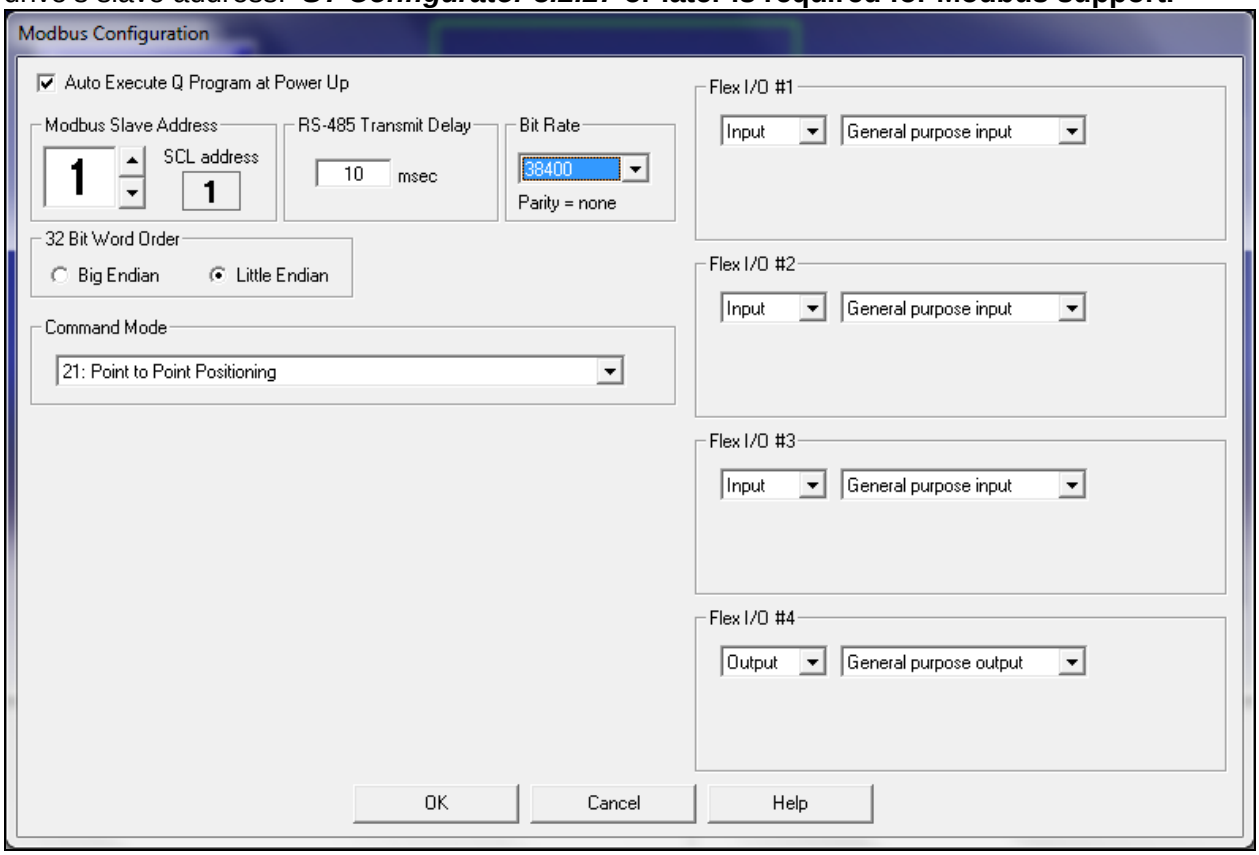
Note: *For some Pro-face HMI models, the serial comm. port may require a special adaptor to jumper pins 7 & 8 together; the RS-232 port on the GP4201T can also be converted to an RS-422 port by simply plugging in part number PFXZCBADTM1 and changing the communication settings in the Device/PLC dialog of the GP-Pro EX software. See Pro-face documentation for additional details.*





Serial Port Settings

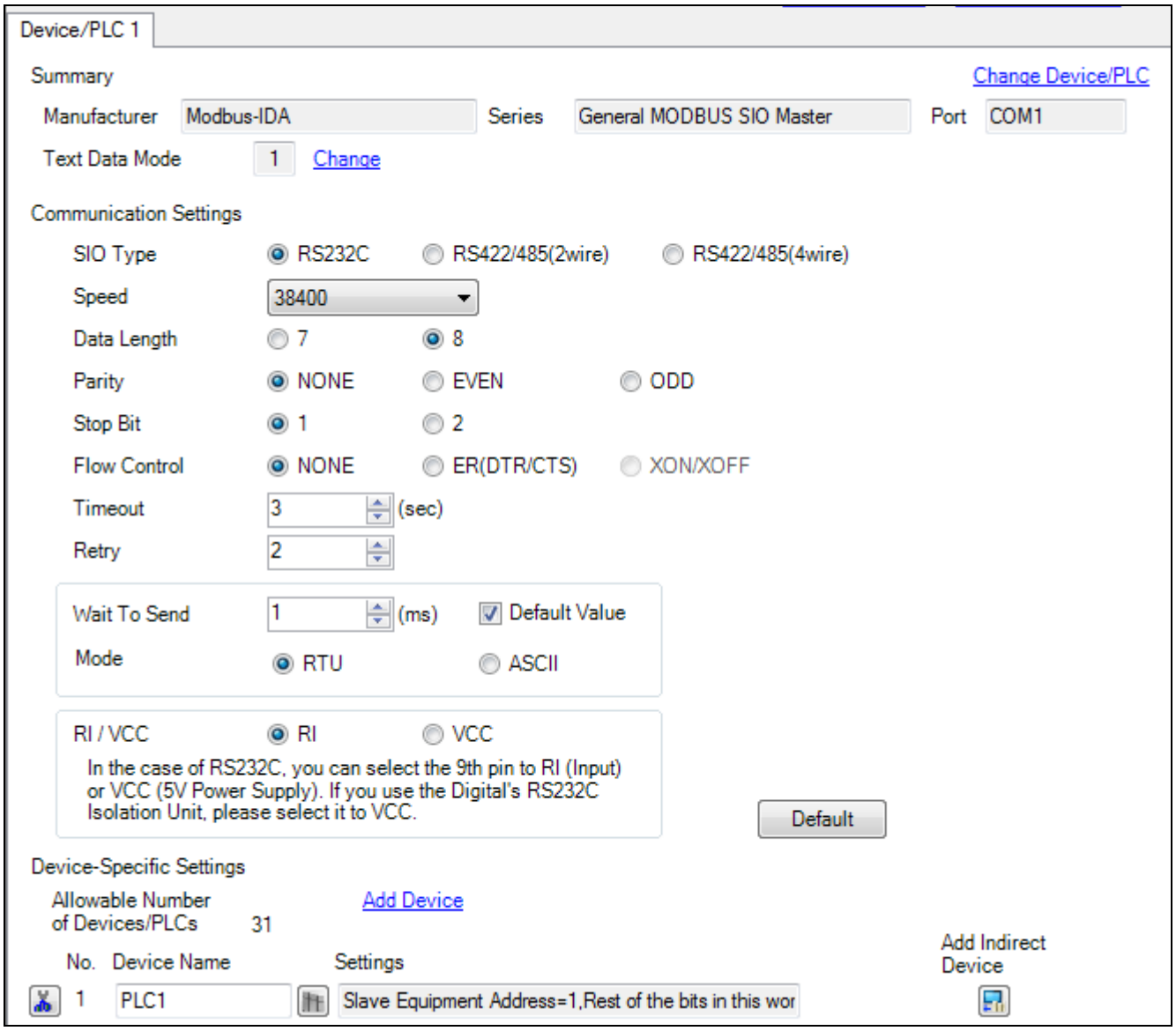
On the drive end: use *ST Configurator** to set the drive for Modbus mode, command mode 21 (point to point positioning), 38400 bps. Our drives are always set for "no parity". This is also the place to enter the drive's slave address. ***ST Configurator 3.2.27* or later is required for Modbus support.**



After downloading to the drive and closing *ST Configurator*, be sure to power cycle the STM24 so it wakes up at the correct bit rate. (The bit rate chosen for this example was arbitrary; bit rates from 9600 up to 115,200 are available.) An example *ST Configurator* file for the STM24QF-3AE is included in the ZIP file that contains this application note.

At the HMI end, the drive is connected to the GP-4201T RS-232 comm. port, leaving two USB ports and an Ethernet connection for programming the HMI from a PC. This RS-232 port is configured in the GP-Pro EX software by going to the Project Window and double clicking Device/PLC. Be sure to set the Manufacturer to "Modbus-IDA" and the Series to "General MODBUS SIO Master". The Port should be set to COM1. Be sure to set the slave address to match the STM24 setting that was entered into *ST Configurator**, in this case "1".

***NOTE:** For StepSERVO™ TSM and TXM models, the Step-Servo Quick Tuner software is required. For SV200 Series servo drives, the SVX ServoSUITE™ software is required.



Note: *If using another Pro-face HMI model, then it will be necessary to change the display model in GP-Pro EX software by going to the Display page (accessed from Project pull-down menu -> System Settings -> Display). Click on 'Change Display' on the Display page to change to another model.*

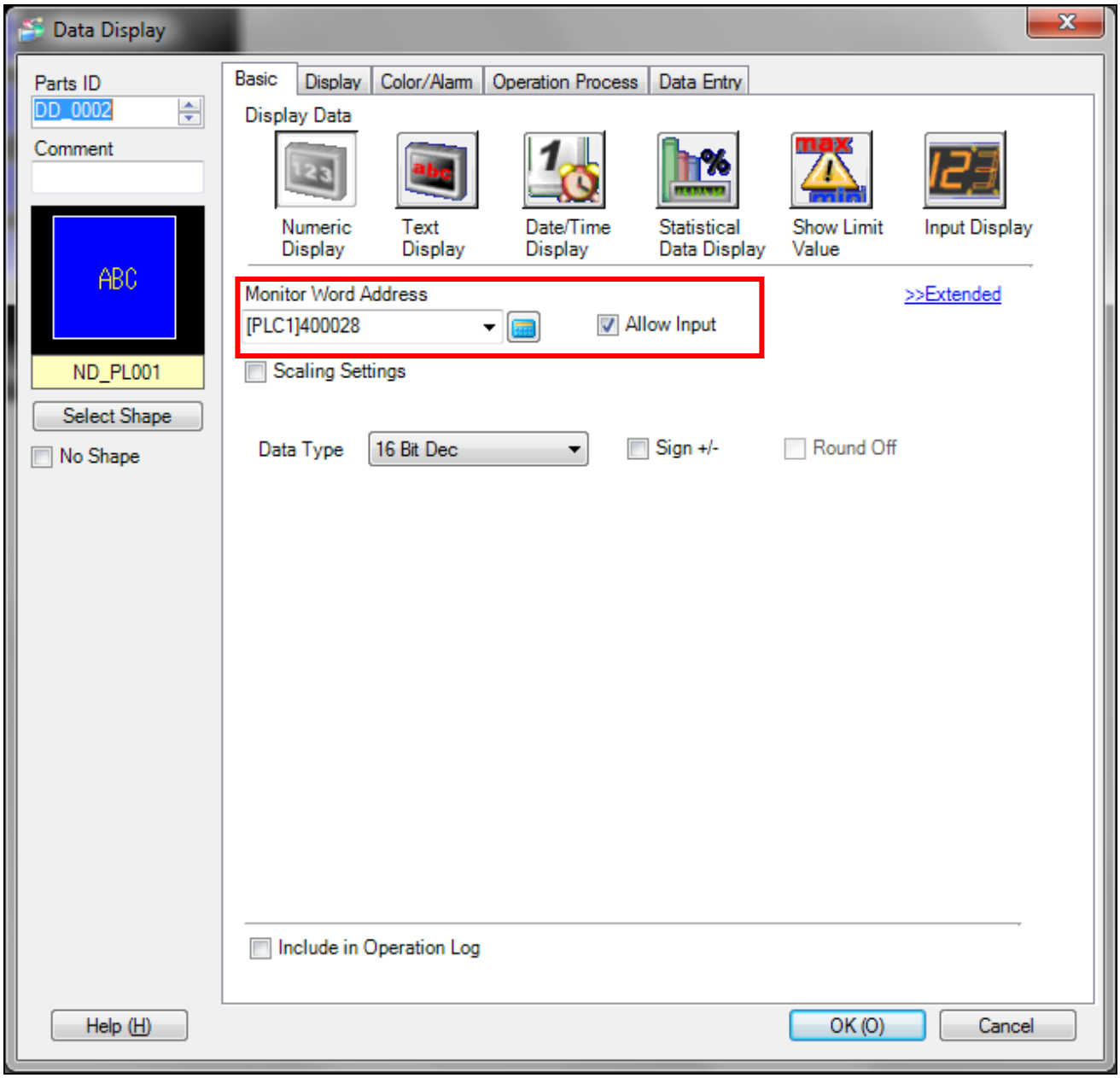
Register Mapping

The Modbus protocol is all about moving data from the memory of one device to that of another. You can move as little as one bit or you can move one or more 16 bit words. We'll be moving words, usually one at a time.

These are the STM24 Modbus registers that we'll be using in this exercise:

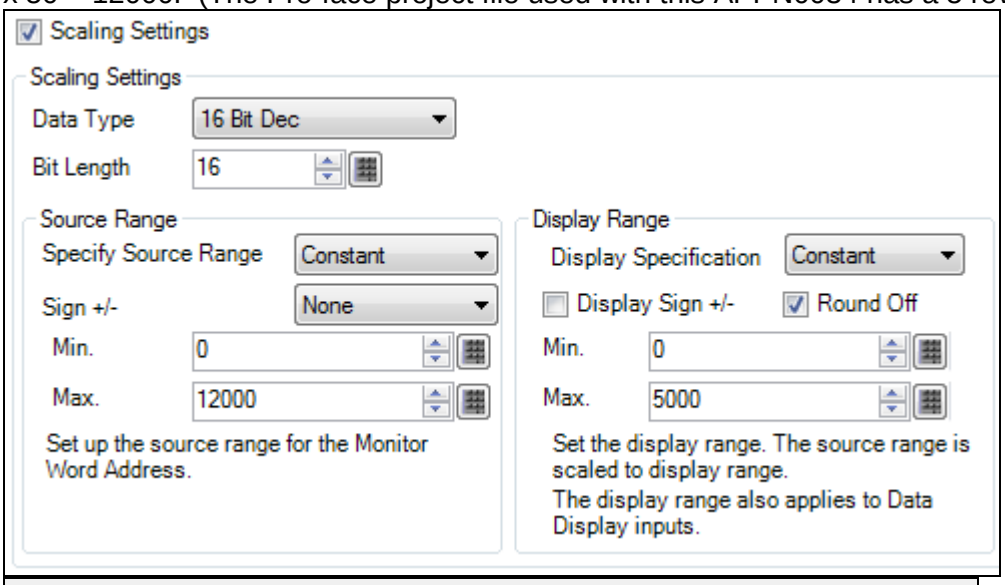
Modbus Register Number	Use
40125	command register
40126	command info 1 field (used, among other things, for specifying which Q segment to launch)
40028..40032	move parameters (AC, DE, VE and DI)
40043..40045	jog parameters (JA, JL and JS)
40001..40015	immediate registers for monitoring the drive (AL, SC, IT, IU, etc)

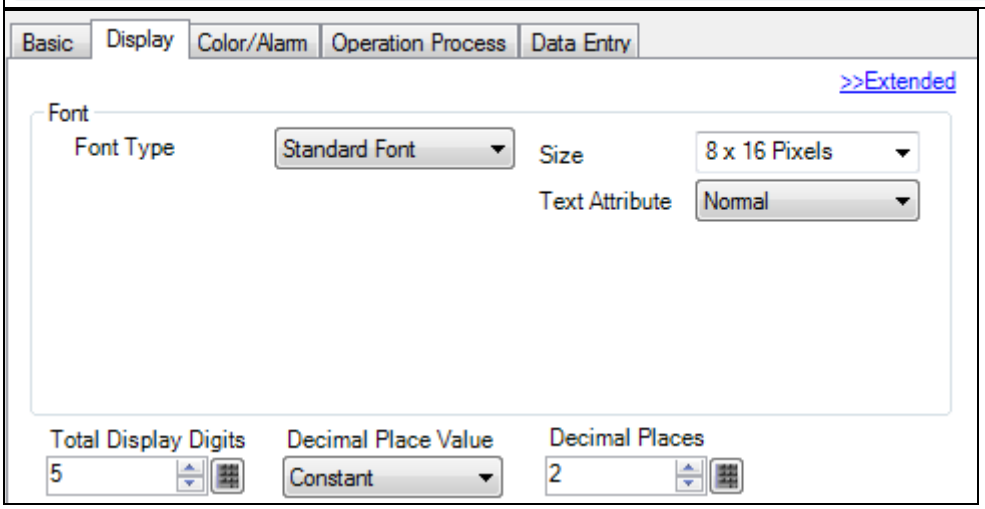
We'll need to map those to HMI objects that will display data from the registers and in some cases allow the operator to enter new data. This is straightforward, just add a data display object to an HMI screen and double click it to bring up the dialog. You set the Modbus register by clicking the blue button next to the word address. ***In the GP Pro software, Modbus addresses show an extra zero; ignore it. You'll only be entering the last three digits anyway.***



For the most part, the mapping is simple, just create a data display object and click the “Allow Input” box if you want the user to be able to change the register’s contents. If you want values displayed in more friendly units, you can select “Scaling Settings” and enter the range of values for the data register and the display. For example, the STM24 stores speed values as 0.25 RPM. Example: say we want to work in revs/sec with a range of 0 to 50 rev/sec, and two decimal places. Set the display range to 0 min, 5000 max. With the two decimal places, 5000 will appear on the HMI as 50.00 rev/sec. Don’t forget to click the Display tab and set the number of decimal places and total digits to be displayed on screen.

To complete the scaling, we'll enter the equivalent range of raw data from the register. Since the STM24 works in units of .25 RPM (1/240 rev/sec), we'll enter a source range minimum of 0 and a maximum of 240 x 50 = 12000. (The Pro-face project file used with this APPN0034 has a 5 rev/sec maximum for safety.)

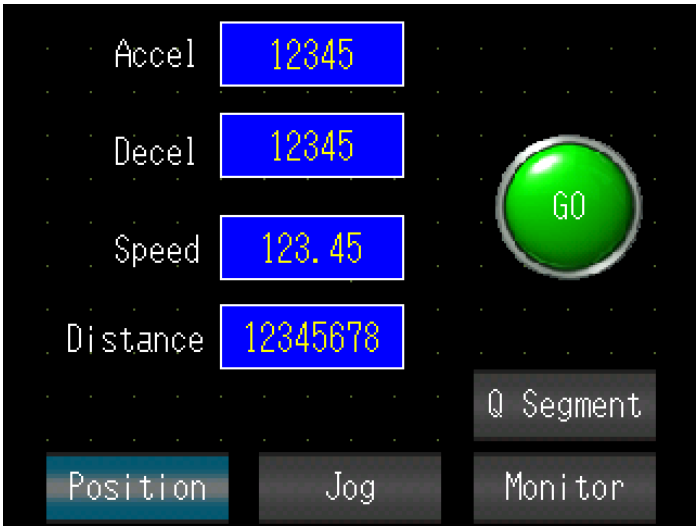


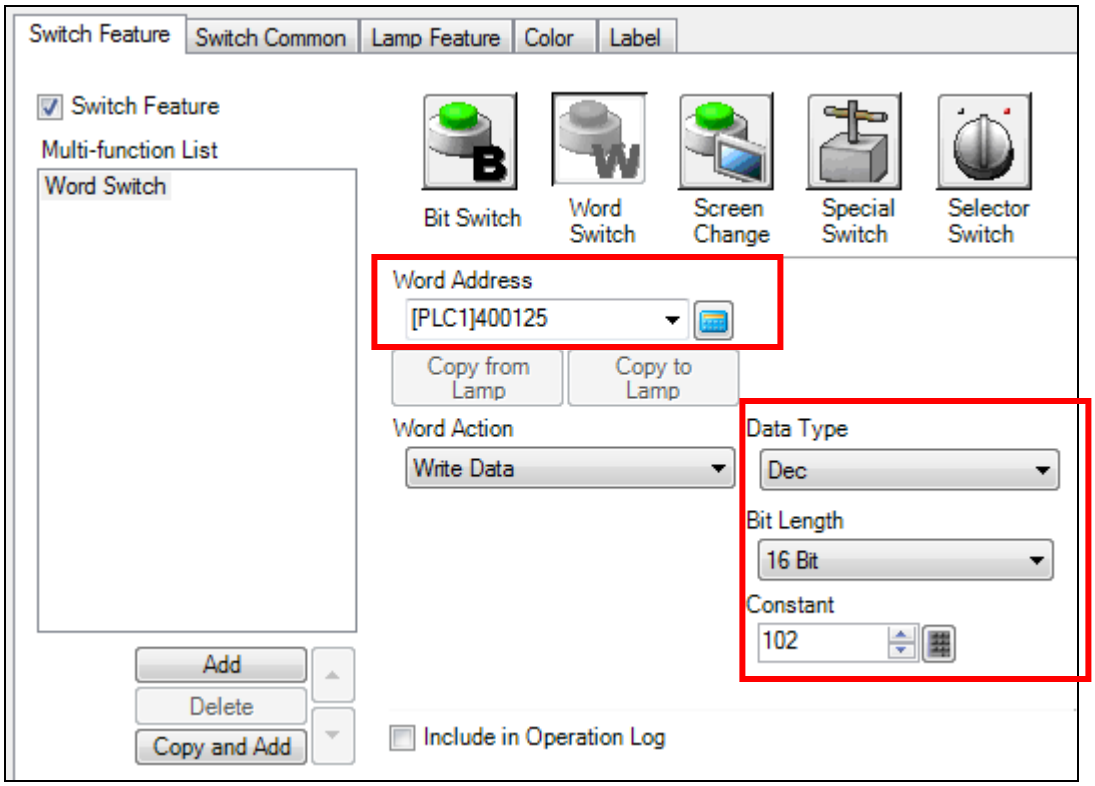


Once all the move parameters have been mapped to data display objects, it will be easy for the operator to adjust them. To execute a move, we'll need to send a specific value to the command register. We placed a pushbutton on the HMI for this, that maps to the command register (40125) and sends the fixed value 102 when pressed, which is the opcode for a point-to-point (FL) move.

During this exercise, we'll be writing opcodes to the command register to initiate various actions. This is a mere sampling of the many opcodes supported by Applied Motion Products drives. Please refer to our Modbus literature for more information.

Function	SCL equivalent	Opcode	Description
Feed to Length	FL	102	Starts a point to point move
Commence Jogging	CJ	150	Starts jogging
Stop Jogging	SJ	216	Stops jogging
Q Load & Execute	QX	120	Launches a Q segment
Stop & Kill	SK	225	Stops the Q program and halts motion





Big Endian, Little Endian

Modbus transfers 16 bit words. That's great for parameters like speed or acceleration because they are 16 bits. But move distance (DI) is 32 bits in our drives, as are some of the monitor values. Modbus is happy to move more than 16 bits of data at a time, but we need to pay attention to word order or we may be in for some unpleasant surprises.

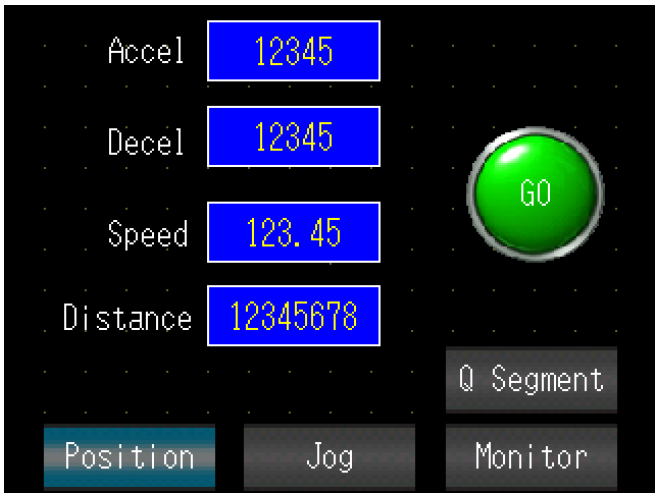
In our Modbus implementation, we default to storing the big end of 32 bit values in the first word of memory. That's called big endian. Consider, for example, setting DI for 100,000. That's 000186A0 hex, which is stored as two 16 bit words: 0001 and 86A0. The big end of the word (the most significant word, or MSW) is 0001 and it goes into the first register location, 40031. The little (least significant) end is 86A0 and that goes into the second word, 40032.

Great, but what if the HMI has other ideas? In fact, the GP4201 uses little endian word order for 32 bit values, so if I write 100,000 to a memory location, it will write the little end (LSW) first and the drive will see it as 86A00001. Not good. 86A00001 hex equals 2,258,632,705 decimal. That's a very long move.

To keep things simple for the PLC or HMI programmer, our drives have a switch that allows them to use little endian word order. Just select "little endian" when configuring the drive with the *ST Configurator* software. If your PLC/HMI needs big endian, select "big endian".

Point-to-point Move

Building on what we've already accomplished, let's program the HMI to initiate a point to point (fixed distance) move. We've constructed a screen with four numerical entry objects for entering move distance, speed, acceleration and deceleration. We'll also add a pushbutton to start the move when pressed.



The four numeric entry boxes are mapped to the following Modbus registers:

Function	Modbus register	Data type
accel	40028	16 bit unsigned
decel	40029	16 bit unsigned
speed	40030	16 bit unsigned
Distance	40031	32 bit unsigned

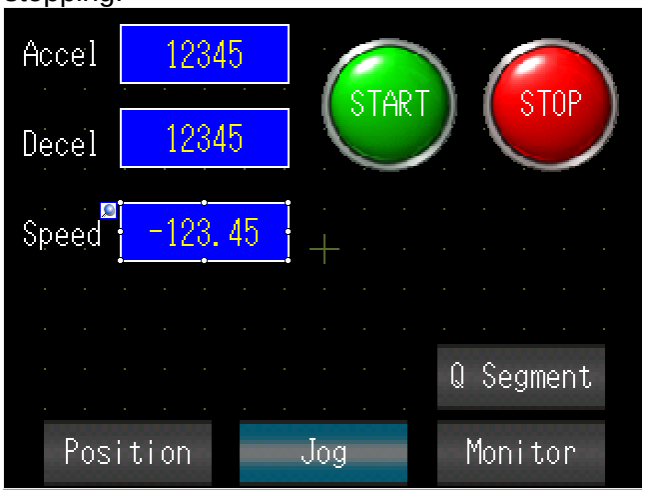
The GO button, when clicked, sends the opcode for a feed to length move (102 decimal) to the Modbus command register 40125, which starts the move.

Jogging the Actuator (Velocity Move)
We've created another screen in the HMI for velocity mode and called it "JOG". This time we have three numeric entry boxes that connect to Modbus registers in the drive:

Function	Modbus register	Data type
Jog accel	40047	16 bit unsigned
Jog decel	40048	16 bit unsigned
Jog speed	40049	16 bit signed

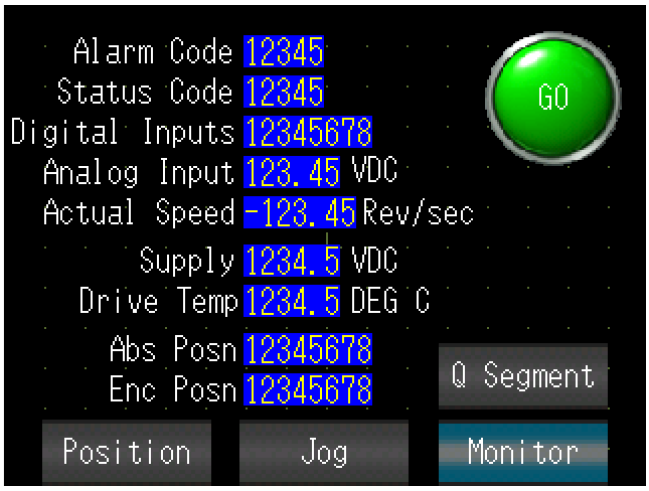
The velocity has been scaled into revs/second, assigned two decimal places and allowed a range of +/-5 rev/sec. That's done by setting to source range to 1200 max and -1200 min. The display range is 500 max and -500 min. The scaling is a bit counterintuitive: our internal unit of speed is rev/sec*240. To achieve a range of +/-5 rev/sec, the source range must be set to +/-5*240 = +/-1200. If we were working in whole numbers (1 rev/sec, 2 rev/sec, etc) then we'd set the display range to +/-5. To get two decimal places, we must use a display range of +/-500 which will show up as +/-5.00.

Also present are pushbuttons for starting and stopping the move. These send the proper opcodes to the command register (40125) for starting and stopping a jog move: 150 for starting motion and 216 for stopping.



Monitor the Drive on the HMI

To monitor drive status, we've created an HMI screen with ten numeric display objects mapped to the STM24's Modbus registers so the user can observe drive status, motor speed, encoder position and much more. The monitor screen also includes a GO button so that the operator can observe the monitor data while a point-to-point move is taking place.



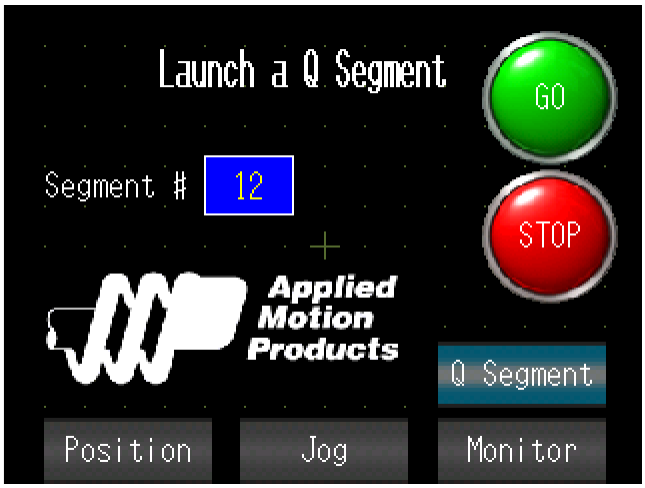
These are the ten Modbus registers connected to the monitor:

Function	Modbus register	Data type	Source Range	Display Range	Dec Places
Alarm code	40001	16 bit hex	-	-	0
Status code	40002	16 bit hex	-	-	0
Digital inputs	40004	8 bit binary	-	-	0
Analog Input	40017	16 bit decimal	0..16384	0..500	2
Actual Speed	40011	16 bit decimal	-12000..12000	-5000..5000	2
Supply	40014	16 bit decimal	-	-	1
Drive Temp	40013	16 bit decimal	-	-	1
Abs Position	40007	32 bit hex	-	-	0
Enc Position	40005	32 bit hex	-	-	0

Launching a Q Segment

One of the strengths of the Applied Motion Modbus implementation is distributed intelligence. You can create and store up to 12 Q segments in the drive and launch them from the HMI. The Q segments can then operate the motor, interact with I/O, and make decisions on their own.

The segment number connects to Modbus register 40126. The segment is loaded and executed by clicking the GO button, which writes 120 to the Modbus command register (40125). To demonstrate the ability for the HMI to stop a Q segment, there is a STOP button that sends the opcode 225 to the command register. This halts the Q segment and stops any motion.



A sample Q program file is available with this application note on the Applied Motion website. There are 12 segments contained in this Q program, each containing sample routines for an actuator with 12-inch stroke. These move routines are described in the table below.

Segment Number	Segment Title	Description
Segment 1	Initialization and Hard-stop Homing	Sets initial parameters, including Stall Detection, then sends the actuator to the fully retracted position for hard-stop homing
Segment 2	Absolute Move and Home	Moves actuator to location corresponding to 40,000 motor steps and then sends it back to home position
Segment 3	Midway and Home	Moves actuator to the middle of its stroke length, returns home
Segment 4	Fast Out and Back	Moves actuator to extended position, then 30k steps beyond home
Segment 5	Out and indexing back	Moves actuator to extended position, then indexes it back home
Segment 6	Retract index out Home	Moves actuator home, indexes it out, then home again
Segment 7	Midway Out and Back with soft accel	Moves actuator to mid-stroke, then fully extended, then home with lowered acceleration and deceleration rates
Segment 8	Fast Out and Back	Increases motor current for fast extension and retract moves
Segment 9	Fast midway-Home	Increases motor current for fast mid-stroke and retract moves
Segment 10	Oscillate near Home	Sets repeat counter for short moves between 40k steps and home
Segment 11	Full Range - cycling	Sets repeat counter for full-stroke moves at medium and fast speed
Segment 12	Fault Recovery and Set Home	Resets alarm created by hard-stop homing move in segment 1, then re-enables the motor, sets home position

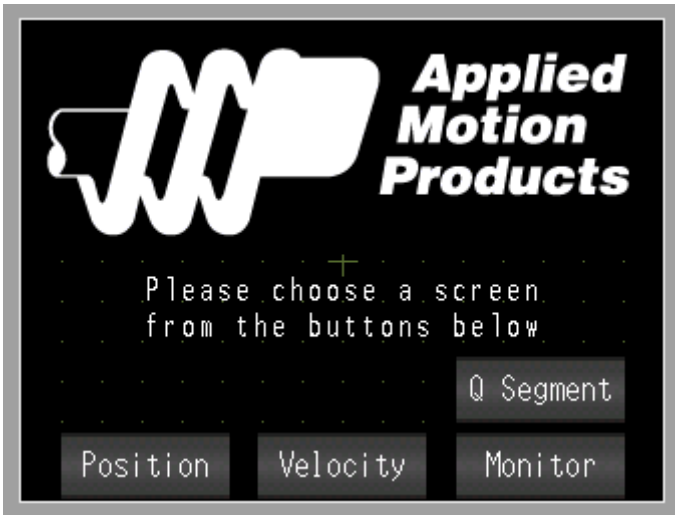
With the Q segments described in the table above, it is easy to see at a glance what can be expected when running a 12-inch or longer actuator having approximately a 0.38-inch per turn lead.

Important Notes about the Q Program and HMI Functionality
Applied Motion Products assumes no liability for the use of the sample Q program segments in conjunction with an actuator. The Q program that is provided with the application note should be used with caution. While steps were taken to avoid damaging the actuator, it may be possible to cause the motor to "lose" its home position and subsequently run a Q segment that could slam the actuator carriage into an end stop. Segment 1 is set to execute automatically when the stepper is powered on. It uses the OF (On Fault) command, which will react to the stall detection event by loading and executing segment 12. Segment 12 will, in turn, clear the fault created by the hard-stop homing stall, then move the motor and set the "Home" position. If segment 12 is run again, the home position will be changed. This is important to understand that segments 2 – 11 depend on the initial home position setting and use absolute Feed to Position (FP) commands.

The additional HMI functionality could also lead to crashing the carriage of an actuator into an end stop. The speed range for the JOG and POSITION screens has been limited to 5 rps to prevent high speed collisions.

Additional steps can be taken to protect your hardware. For example, consider adding limit sensors and using the built in functionality of the Applied Motion hardware and software to configure these limit sensors for use. Future developments are planned to implement software limits, but as of the date this was written, this is not yet available.

Finally, we added an intro screen with buttons to take us to the screen of our choice. This screen is shown at power up.



Complete project files for the Proface GP-Pro EX software, as well as hardware manuals and our Modbus Manual are available at www.applied-motion.com.