PRODUCT FAMILY: Sure Servo

Number: AN-SERV-011

Subject: Sureservo position with PAC

Date issued: Jul-30-2010

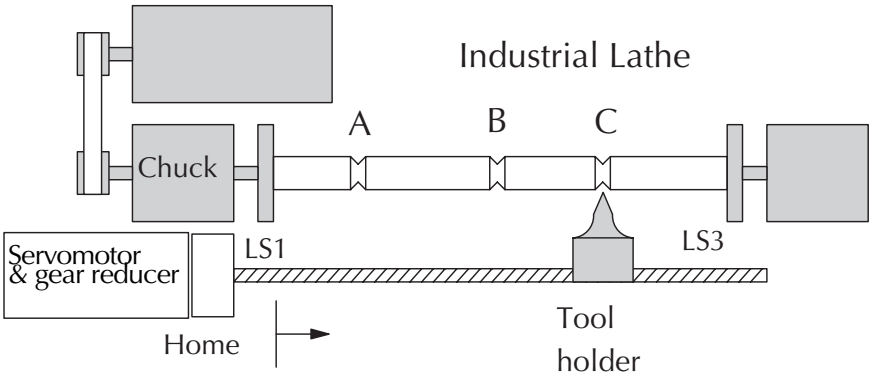
Revision: 3rd Edition -01-12

On this example we will control the position of the tool holder on an industrial lathe to execute 3 grooves at certain distance from the home position on the material of the spindle by means of programmable position targets on the *Sure*servo drive, with variable distance defined by the operator with a touch panel C-more model EA7-T10C as well as by means of MODBUS communication from a P3-550 CPU.

This document assumes that you already have a working knowledge with programming the PAC as well as the C-more touch screen.

For this example we have determined the kinematics of the movement, the sizing of the servo motor and design considerations on the application note AN-SERV-001. This note is an upgrade of the referenced document. We will wire the servo drive to the PAC. We will show the wiring of the drive, the programming of the PAC, the touch screen panel and the servo drive.

See the following diagram to explain the concept.



LS1 and LS3 are the overtravel limit switches

The normal position of the tool holder or carriage is the home position. It may happen that the system gets powered off while the tool holder is in an intermediate position. The PAC detects this fact and, while starting, the tool holder will move to Home.

The part installed on the spindle of the lathe will receive 3 grooves that will be located on positions A, B and C, freely defined in distance from a Home position.

Recalling basic data from the application note AN-SERV-001, the pitch of the lead screw is 5 revolutions per inch (or a lead of 0.20 inch/revolution). The lead screw cannot run more the 500 rpm, per manufacturer limitations. Gear reducer ratio is 7:1.

The lead screw that displaces the tool is 108 inches long and one of the sides is coupled to the servo motor, thru a gear-reducer.

The operation will be as follows:

At the start of the job, the servo motor brake is applied to hold the shaft; when the **Start** button on the touch panel is touched, the brake is released and the lead screw will position the tool holder at Home, if not in **Home** position. After that, the system will move the tool holder to position A; the spindle will begin to rotate; then the tool will advance while the spindle rotates until the groove is done; The spindle will continue to rotate; the tool will return to retracted position and the spindle stops rotating.

Position sensors will detect the position of the tool holder and will not allow the servo to move unless the tooling is retracted. The servo's brake is applied during this time. Using the same procedure, the tool holder will repeat the sequence at position B and Position C.

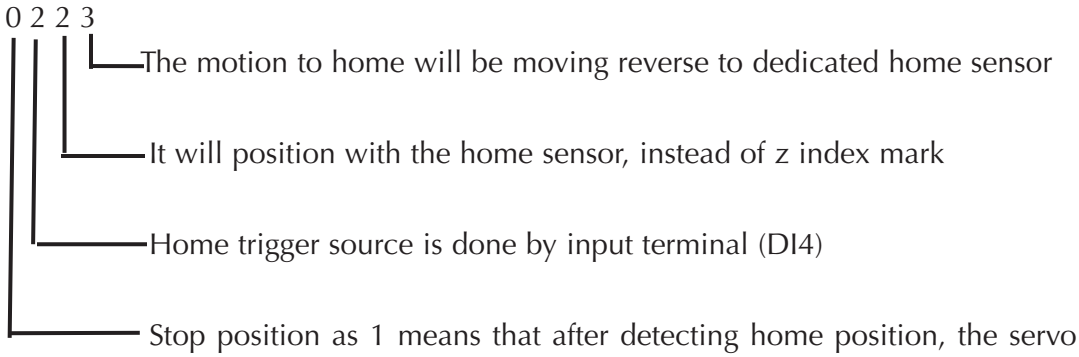
At the end of the cycle, the tool holder will return to the home position, to wait for next cycle. Positions A, B and C are defined as distance in inches from the home position, with 3 decimals of precision.

There is a proximity sensor to indicate the Home position to the drive.

Servomotor position concept

The most important concept here is that the servo parameter P1-01 is set as 101, which defines internal register position with forward rotation.

The home position is set with the parameter P1-47 and will do the following:



The positioning with internal register may have absolute or incremental mode.

We have elected to use absolute mode. The use of a home routine will help to start always from the same point.

An offset may be defined in parameter P1-51 to avoid that the sensor gets activated while in home position.

When the servo is in the condition of **Home completed**, the current counts of the revolutions and the fractions of a revolutions will be set to zero by the clear command, if parameter **P2-50** is 1. This will be handled by the PAC, without intervention of the operator.

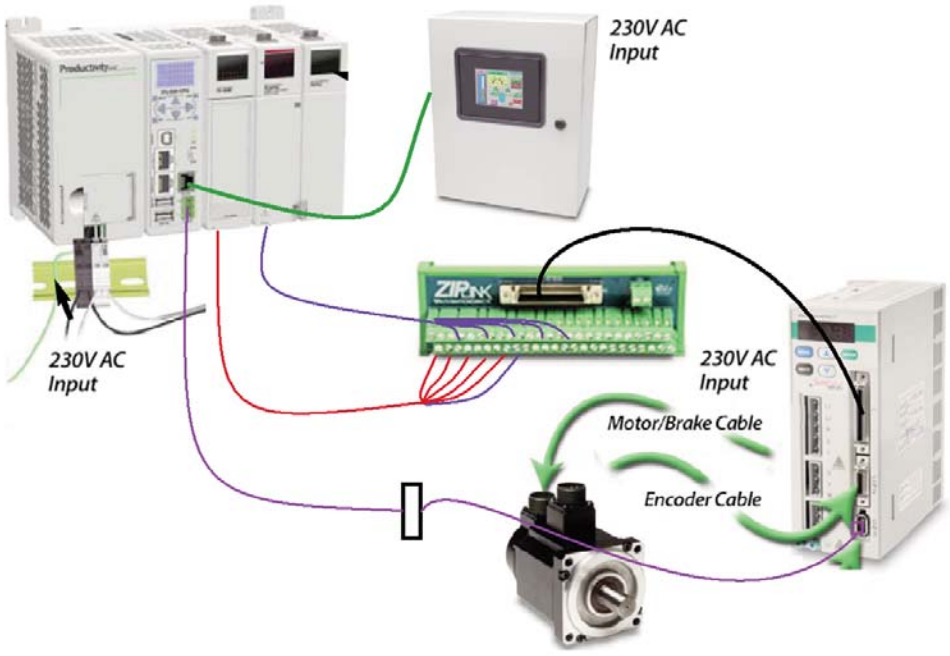
Hardware selection

This section of the document explains how to select the modules of the PAC, the C-more panel accessories and the servo hardware.

We need:

- A PAC base for 230 Volt incoming power,
- A CPU P3-550,
- An output module, for example, a P3-16TR
- The touch panel EA7-T10C
- A power adapter EA-AC, connected to the 230 Volt supply, to power the C-more panel.
- A cable to connect the PAC to the C-more panel, in this case a crossover cat 5 cable
- The servo motor
- The servo drive
- The breakout board part ZL-RTB50 to make the wiring from the sensors to the servo and PAC with the cable ZL-SVC-CBL50-1.
- The cables to link the PAC to the servo drive for communications. One of them might be the cable SVC-MDCOM-CBL. The other may be a Belden cable.
- A relay for brake control and the 24 VDC power supply
- Proximity sensors to determine the overtravel limits and the home position as well as in the tool holder,.
- The necessary hardware for electrically protecting the PAC, the C-more panel and the servo drive.

The layout of the components for the control system will be similar to this sketch:



Touch panel and PAC I/O definition

The next step will be to define the touch panel and PAC command control.

The touch panel has objects to execute different functions in 2 screens.

The objects have associated one tag to execute each one of the operations. The definition of the position is done with screen 2. This screen can be reached from screen 1, the default screen, with a button to change screens.

The touch screen panel will have screen 2 to define the distance for every one of the positions, directly in inches.

The operator should enter the desired positions A, B and C before the start signal is given on screen 2. When the button "PRESS TO TRANSFER DATA" is touched, the data is copied or moved from PAC to the servo drive. See page 7 for more details.

On screen 1 of the touch panel the lathe operator will have available 2 pushbuttons to give the command to START the cycle and the other, STOP, to return to Home, in case there is a power shutdown during the operation or just to stop the cycle.

The operator will press the pushbutton START and the PAC will indicate to the servo to move the tool holder to Home, if still not in Home. The home sensor will be adequately located close to the lead screw and will be wired directly to the servo drive.

When the tool holder is at home, the PAC will define to select the current position as position A and the values will be sent to P1-15 and P1-16 thru MODBUS..

When this condition is reached, the PAC gives the command trigger to move to position A. The servo brake is released.

At a certain point after the servo shaft rotates, the position will be reached, the servo shaft stops and the servo will give a confirmation "at position"with the digital output D02, that corresponds to a PAC input read by MODBUS.

Next the tool will move to execute the groove on the piece of the lathe.

The spindle rotates.

When the tool retracts, a sensor adequately located tells the PAC that the groove is done and the next movement could be triggered.

In the same way, position B and C are reached.

At this time the tool holder will do the same operation. Of course, for position B, the desired position should be sent to the servo before the motion. For position C, the corresponding desired position should be sent to the servo before the motion.

At the end of the cycle, the tool holder will move back to the Home position and will turn off the cycle to make the 3 grooves.

Of course, more than 3 grooves can be programmed, if so desired. This is only a simple example.

On the PAC, we will select the following functions and the corresponding outputs:

The start command to initiate the movement is given thru the touch panel with the object button START on screen 1. The STOP command is also given thru the touch panel with the button object STOP on screen 1.

The tag on DI-0.1.1.1 is the **Tool in retracted position,** wired directly to the PAC from a sensor located in the tool holder. The tag **At position is** generated by the servo drive output D02 coming from the servo drive outputs; the **Home completed,** is generated by the servo drive output D03.

The servo enable command will have the command **Servo Enable** and go to the servo input DI3 thru wires coming from the output DO-0.1.2.2.

Other signal from the PAC will have the signal **Home Search** and go to the servo input DI4 thru wires coming from the output DO-0.1.2.3..

The motion will be started with the command **Trigger to Start** and go to the servo input DI5 thru wires coming from the output DO-0.1.2.4.

The **FWD overtravel** and the **REVERSE overtravel** limit switches are wired directly to the drive in the digital inputs DI6 and DI7, as well as the home sensor in DI2.

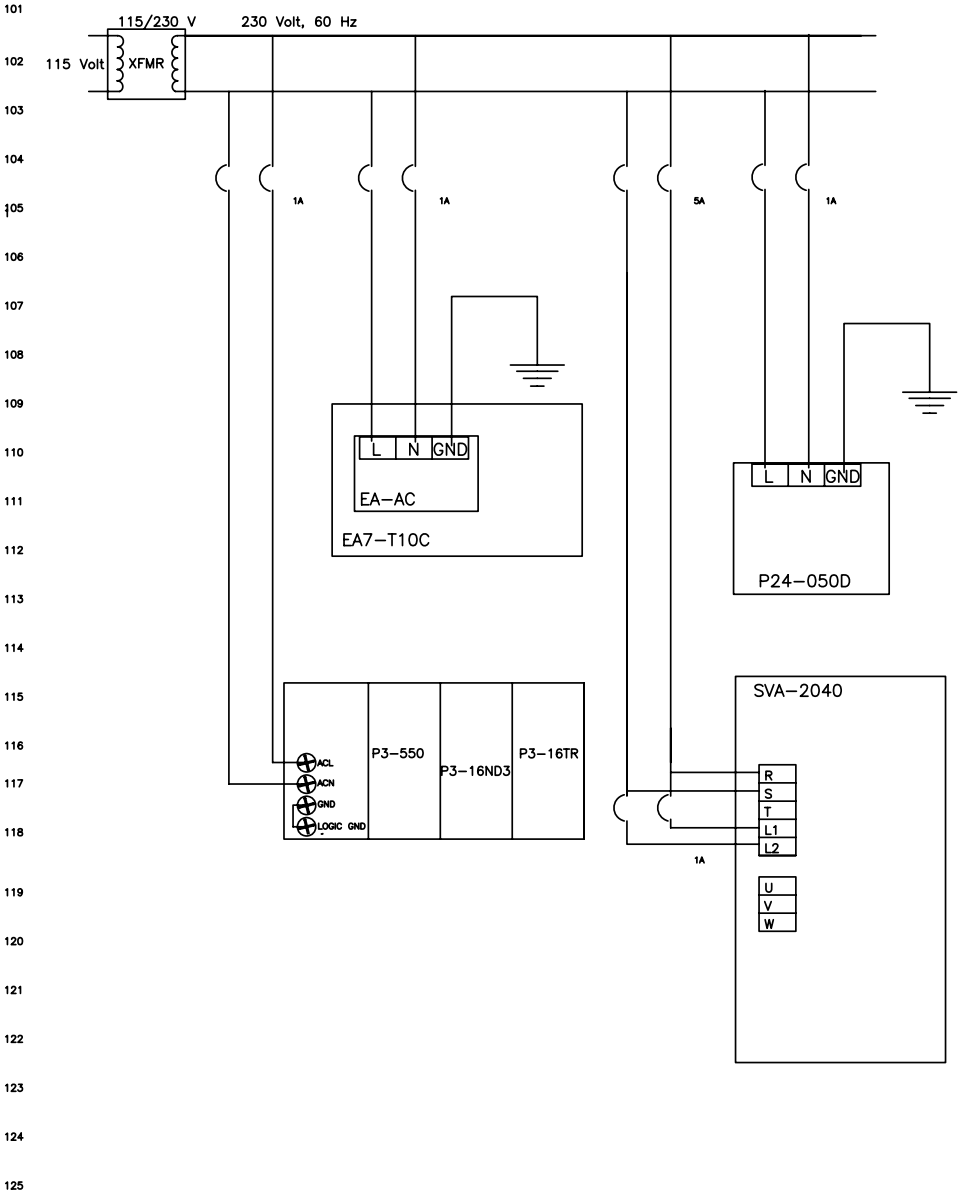
The servomotor brake coil is wired thru a relay and the command will come from one of the servo outputs.

DO-0.1.2.5 is the output for the command to start the spindle rotation and is related to the tool start (To begin the groove) and is connected to other parts of the system, not shown here.

We show below a partial summary of the inputs and outputs on the PAC.

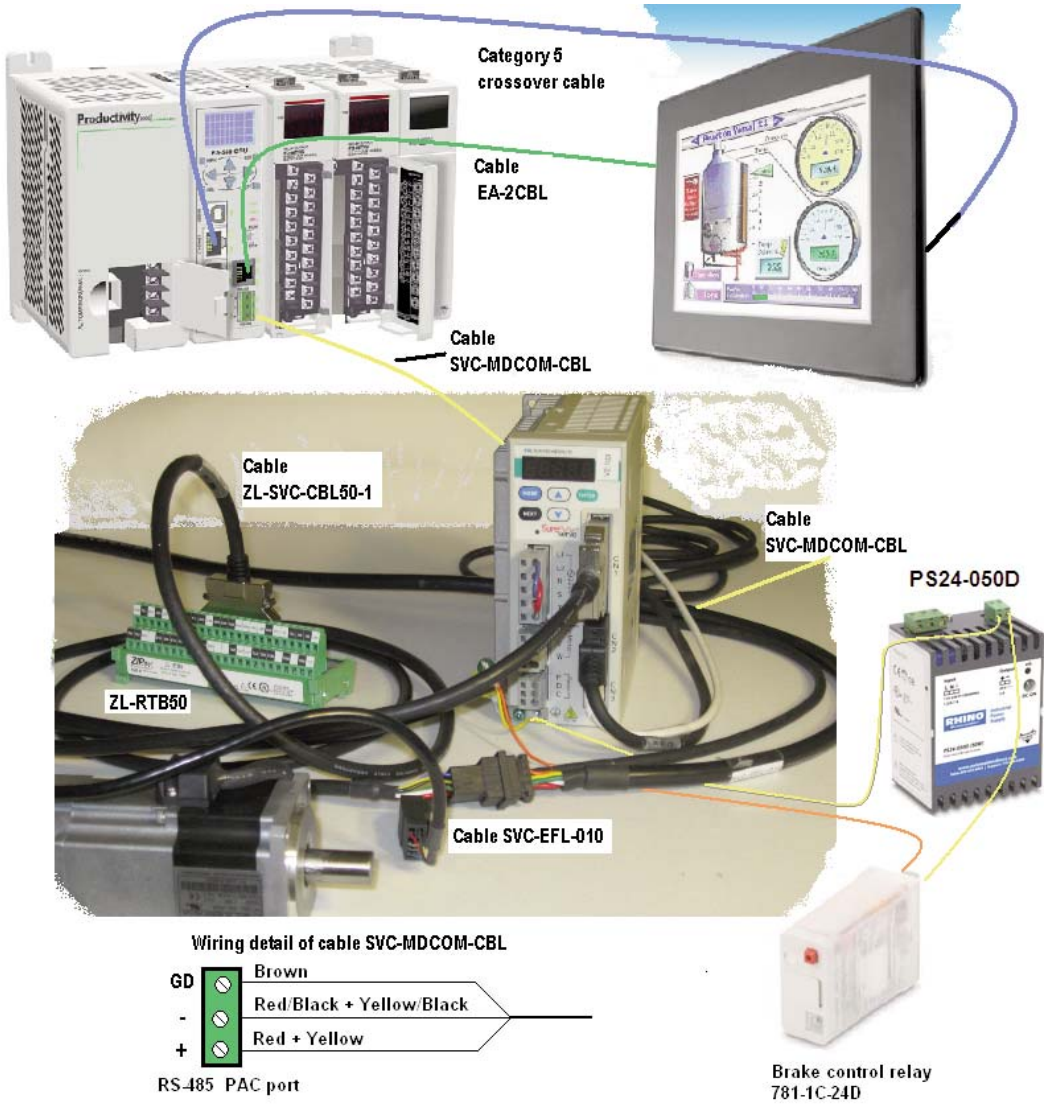
Tag	**Definition**	**Comments**
DI-0.1.1.1	Tool retracted	Proxi sensor located on the tool holder
Home sensor	Home sensor	Proxi sensor located on the lathe
Bit 3 P4-09	Home completed	Signal coming from the servo
Bit 2P4-09	Position completed	Signal coming from the servo
Trigger	Trigger to start	Signal goes to servo
DO-0,.1.2.2	Servo Enable	Signal goes to servo input DI3
DO-0.1.2.3	Search home	Signal goes to servo input DI4
DO-0.1.2.5	Run spindle	The lathe spindle will turn when activated and will activate the tool holder motion

The power wiring among all the pieces is shown on the diagram below:



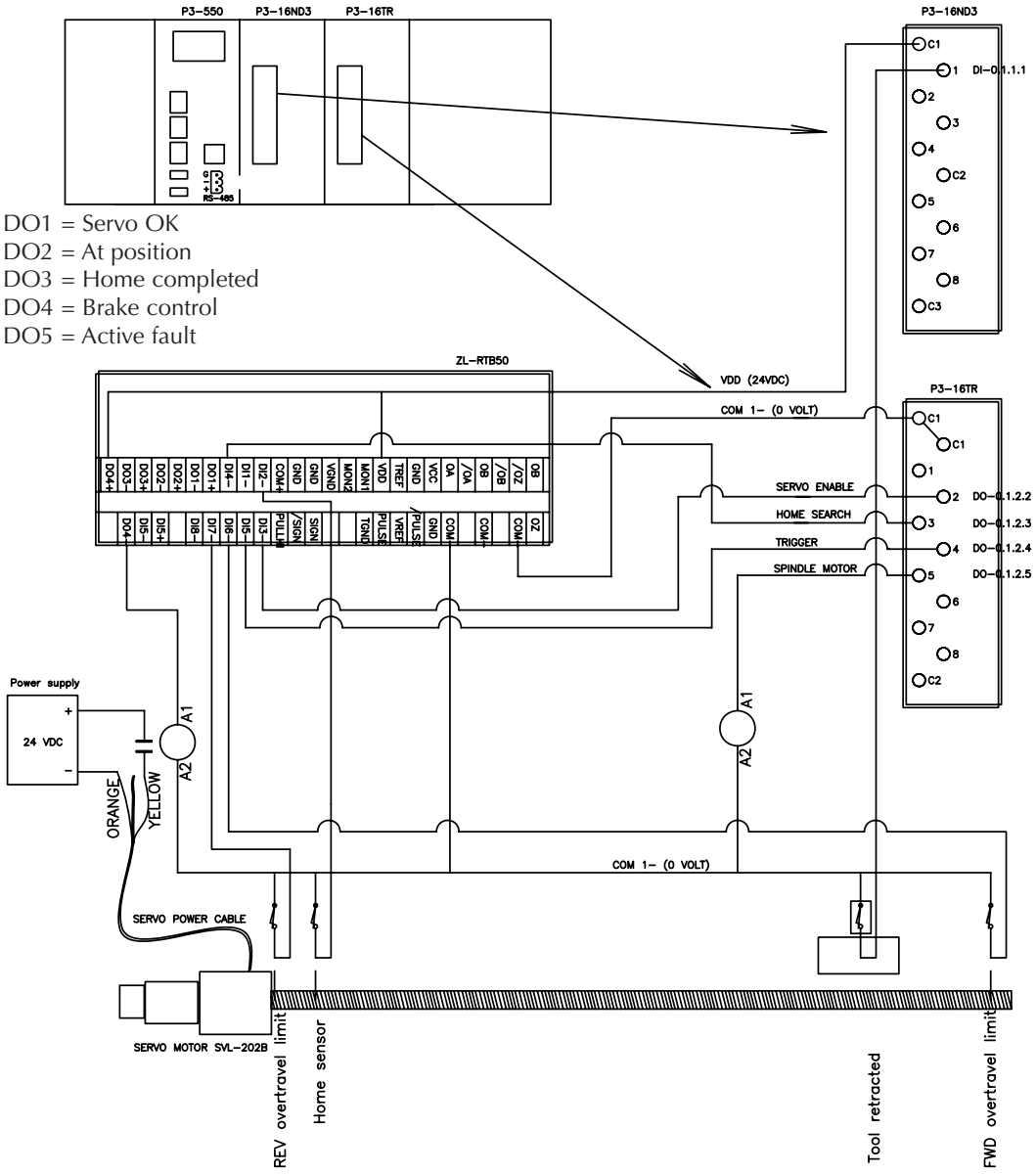
In the prototype used for testing the program was a supply of single phase 240 Volt, 60 Hz obtained with a control transformer of 1 kVA to go from 115 V from a receptacle.

The control wiring overview is shown on the diagram below:



The detail of the control wiring is shown on the following diagram

Wiring between PAC and servo drive



On the PAC and the C-more panel we have selected the following functions and the corresponding output:

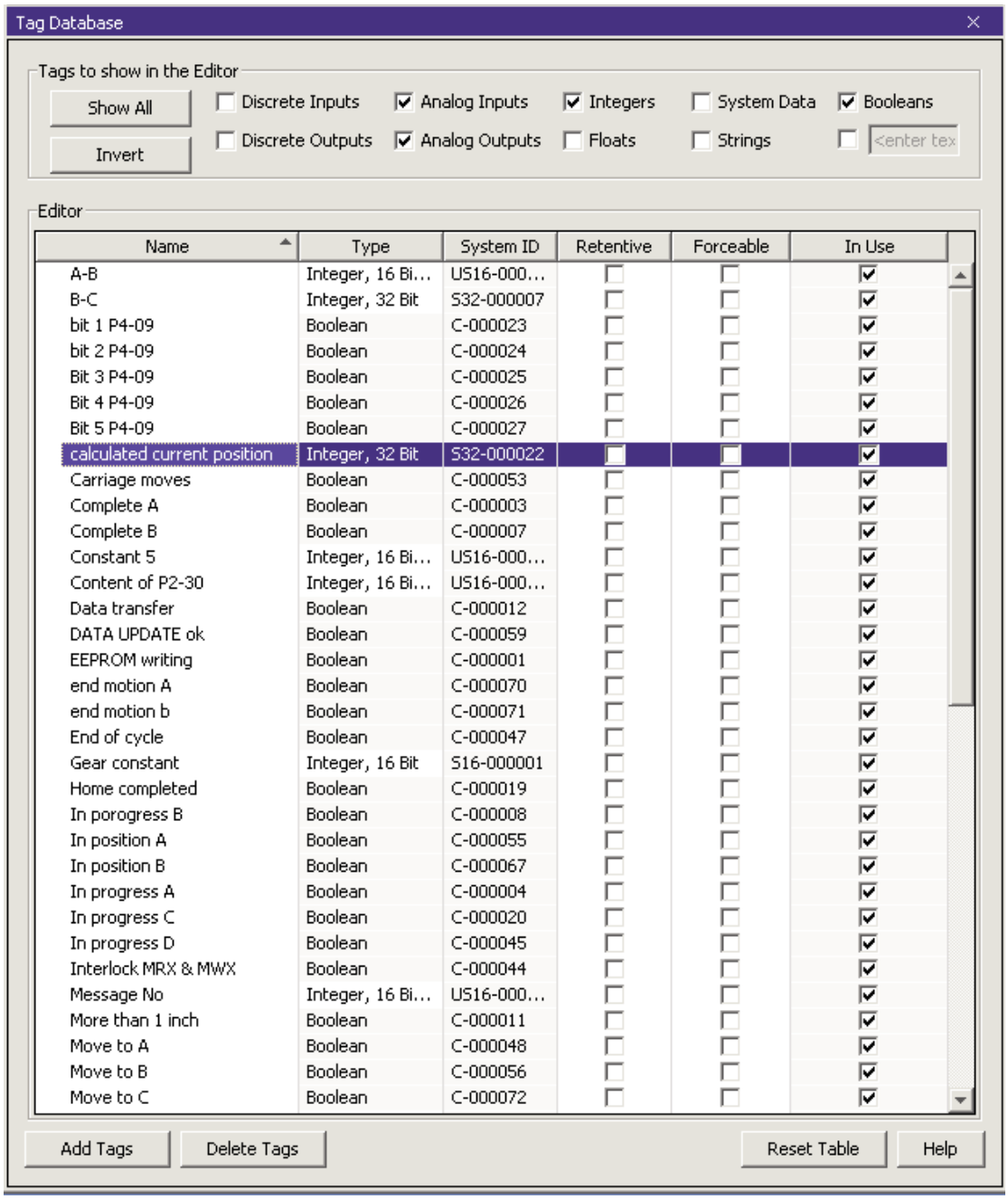
START LATHE is the start command to initiate the movement. This is generated by the operator on the touch panel.. The PAC input DI-0.1.1.1 is the **Tool in retracted position**. We have changed the tag to TOOL RETRACTED with the help of the Tag database. The **FWD overtravel and** the **REVERSE overtravel** limit switches are wired directly to the drive, as well as the **home sensor**.

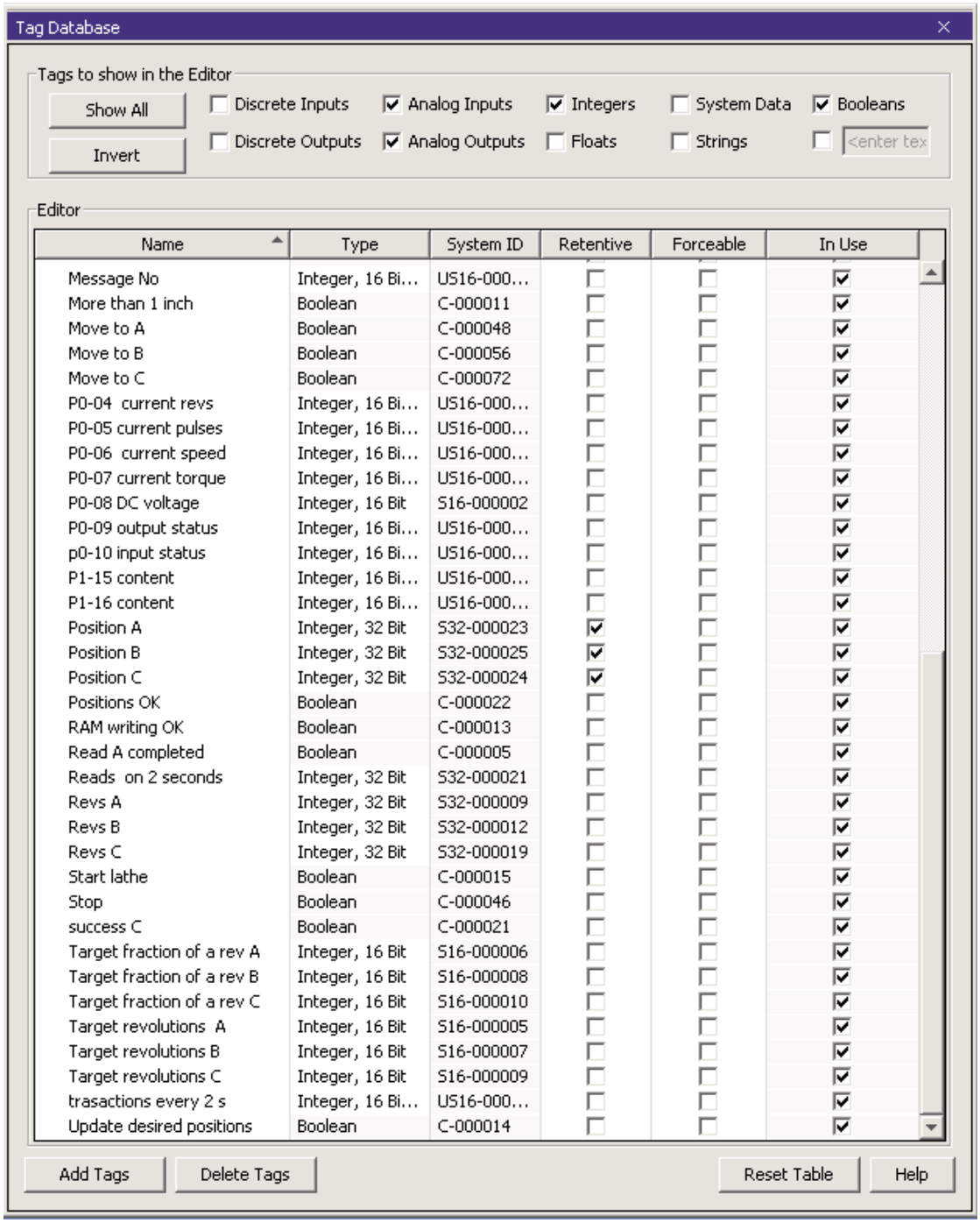
Servo output DO4 is related to the brake and the PAC output DO-0.1.2.5 is the command to run the lathe spindle that will also command the tool start (To make the groove). More in the wiring diagram.

In a tag based PAC it is recommended to create the tags before starting the programming, with the option to create the tags while you program the controller.

In any case, the tags should be created before creating the different objects within the C-more panel. The PAC creates a CSV file automatically, that can be imported to the C-more panel.

In the following figures it is shown the list of tags done for this example,





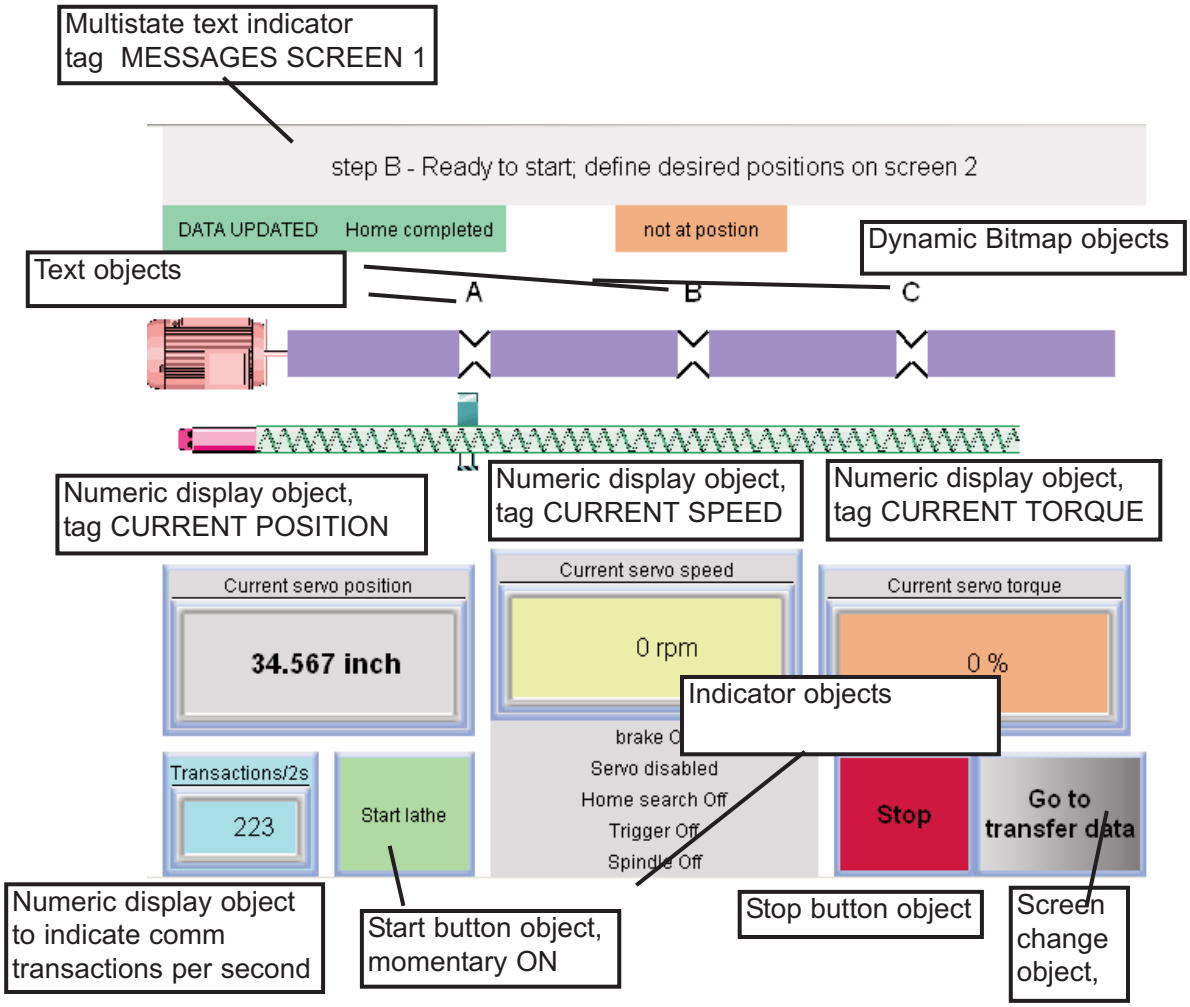
Notice that some integers are 32 bit and others are 16 bit. The reason to use tags with 16 bits is that those tags are used in conjunction with the servo drive MODBUS communication memories that only works with 16 bit integers.

Touch screen programming

The touch screen panel will have 2 screens, one to start and stop the lathe operation and the other to configure the predefined positions A, B and C.

The communication implemented on this example is serial, with K-sequence protocol, and the panel is connected to the port 1 of the PAC.

See below for a figure with a summary of the objects created on screen 1; the next page shows screen 2; then there are some other details of the objects created on the touch screen,as well as the expected operation.



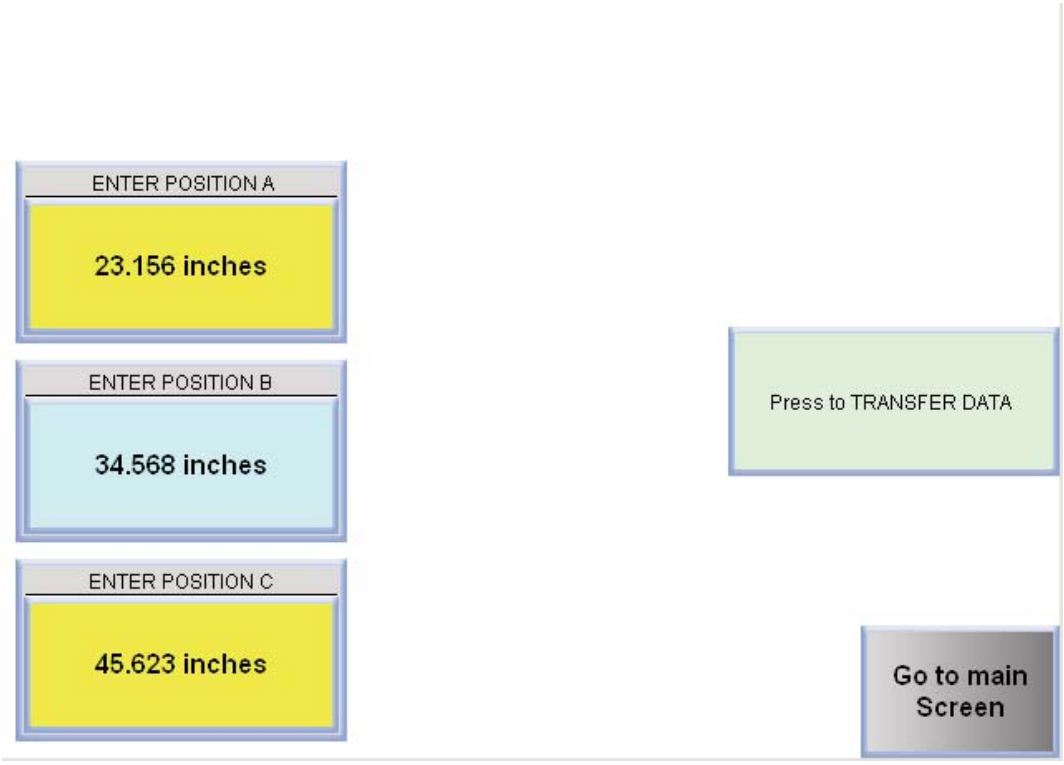
The panel screen have the following functions implemented with the objects:

Screen 1 (Main screen)

- Starting and stopping the operation (pushbuttons Start and Stop),
- Display the status of the servo and the spindle showing it as a graphical interface, by indication when the tool holder is in position and when the tool is doing the groove. The piece is green when rotating and orange when stopped. A dynamic bit map will show the tool holder on the proper position when stopped and penetrating the material.
- The numeric displays for servo position, speed and torque at any time,

- Show the communication transactions per second (to verify that there is communication in good condition),
- The indication of which step of the operation it is, in plain English by using a multistate text indicator object . The messages are:
- Indication if the tool holder is retracted or in movement.
- A button to change screen to be able to define the new desired positions.

See the rest on the figure of next page.



Screen 2

Screen 2 (New position screen)

- 3 numeric entry objects to change the positions, in inches, with 3 decimals of precision
- Pushbutton to enter the 3 new positions when touched.
- Display the status of the new positions.
- A multistate text indicator to generate an alarm if the positions are not separated for at least one inch and that the addition of the displacements are not over 108 inches
- A button to change screen to be able to go to the main screen.

The Multistate indicator may have the following messages. This has not been developed totally in this document and it is orientative in nature.

1	1 Ready to start; checking P2-30
2	2 Ready to set positions; EEPROM not being written
3	3 Let us set new positions
4	4 New positions defined
5	5 Moving to Home
6	6 Defining position A
7	7 Moving to position A
8	8 IN POSITION A
9	9 Defining Position B
10	10 Moving to position B
11	11 IN POSITION B
12	12 Defining position C
13	13 Moving to position C
14	14 IN POSITION C
15	15 END=---- go to home
16	16 Home and stop

The link between the C-more and the PAC has been done with Ethernet.

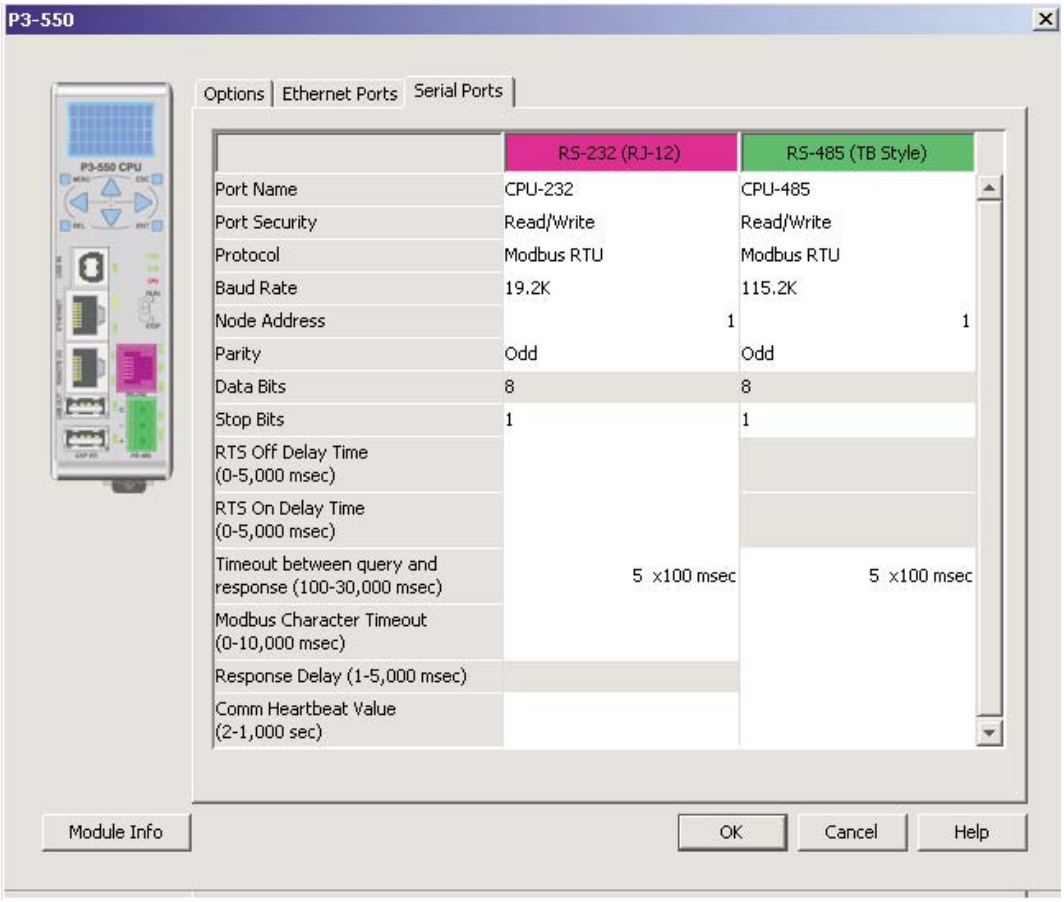
We selected the following IP addresses, arbitrarily.

- For the PAC = 123.122.121.50 configured with the help of the Productivity 3000 suite.
- For the C-more = 123.122.121.100, configured by the set up screen on the panel itself.

Concepts of MODBUS communications

We will define the servo as slave 2 and we will establish a baud rate of 115.2 kBaud, linked thru the port of the PAC, using the MODBUS RTU protocol.

The communication port RS-485 on the CPU should be configured according to the dialog box shown on the adjacent figure.



On the Servo drive, we have to set the following parameters to match the settings on the PAC:

P3-00= 2 (station o slave address)

P3-01= 5 (baud rate)

P3-02= 8 (data,parity,stop bits)

P3-05= 2 for RS-485

It is important to define the parameter **P2-30** as 5. That is why we created a routine to check, at the beginning of the operation, that the parameter P2-30 is 5. It if is not 5, the PAC will force a value of **5** into it. The MODBUS address is 400543 decimal.

The rest of the parameters will be changed to .

P0-04= 00; motor feedback pulses;

P0-05= 01; motor feedback revolutions;

P0-06= 06; motor feedback rpm;

P0-07= 11; motor feedback torque %;

P0-08 = 13 DC bus voltage within the servo drive

P0-09 = 409 to allow the parameter P4-09 to be read

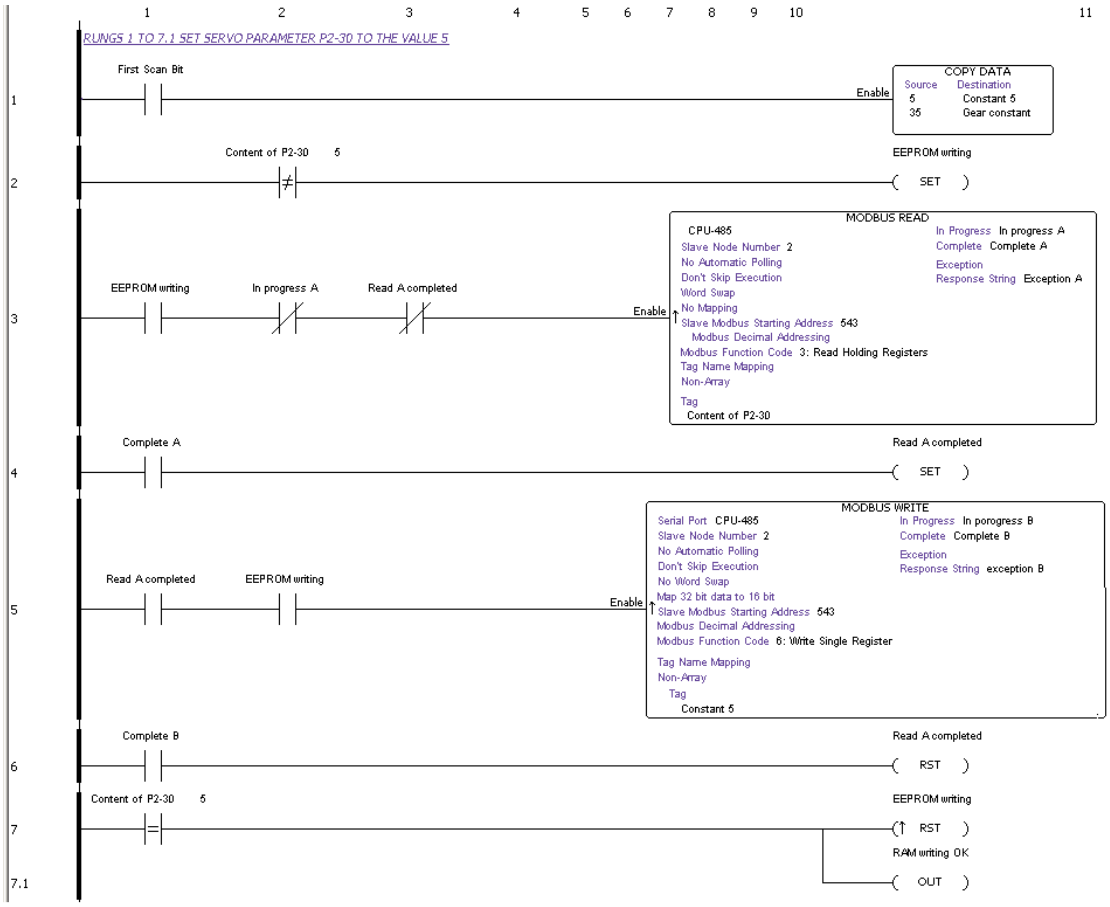
You may use the PAC RS-485 serial port to connect the pigtail end of the SVC-MDCOM-CBL cable which gets connected to the CN3 port of the servo drive.

The data to be written to the servo drive thru MODBUS are the positions A, B and C, that are written from the calculated equivalent revolutions and fractions of a revolution to the parameters P1-15 and P1-16, corresponding to the preset revolutions and preset pulses.

The trigger to transfer the data is one button on the touch panel whose associated tag is DATA TRANSFER. Note that it is not necessary to transfer data continuously. Also, that the reading is interlocked with the writing with the contact that corresponds to the tag DATA TRANSFER.

First Ladder code

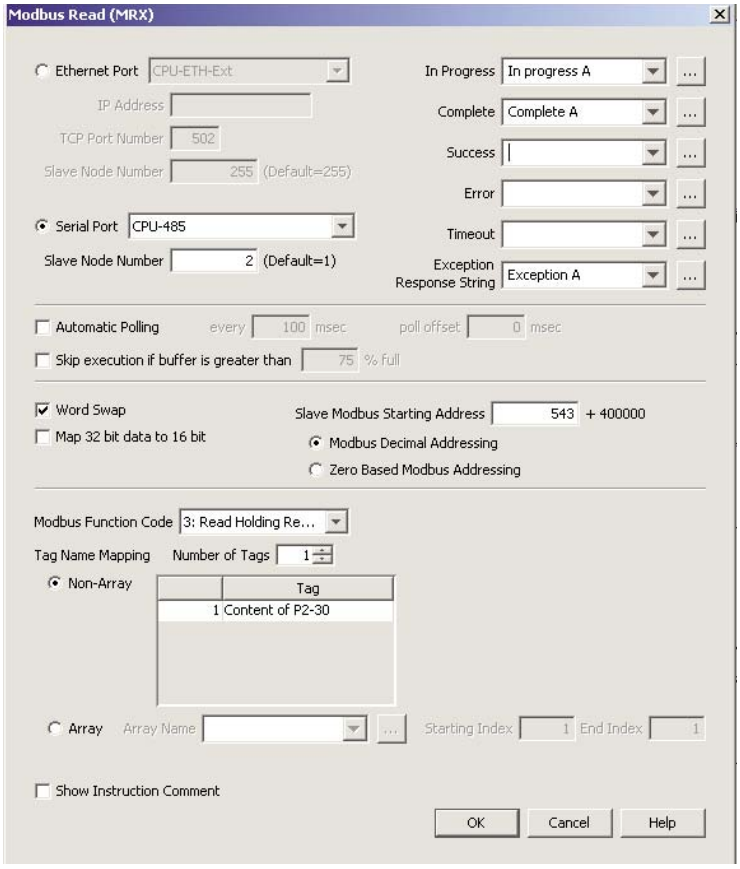
This is the initial PAC programming, to set the content of the parameter P2-30 on the servo drive as 5.



We strongly recommend to test this code before continuing

The first instruction MRX will be configured as follows:

- Select RS-485 in the field **Serial port** instead of RS-232, from the drop down menu
- Select the slave number to be read (2 in this case)
- The register decimal addressing is 40543. **Fill the field Slave Modbus Starting address** as 543. The prefix 40 is added automatically.
- Select the function mode 3 in the field **Modbus Function Mode**
- Select 1 tag in the **Tag Name Mapping** field
- Set the tag that will receive the content of P2-30. Here is CONTENT OF P2-30.



On rung 2 we compare the CONTENT OF P2-30 to the value 5. If the content is not 5 we set the tag EEPROM WRITING.

On rung 3 we read the address 400543 with the instruction MRX and copy it to the tag CONTENT OF P2-30. Notice that we have set the slave as 2, and use the function **code 3. Read Holding registers**.

On rung 4 we set the tag READ A COMPLETED if the reading with MRX has been completed, to continue the programming, namely, to set a 5 into P2-30.

On rung 5 we write a 5 into the register 400543 of the servo drive, the parameter P2-30, with the help of the instruction MWX, if the tag READ A COMPLETED is ON, and the tag EEPROM WRITING is ON.

On rung 6 we reset the tag READ A COMPLETED.

On rung 7 we reset the tag EEPROM WRITING and set the tag RAM WRITING OK.

You can monitor the status of the different tags with **Data View** or by checking the Monitor mode in the screen of the PC within the configuration software.

Code for defining to the servo the desired revolutions.

Recall that the maximum allowable position is 108 [inch] and 0 counts.

The operator will write into the C-more panel a linear distance in inches, with 3 decimal places of precision and:

- The position A should be less than the position B and also the position B should be less than the position C.
- The distance from one groove to the other cannot be less than 1 inch.

The data entry will be accomplished on the C-more panel by creating a numeric entry for each one of the positions in inches..

a) **Position A**: Let us say, just for doing an example, that the first position is 30 inches and 345 thousands of an inch.

This will be stored in the tag REVS A. The tag would contain the number 303450. This data will be scaled on the touch screen panel to show the real servo shaft revolutions and counts in the tags TARGET REVOLUTIONS A and TARGET FRACTIONS OF A REV A.

b) **Position B**: This will be stored in the tag REVS B. This data will be scaled on the touch screen panel to show the real servo shaft revolutions and counts in the tags TARGET REVOLUTIONS B and TARGET FRACTIONS OF A REV B.

c) **Position C**: This will be stored in the tag REVS C. This data will be scaled on the touch screen panel to show the real servo shaft revolutions and counts in the tags TARGET REVOLUTIONS C and TARGET FRACTIONS OF A REV C.

The operation to translate the unit inches into revolutions and counts, is done with the following concept:

108 inches will correspond to 108x5 revolutions of the lead screw = 540 revolutions. One revolution of the lead screw corresponds to 7 revolutions of the servomotor shaft. Then the data of the touch screen should be multiplied by 35 to get the proper displacement. This calculation is done scaling this data in the touch screen panel).

We should assure that there is always a distance between positions of at least 1 inch as well as the desired position is not greater than 108 inches.

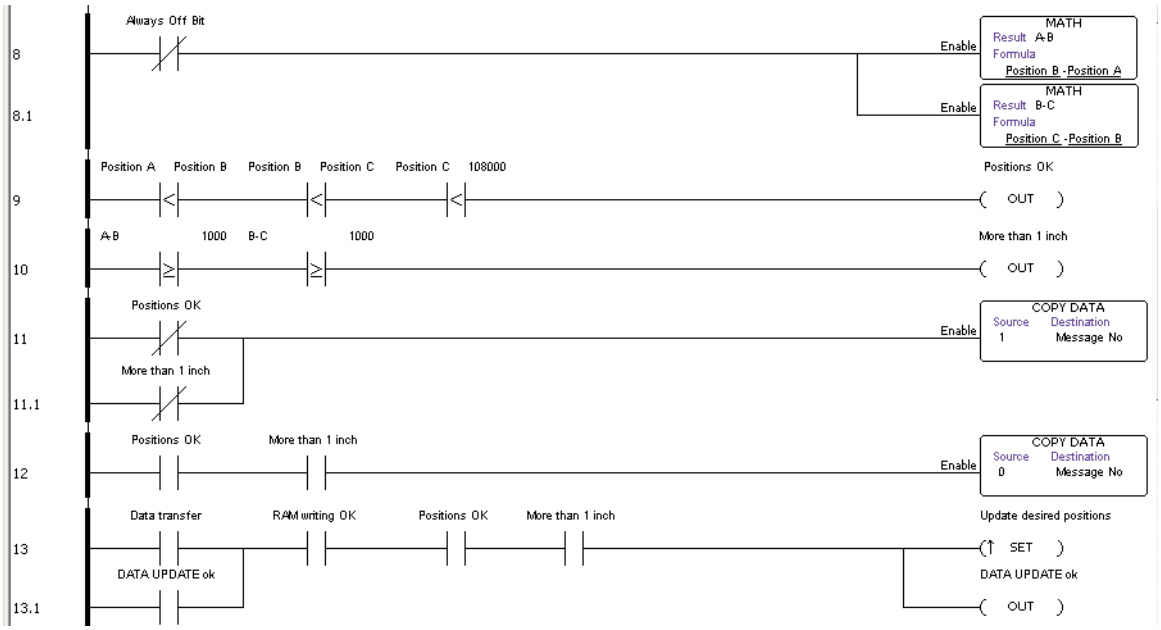
The touch screen panel will have 2 screens; the first screen will have the operator main interface, for starting and stopping the sequence, as well as showing the status of the operation. There is a button to go to the second screen.

The second screen will have the interface to enter the preset positions using 3 numeric entry objects, that allow to define the distance directly in inches.

There is also a button to go to the main screen.

The trigger to transfer the data is the button whose associated tag is TRIGGER on the touch panel. It is not necessary to transfer data continuously. Notice that the reading is interlocked with the writing using the contact TRIGGER.

On next figure we show the second part of the PAC programming to execute the concepts explained in this page.



Rung 8 calculates the distance between desired positions. Rung 9 checks that the positions A, B and C are increasing values and the position C is below 108 inches.

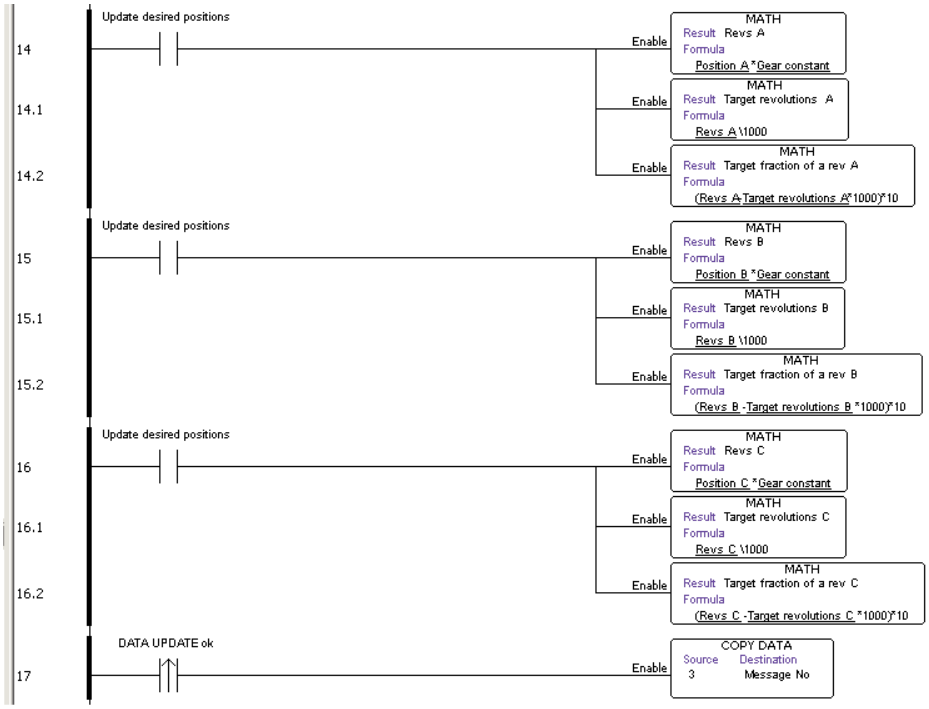
Rung 10 checks that the difference between positions is equal/ greater than 1 inch.

Rung 11 sets a number 1 into the tag MESSAGE NO to define the message of the multistate indicator in the C-more panel if the desired positions are not correct and Rung 12 sets a 0 to the tag MESSAGE NO to indicate that the values are correct.

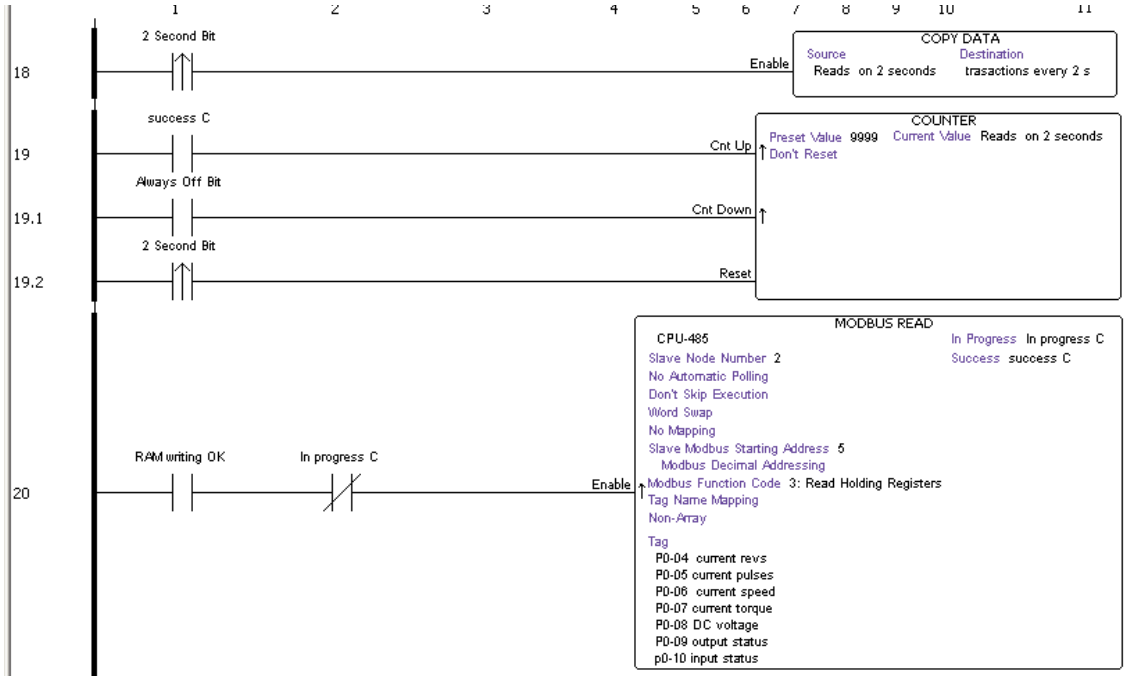
Rung 13 sets a tag as a bit DATA UPDATE OK, that allows to assure the data transfer has been done.

Note that the tag UPDATE DESIRED POSITIONS is a bit controlled by the C-more panel.

Next, we transfer the desired positions to the PAC registers.

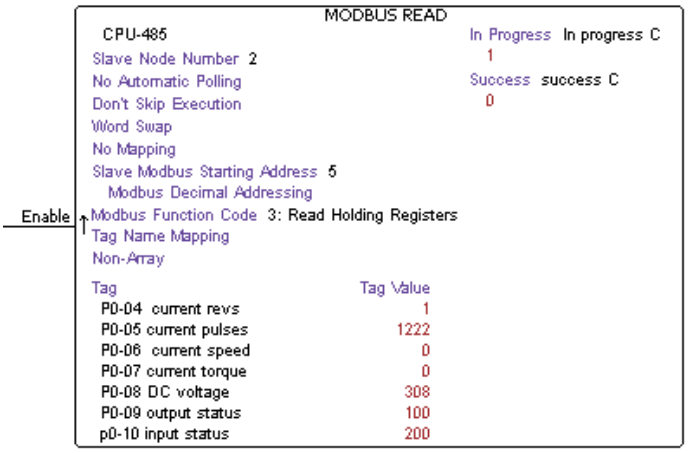


Next, we will create a code to read data from the servo. We are interested to know what is the rate the data is being transferred. For this, we will create a counter that counts the SUCCESS bit of the of the transactions and then we will reset the counter every 2 seconds.. The counts at the end of the 2 seconds is transferred to a tag called TRANSACTIONS EVERY 2 S.



At this time we can read about 396 counts, that is, we are reading 396/2 = 198 transactions every second, or one transaction every 5.05 ms. The count value will eventually decrease with more code and the addition of the C-more panel.

If you set the Monitor mode, you can see something similar to the adjacent figure.



These values mean that for the specific position at this time, the current position is 1 revolution (P0-04) and 1222 1/1000th of a revolution(-0-05, P0-6 is the speed in rpm, at this time, 0 rpm, P0-07 is 0 % of the torque applied by the servo motor to the load and the DC bus voltage inside the servo drive is 308 Volt.

The parameter P0-09 has been set with the value hexadecimal 409_h and this parameter is a block transfer parameter. This is a parameter that can be set to any other parameter and for our convenience we defined like that, to be able to read the bits of the parameters P4-09, which contain information about the servo output status.

Other parameters to be defined in the servo are:

P2-10 (DI1) 0 Input function is disabled, arbitrarily

P2-11 (DI2) 24 Home sensor N.C. (The sensor will turn OFF when at home).

P2-12 (DI3) 101 Servo enable

P2-13 (DI4) 127 Home search trigger

P2-14 (DI5) 108 Trigger

P2-15 (DI6) 22 Reverse overtravel condition

P2-16 (DI7) 23 Forward overtravel condition

P2-17 (DI8) 0 Input with function disabled

In this case the digital inputs DI2,DI3, DI6 and DI7 will be controlled by real wires.

The servo outputs will be defined as follows

P2-18 = (DO1) Servo OK

P2-19 = (DO2) At position

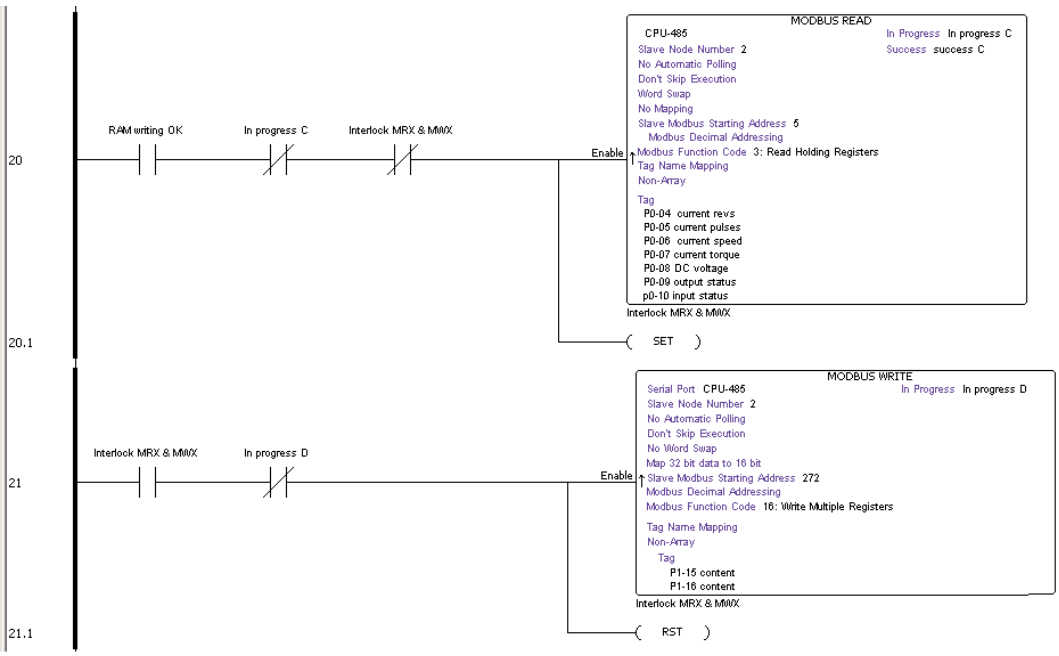
P2-20= (DO3) Home completed

P2-21 = (DO4) Servo brake control

P2-22 = (DO5) Active fault.

We are ready to create a MWX command to write data to the servo drive. This has to be interlocked with the MRX command on rung 20.

Let us see how the code stays at this time in the following figure.

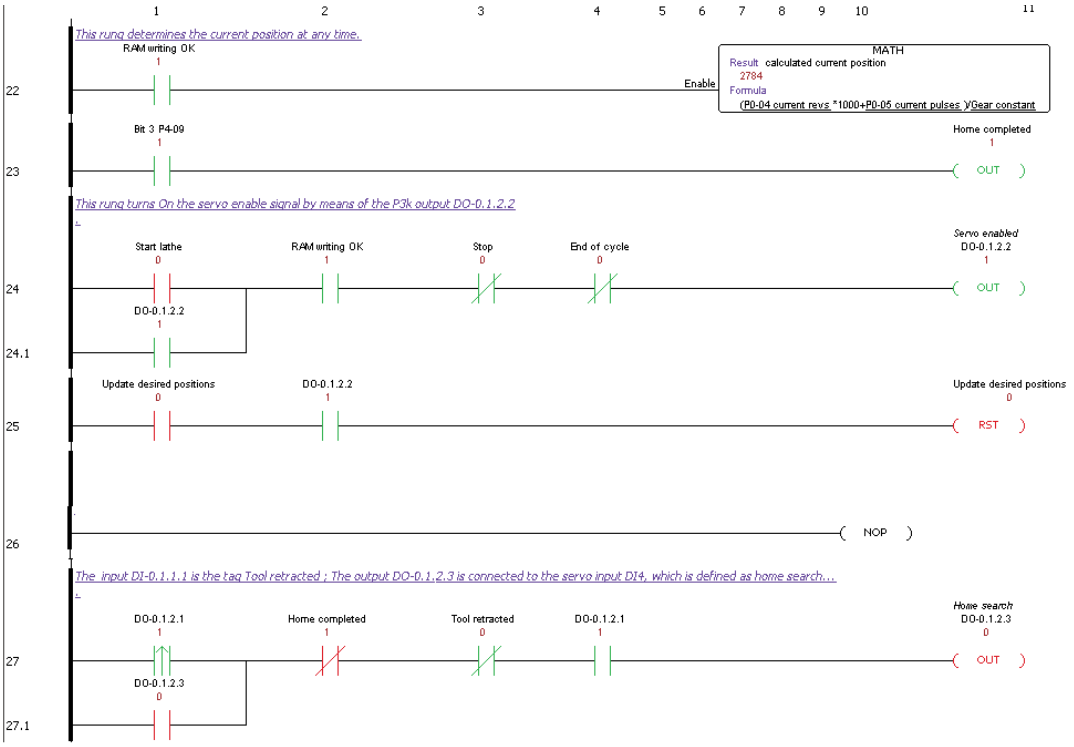


Let us do some explanation before continuing:

On rung 20 we have added an interlock bit, to allow only one read or write to be active at any given time. Notice that the instruction MRX will initiate the reading but the bit INTERLOCK MRX & MWX being ON will not allow the rung to execute another reading until the bit is reset to OFF.

On rung 21 we create a rung to write data to the drive. If the tag RAM WRITING OK is ON, we can start the control and we will set a bit SERVO ENABLE to enable the servo. This is latched since the START LATHE tag is a momentary pulse coming from the C-more touch panel.

Let us continue with the PAC programming:



On rung 22 we read the current revolutions and the fractions to calculate the current position in inches. This value will be displayed to the operator in screen 1 of the C-more panel.

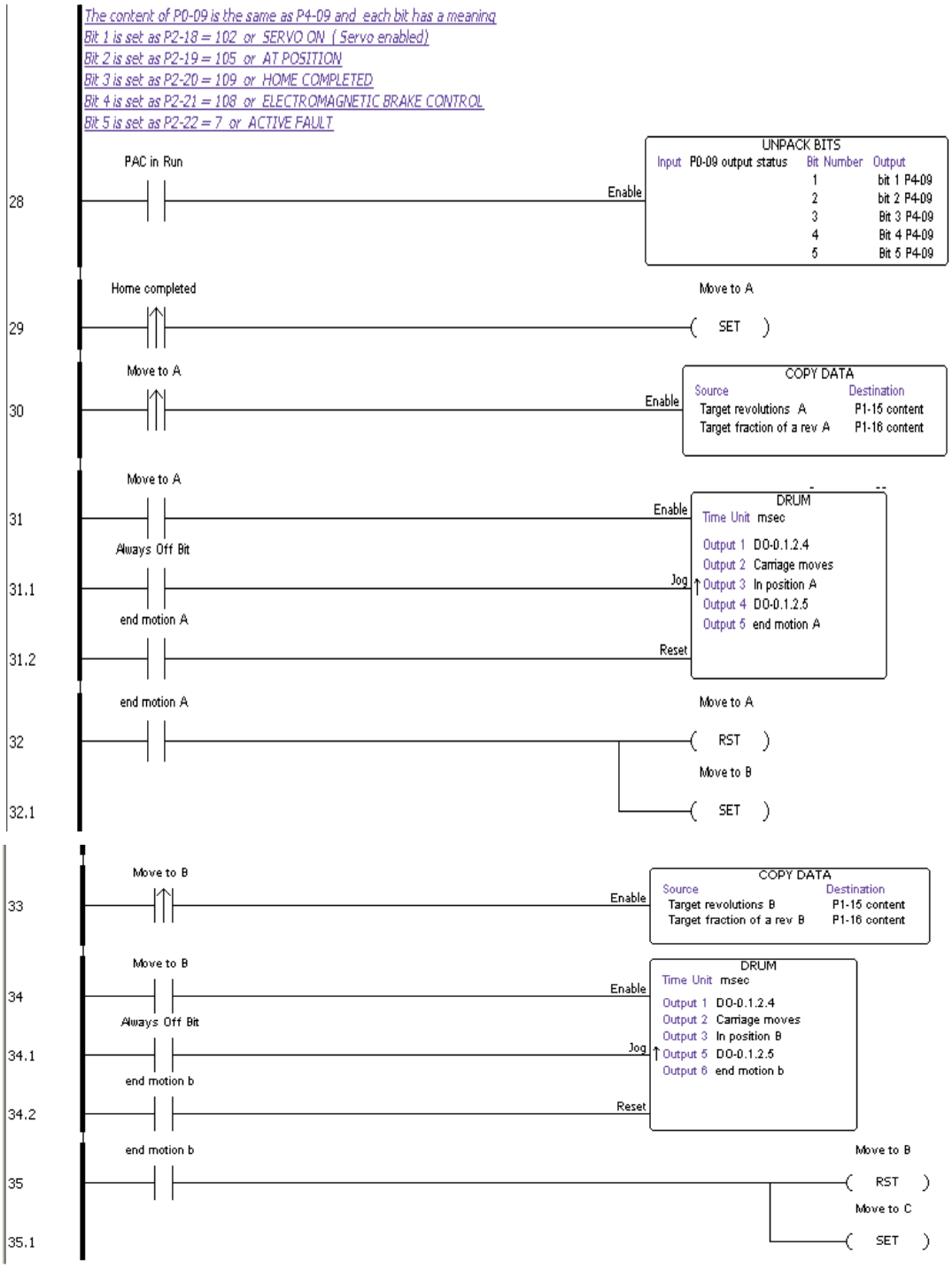
On rung 23 we create a bit with a meaningful name to indicate when the home search has been completed.

On rung 24 we have the START LATHE tag that allows the operator to begin the operation of the servo. When this is done the servo is enabled by means of the PAC output DO-0.1.2.2.

On rung 25 we reset the bit whose tag is UPDATE DESIRED POSITIONS.

On rung 26 we do not have any operation.

On rung 27 we initiate the motion to search for the home sensor, when the servo is OK, that is, when the servo has been confirmed enabled and, as conditions, the Home should NOT be completed, the tool shall be retracted and the servo should be enabled (with the output SERVO OK). This turns ON the PAC output DO-0.1.2.3 that is linked to DI4-. The DO-0.1.2.3 signal is latched and only unlatches when the Home has been found, with the HOME COMPLETED signal.



On rung 28, with the instruction UNPACK BITS, we can see the status of each bit on word P4-09, read with MODBUS, and will detect that the servo is enabled with the output SERVO OK, whose status has been read in rung 20 with the data in P0-09 OUTPUT STATUS.

On rung 29 we SET the bit MOVE TO A in the transition from OFF to ON of the tag HOME COMPLETED;

The content of the tag TRANSACTIONS EVERY 2 S has decreased to 224. This is because the writing instruction is taking an additional time to process the data.

On rung 30 we set the **Copy data** instruction in the transition from OFF to ON of the tag MOVE TO A that allows to copy the calculated target revolutions and fractions to other tags that will be transferred with the MODBUS MWX instruction to the proper register in the servo, that is the parameters P1-15 and P1-16.

On rung 31 we have a DRUM instruction, that allows to control the tool holder motion, simulated in this case.

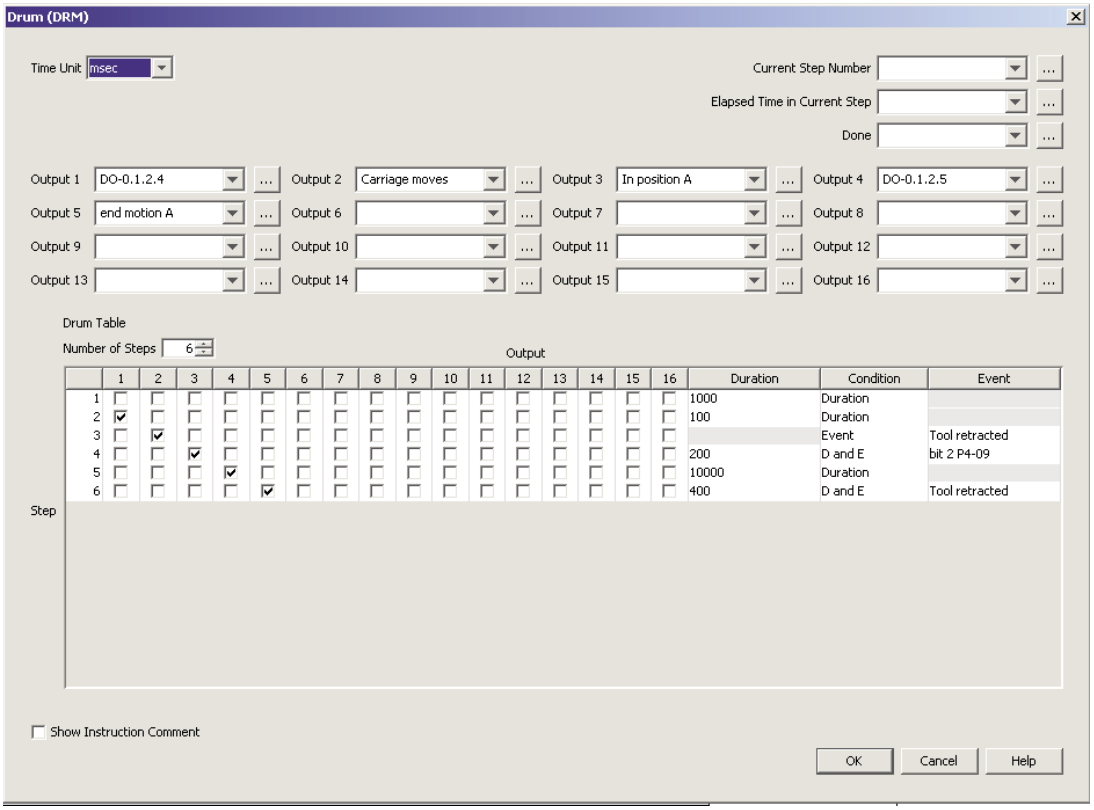
See the figure on next page for the following explanations of the operation.

On step 1, there is a waiting time of 1 second or 1000 ms.

On step 2, we activate the output DO-0.1.2.4 for 100 ms, output that is wired to the servo input DI5-, programmed as TRIGGER. When this bit is turned ON, the carriage or tool holder will move to the position A, and the servo will position itself in the desired position. The AT POSITION condition should turn on.

On step 3, the tool retracted condition shall be on, in order to allow to go to the next step.

On step 4, P4-09 bit 2 (which is the AT POSITION signal) shall turn on, and then goes to next step. The tag IN POSITION A turns ON in the PAC and allows the C-more to indicate this condition.



On step 5, the output DO-0.1.2.5 will turn on, and this allows the spindle motor to rotate for at least 10 seconds.

Finally, the tool retracted shall be ON again to activate the output END MOTION A.

On rung 32 the signal end motion A will reset the tag MOVE TO A and at the same time will set MOVE TO B.

On rung 33, we set the **Copy data** instruction in the transition from OFF to ON of the tag MOVE TO B that allows to copy the calculated target revolutions and fractions to other tags that will be transferred with the MODBUS MWX instruction to the proper register in the servo, that is the parameters P1-15 and P1-16.

On rung 34, we have a second DRUM instruction, that allows to control the tool holder motion to desired position B, simulated in this case in the same way as for position A..

Finally the tool retracted shall be ON again to activate the output END MOTION B.

In the same way, we will move the tool holder to desired position C and the end of the motion will end the cycle.

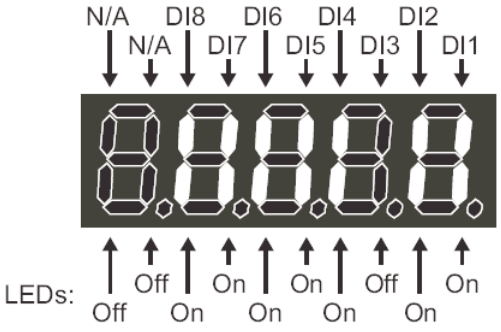
PAC- Servo drive wiring tests

The wiring between the C-more and the PAC is tested using the Data View on the PAC configuration program. The data has to be listed on the Data View and when the PAC and touch panel are energized, the programmer has to check that the panel is writing data on the PAC and also by writing data on the PAC by means of the Data View Dialog box, the touch panel should display the same data on the corresponding screen.

The wiring between the PAC and the servo can can be tested using the overrride function on the PAC.

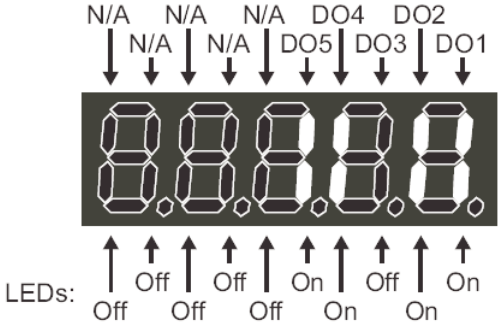
When the PAC outputs are wired to the servo, parameter P4-07 allows testing of the corresponding digital inputs. For that test, select the parameter P4-07 with the keypad, and then ENTER.

Every time that one digital input is activated, the corresponding LEDS will turn ON on the servo drive display, per the adjacent diagram.



Similarly, the parameter P4-09 shows the status of the servo drive digital outputs and allows for testing the corresponding PAC digital inputs. For that action, select the parameter P4-09 with the keypad and then ENTER.

Every time that one digital output is activated, the corresponding LEDS will turn ON on the servo drive display, per the adjacent diagram.



At this time it would help to create a table with the I/0 assignments:

Servo	Parameter	Value	Description
DI1	P2-10	101	Servo enable
DI2	P2-11	108	Command trigger
DI3	P2-12	022	Reverse overtravel (directly connected)
DI4	P2-13	127	Search home trigger
DI5	P2-14	023	Forward overtravel (directly connected)
DI6	P2-15	024	Home sensor (works with X21)

and so on.

Next it would be good to do the tuning of the system, if necessary, before testing the operation.The tuned system will allow testing of the system without any disturbance caused by bad tuning.

It is necessary to define the number for the messages in the multistate indicator.

At this time, this is left to the reader for lack ot time to complete the project.

We do not claim that this is the best program for the application.

It has been done with a lot of detail in order that the readers can follow the logic. The program only works for this specific example and there are, alternatively, many other codes to execute the same functions.

The PAC program has a scan time of 0.4 ms on this unit.

As described before, the baud rate selected was 115.2 kbps, and with this speed the normal transitions per 2 seconds are on the order of 224, that is, typically the update of the data reading will take less than 10 milliseconds.

During the programming and test, the PAC was communicating with the configuration software thru a USB cable to access the PAC while the Ethernet port was used for the C-more panel; and the RS-485 port was used to communicate to the servo drive. The C-more panel communicates with PC through USB. In this way, the program can be developed and monitored and changed easily.

The home sensor used on the prototype is a photocell. Proximity sensors can be used alternatively.

Automation Direct SureServo PRO Drives Configuration Report
Report Generated: 5/1/2010 4:29:45 PM
Motor Code: 11
Rev: 2.105

Parameter	Value	
P0.00 - Software Version	2105	
P0.01 - Drive Fault Code	0	
P0.02 - Drive Status (Front panel display)	0	
P0.03 - Analog Monitor Outputs	1	*Entered by keypad*
P0.04 - Status Monitor 1	1	*Entered by keypad*
P0.05 - Status Monitor 2	0	*Entered by keypad*
P0.06 - Status Monitor 3	6	*Entered by keypad*
P0.07 - Status Monitor 4	11	*Entered by keypad*
P0.08 - Status Monitor 5	13	*Entered by keypad*
P0.09 - Block transfer parameter 1	*409*	*Entered by keypad*
P0.10 - Block transfer parameter 2	*0*	
P0.11 - Block transfer parameter 3	*0*	
P0.12 - Block transfer parameter 4	*0*	*)*
P0.13 - Block transfer parameter 5	*0*	
P0.14 - Block transfer parameter 6	*0*	
P0.15 - Block transfer parameter 7	*0*	
P1.00 - External Pulse Input Type	2	
P1.01 - Control Mode and Output Direction	101	Position with internal registers
P1.02 - Speed and Torque Limit	1	
P1.03 - Output Polarity Setting	0	
P1.04 - Analog Monitor Output Scaling 1 (CH1)	100	
P1.05 - Analog Monitor Output Scaling 2 (CH2)	100	
P1.06 - Analog Velocity Command Low-pass Filter	0	
P1.07 - Analog Torque Command Low-pass Filter	0	
P1.08 - Position Command Low-pass Filter	0	
P1.09 - Preset Velocity Command / Limit 1	100	
P1.10 - Preset Velocity Command / Limit 2	200	
P1.11 - Preset Velocity Command / Limit 3	300	
P1.12 - Preset Torque Command / Limit 1	100	
P1.13 - Preset Torque Command / Limit 2	100	
P1.14 - Preset Torque Command / Limit 3	100	
P1.15 - Position 1 Command (Revolutions)	0	Variable through MODBUS
P1.16 - Position 1 Command (Counts)	0	Variable through MODBUS
P1.17 - Position 2 Command (Revolutions)	0	Variable through MODBUS
P1.18 - Position 2 Command (Counts)	0	
P1.19 - Position 3 Command (Revolutions)	0	
P1.20 - Position 3 Command (Counts)	0	
P1.21 - Position 4 Command (Revolutions)	0	
P1.22 - Position 4 Command (Counts)	0	

P1.23 - Position 5 Command (Revolutions)	0	
P1.24 - Position 5 Command (Counts)	0	
P1.25 - Position 6 Command (Revolutions)	0	
P1.26 - Position 6 Command (Counts)	0	
P1.27 - Position 7 Command (Revolutions)	0	
P1.28 - Position 7 Command (Counts)	0	
P1.29 - Position 8 Command (Revolutions)	0	
P1.30 - Position 8 Command (Counts)	0	
P1.31 - Motor Code	11	200 watt for this prototype
P1.32 - Motor Stop Mode Selection	0	
P1.33 - Position Control Mode (Internal Indexer)	1	
P1.34 - Acceleration Time (Internal Indexer)	1000	
P1.35 - Deceleration Time (Internal Indexer)	1000	
P1.36 - Accel / Decel S-Curve	20	
P1.37 - Inertia Mismatch Ratio	5	
P1.38 - Zero Speed Output Threshold	10	
P1.39 - Target Speed Output Threshold	3000	
P1.40 - Max Analog Velocity Cmd or Velocity Limit	3000	
P1.41 - Max Analog Torque Cmd or Torque Limit	100	
P1.42 - On Delay Time of Electromagnetic Brake	20	
P1.43 - Off Delay Time of Electromagnetic Brake	20	
P1.44 - Electronic Gear Numerator 1	1	
P1.45 - Electronic Gear Denominator	1	
P1.46 - Encoder Output Scaling Factor	1	
P1.47 - Homing Mode	223	
P1.48 - Homing Speed 1 Fast Search Speed	240	
P1.49 - Homing Speed 2 Creep Speed	50	
P1.50 - Home Position Offset (Revolutions)	0	
P1.51 - Home Position Offset (Counts)	0	
P1.52 - Regenerative Resistor Value	40	
P1.53 - Regenerative Resistor Capacity	60	
P1.54 - In Position Window	100	
P1.55 - Maximum Speed Limit	3500	
P1.56 - Output overload warning threshold	120	
P2.00 - Proportional Position Loop Gain (KPP)	35	
P2.01 - Position Loop Gain Boost	100	
P2.02 - Position Feed Forward Gain (KFF)	5000	
P2.03 - Smoothing Constant of Position Feed Forward Gain	5	
P2.04 - Velocity Loop Proportional Gain (KVP)	500	
P2.05 - Velocity Loop Gain Boost	100	
P2.06 - Velocity Loop Integral Compensation (KVI)	100	
P2.07 - Velocity Feed Forward Gain (KVF)	0	
P2.08 - Factory Defaults and Security	0	
P2.09 - Bounce Filter	2	
P2.10 - Digital Input Terminal 1 (DI1)	0	
P2.11 - Digital Input Terminal 2 (DI2)	24	Home sensor
P2.12 - Digital Input Terminal 3 (DI3)	101	Servo enable

P2.13 - Digital Input Terminal 4 (DI4)	127	Home trigger
P2.14 - Digital Input Terminal 5 (DI5)	108	Command trigger
P2.15 - Digital Input Terminal 6 (DI6)	22	Forward Inhibit
P2.16 - Digital Input Terminal 7 (DI7)	23	Reverse inhibit
P2.17 - Digital Input Terminal 8 (DI8)	0	Disabled
P2.18 - Digital Output Terminal 1 (DO1)	102	Servo OK
P2.19 - Digital Output Terminal 2 (DO2)	105	At position
P2.20 - Digital Output Terminal 3 (DO3)	109	Homing completed
P2.21 - Digital Output Terminal 4 (DO4)	108	Brake control
P2.22 - Digital Output Terminal 5 (DO5)	7	
P2.23 - Notch Filter (Resonance Suppression)	1000	
P2.24 - Notch Filter Attenuation	0	
P2.25 - Low-pass Filter (Resonance Suppression)	2	
P2.26 - External Anti-Interference Gain	0	
P2.27 - Gain Boost Control	0	
P2.28 - Gain Boost Switching Time	10	
P2.29 - Gain Boost Switching Condition	10000	
P2.30 - Auxiliary Function	0	
P2.31 - Auto and Easy Tuning Mode Response Level	68	
P2.32 - Tuning Mode	0	
P2.33 - Reserved	0	
P2.34 - Overspeed Fault Threshold	5000	
P2.35 - Position Deviation Fault Window	30000	
P2.36 - Position 1 Velocity	3000	
P2.37 - Position 2 Velocity	1000	
P2.38 - Position 3 Velocity	1000	
P2.39 - Position 4 Velocity	1000	
P2.40 - Position 5 Velocity	1000	
P2.41 - Position 6 Velocity	1000	
P2.42 - Position 7 Velocity	1000	
P2.43 - Position 8 Velocity	1000	
P2.44 - Digital Output Mode	0	
P2.45 - Index Mode Output Signal Delay Time	1	
P2.46 - Index Mode Stations	6	
P2.47 - Position Deviation Clear Delay Time	0	
P2.48 - Backlash Compensation (Index Mode)	0	
P2.49 - Jitter Suppression	0	
P2.50 - Clear Position Mode	1	
P2.51 - Servo Enable Command	0	
P2.52 - Dwell Time 1 - Auto Index Mode	0	
P2.53 - Dwell Time 2 - Auto Index Mode	0	
P2.54 - Dwell Time 3 - Auto Index Mode	0	
P2.55 - Dwell Time 4 - Auto Index Mode	0	
P2.56 - Dwell Time 5 - Auto Index Mode	0	
P2.57 - Dwell Time 6 - Auto Index Mode	0	
P2.58 - Dwell Time 7 - Auto Index Mode	0	

P2.59 - Dwell Time 8 - Auto Index Mode	0	
P2.60 - Electronic Gear Numerator 2	1	
P2.61 - Electronic Gear Numerator 3	1	
P2.62 - Electronic Gear Numerator 4	1	
P2.63 - Velocity and Position Deviation Scaling Factor	0	
P2.64 - Advanced Torque limit Mixed mode	0	
P2.65 - Special inout functions	HEX 0	
P3.00 - Communication Address	2	Slave 2
P3.01 - Transmission Speed	5	115.2 kbps
P3.02 - Communication Protocol	8	MODBUS RTU
P3.03 - Communication Fault Action	0	
P3.04 - Communication Watchdog Time Out	0	
P3.05 - Communication Selection	0	
P3.06 - Reserved	0	
P3.07 - Communication Response Delay Time	0	
P3.08 - Digital Input Software control Mask	0	
P4.00 - Fault Record - Most recent (N)	9	
P4.01 - Fault Record (N-1)	9	
P4.02 - Fault Record (N-2)	9	
P4.03 - Fault Record (N-3)	9	
P4.04 - Fault Record (N-4)	9	
P4.05 - JOG Function	2000	
P4.06 - Force Outputs Command	0	
P4.07 - Input Status	0	
P4.08 - Reserved	0	
P4.09 - Output Status	2	
P4.22 - Analog Velocity Input Offset	0	
P4.23 - Analog Torque Input Offset	0	