User Guide for working with VisiLogic TCP/IP

SM35-J-TA22 PLC

Contents

1 Electrical connections **1**

2 Software **1**

3 Programming **3**

3.1 Connect, Activate and Home 3

3.2 Sending Commands 4

3.3 Receiving Feedback 7

3.4 Receiving variables 7

3.5 Jog Menu 9

1 Electrical connections

Three connections are required for the SM35-J-TA22 PLC:

- a 24V power connection;
- a line of communication to the PC for VisiLogic programming (Serial or COMM depending on model);
- a line of communication with the Meca500 (direct Ethernet connection, Ethernet switch or other) for interfacing and executing Meca500 control programs.

Ensure the above are properly connected before proceeding to make a connection between VisiLogic, the PLC, and/or the Meca500.

2 Software

VisiLogic is an application development software for a range of Unitronics PLCs. It is free and available for download on the Unitronics website[1].

Upon the first initialization of VisiLogic a 'Hardware Configuration' screen will open and prompt you to select your PLC and I/O configuration as shown in Figure 1. Configure as required for your application.

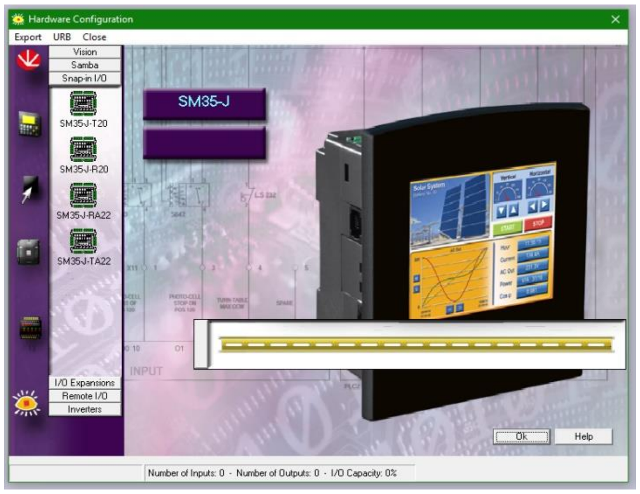


Figure 1 – PLC and I/O configuration

[1] At the time of this writing the latest firmware version is 9.8.91

After configuring your hardware you will be greeted with a blank HMI display, this is your 'Start-Up' display. You may configure it with any buttons, images, LEDs or other tools found in the HMI sidebar as highlighted in Figure 2.

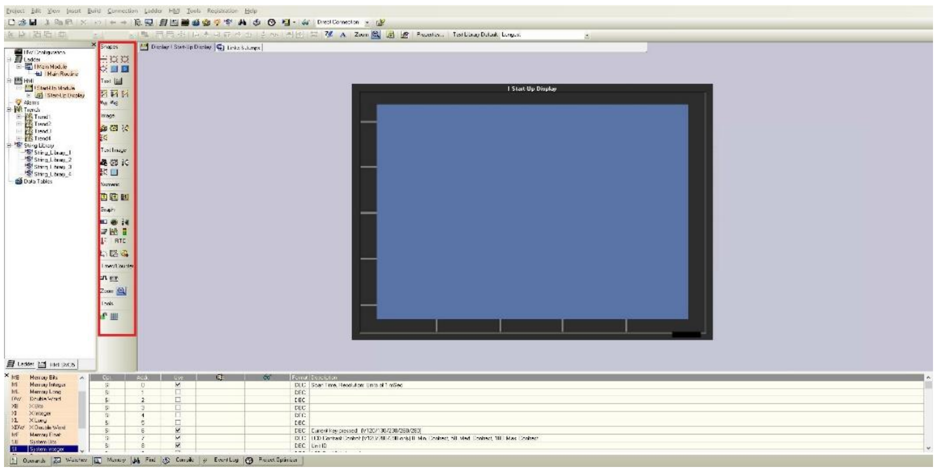


Figure 2 – Start-Up Display

In order to write LADDER logic for your application, double click on the 'Main Routine' button in the side bar as highlighted in Figure 3.

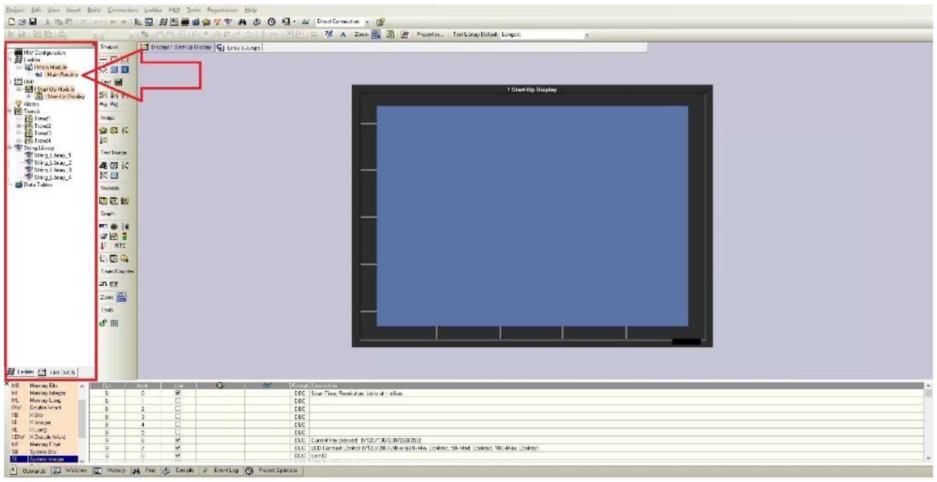


Figure 3 – Main Tree Side Bar

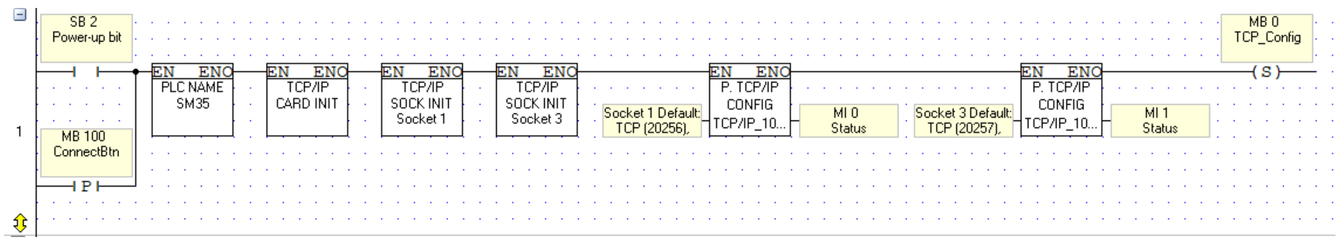
Once you are on the 'Main Routine' page you are ready to program TCP/IP communication with the Meca500.

3 Programming

This section will demonstrate different ways to program a Mecademic robot using Visilogic. The communication will be done using TCP/IP.

3.1 Connect, Activate and Home

Establishing a TCP/IP connection between the meca500 and the Unitronics PLC require some specific steps. An example of a typical first line[2] is provided in Figure 4.

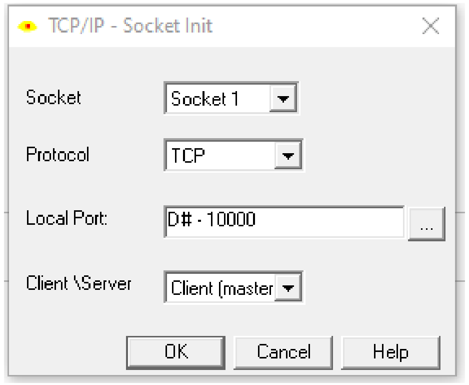


SB 2 Power-up bit
MB 100 ConnectBtn
P
EN ENO PLC NAME SM35
EN ENO TCP/IP CARD INIT
EN ENO TCP/IP SOCK INIT Socket 1
EN ENO TCP/IP SOCK INIT Socket 3
Socket 1 Default: TCP (20256),
EN ENO P. TCP/IP CONFIG TCP/IP_10...
MI 0 Status
Socket 3 Default: TCP (20257),
EN ENO P. TCP/IP CONFIG TCP/IP_10...
MI 1 Status
MB 0 TCP_Config
S
1

Figure 4 – First Line for TCP/IP Communication

In this net, we have a system bit (SB) as well as 4 different function blocks whose functions are detailed as follows:

- 'SB-2' is a system bit that is HIGH only when the PLC initializes
- 'Set PLC Name' block sets the PLC's name. It is important that the PLC name be the same as the PLC associated with the project
- 'Card Init' block initiates the TCP/IP card in the PLC. This is where you set the IP address, subnet mask and default gateway of the PLC. Since the robot's default IP address is 192.168.0.100 you must configure the PLC's IP address appropriately (i.e. 192.168.0.XXX, where XXX is any number except 100)
- 'Socket Init' block initiates the TCP/IP communication


TCP/IP - Socket Init

Socket: Socket 1
Protocol: TCP
Local Port: D# - 10000
Client \Server: Client (master

Figure 5 – Socket Init Configuration

[2] It is recommended to place this line as the first line in the main ladder.

- 'Configuration' block configures the TCP/IP communication and acts as a dispatcher for the incoming and outgoing messages sent over TCP/IP. You can configure the IP address and port number of the Meca500 to correspond with the index value of your choice.

The net shown in Figure 6 provides an example of how to connect to the socket configured in the previous section.

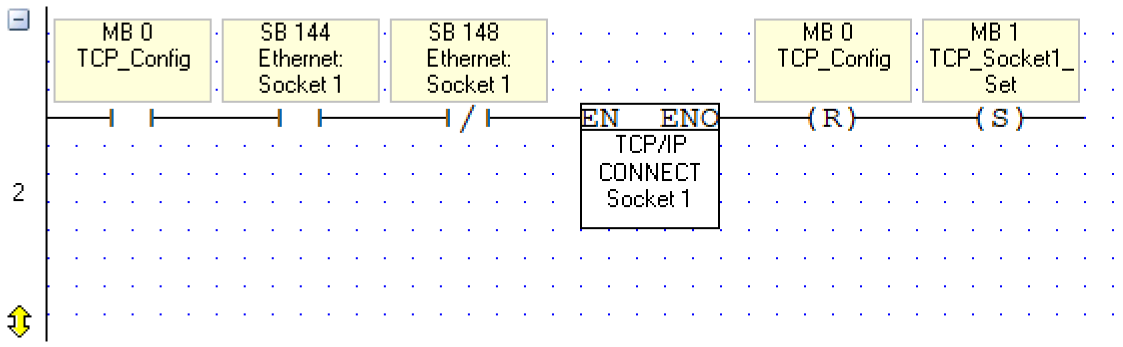


Figure 6 – Connect to Socket

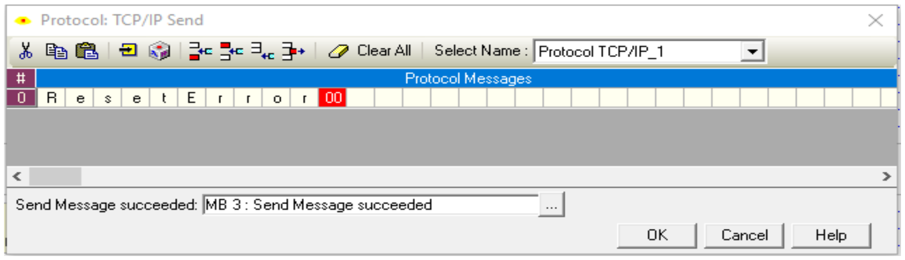
There are two system bits and one function block in net 2, each of their functions are detailed as follows:

- 'SB 144' is a bit that is set to HIGH when socket 1 initialization succeeds
- 'SB 148' is a bit that is set to HIGH when socket 1 connection is established. It is placed on an inverted contact to allow power to pass to the subsequent logic during the first PLC cycle*
- 'Connect TCP' is a function block that executes the TCP/IP connection procedure. You specify which socket you would like to connect to, as well as the IP address of the Meca500 and its port

3.2 Sending Commands

The Meca500 accepts commands in the form of ASCII strings. Luckily, the 'Protocol: TCP/IP Send' function block sends ASCII strings by default, so there is not any configuration to be done.

Further, the Meca500 searches for the NULL terminator character at the end of a command so it knows where commands end. It is thus important that you insert a NULL terminator at the end of each message. An example of a message is shown in Figure 7.

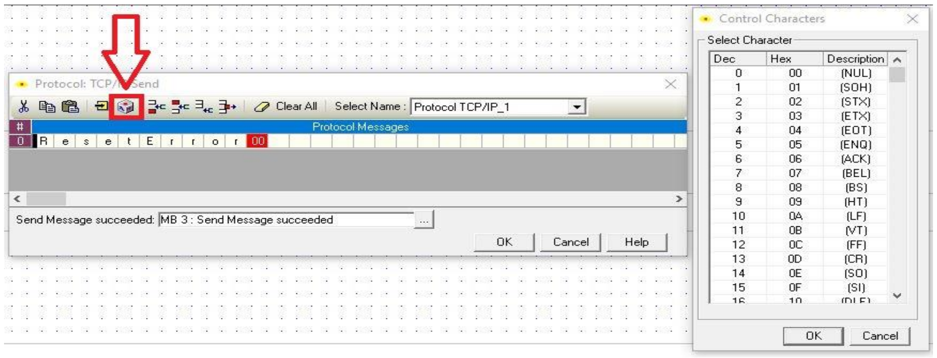


#												
0	R	e	s	e	t	E	r	r	o	r	00	

Send Message succeeded: MB 3 : Send Message succeeded

Figure 7 – ASCII String

The NULL control character can be added by selecting the 'Add Control Char' button and selecting the NULL character from the pop-up window as shown in Figure 8.



Select Name : Protocol TCP/IP_1

#	Protocol Messages
0	R e s e t E r r o r 00

Send Message succeeded: MB 3 : Send Message succeeded

Dec	Hex	Description
0	00	(NUL)
1	01	(SOH)
2	02	(STX)
3	03	(ETX)
4	04	(EOT)
5	05	(ENQ)
6	06	(ACK)
7	07	(BEL)
8	08	(BS)
9	09	(HT)
10	0A	(LF)
11	0B	(VT)
12	0C	(FF)
13	0D	(CR)
14	0E	(SO)
15	0F	(SI)
16	10	(DLE)

Figure 8 – NULL Character

There are certain commands we want to send to the Meca500 for initialization. Such commands include 'ResetError', 'ActivateRobot', 'Home' as well as others to set reference frames, speeds and accelerations. In general, we only want to send these commands once, and we want to send them during the first PLC cycle.

The sequence of nets in Figure 9 exemplifies how to send an initial sequence of messages to the Meca500 over TCP/IP after having connected to the socket as outlined by the previous section.

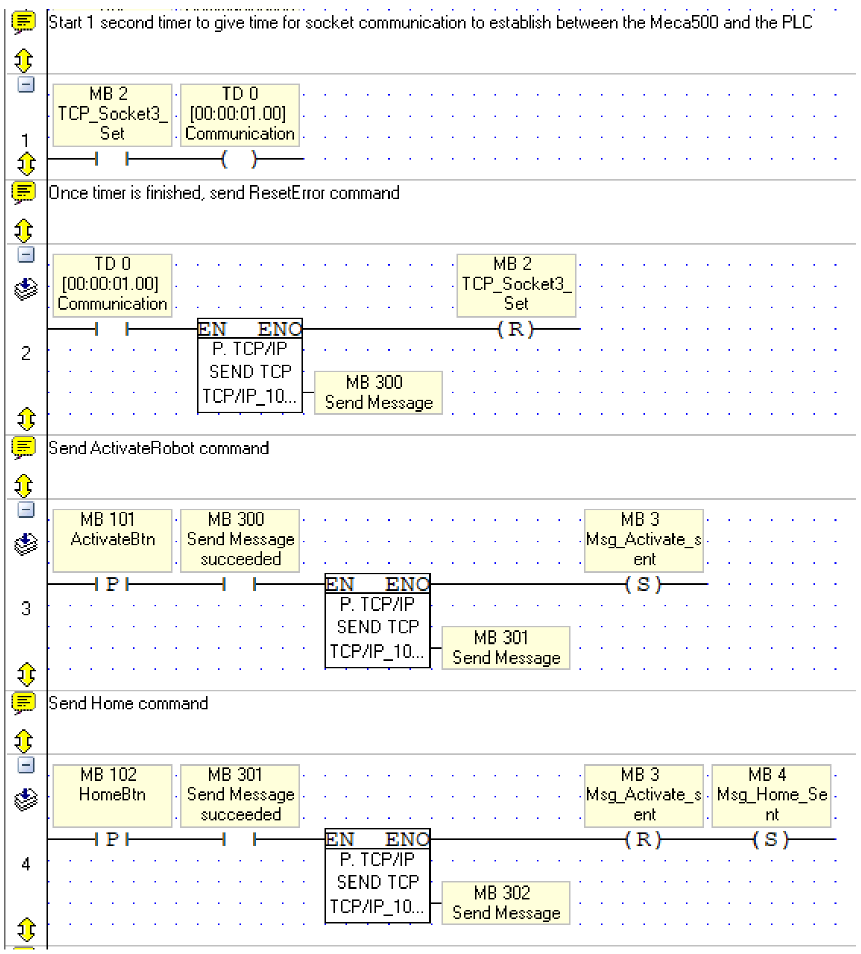
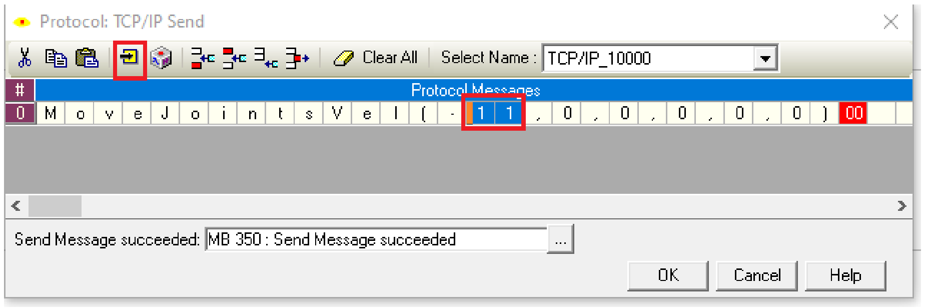

Start 1 second timer to give time for socket communication to establish between the Meca500 and the PLC
1
MB 2 TCP_Socket3_Set
TD 0 [00:00:01.00] Communication
Once timer is finished, send ResetError command
2
TD 0 [00:00:01.00] Communication
EN ENO
P. TCP/IP SEND TCP TCP/IP_10...
MB 300 Send Message
MB 2 TCP_Socket3_Set
R
Send ActivateRobot command
3
MB 101 ActivateBtn
P
MB 300 Send Message succeeded
EN ENO
P. TCP/IP SEND TCP TCP/IP_10...
MB 301 Send Message
MB 3 Msg_Activate_sent
S
Send Home command
4
MB 102 HomeBtn
P
MB 301 Send Message succeeded
EN ENO
P. TCP/IP SEND TCP TCP/IP_10...
MB 302 Send Message
MB 3 Msg_Activate_sent
R
MB 4 Msg_Home_Sent
S

Figure 9 – Start Sequence

- Net 1 – A delay provides time to establish a connection between the PLC and the Meca500
- Net 2 – The first message is sent to the robot. In this case, it is a “ResetError” command
- Nets 3 and 4 – The “Activate” and “Home” commands are sent respectively when the associated button is pressed

The pattern of activation bit HIGH, followed by command, then reset activation bit to LOW, followed by command bit HIGH is used to ensure that only one command is sent at once.

To send a variable, you need to integrate a variable into the string to be sent. To do so, you can refer at Figure 10 and Figure 11. As you can see in Figure 11, the particular variable was configured to send a two digit speed that will be set in the jog menu. Using this for every axis give us a simple and effective way to have a user settable variable jogging speed.


Protocol: TCP/IP Send

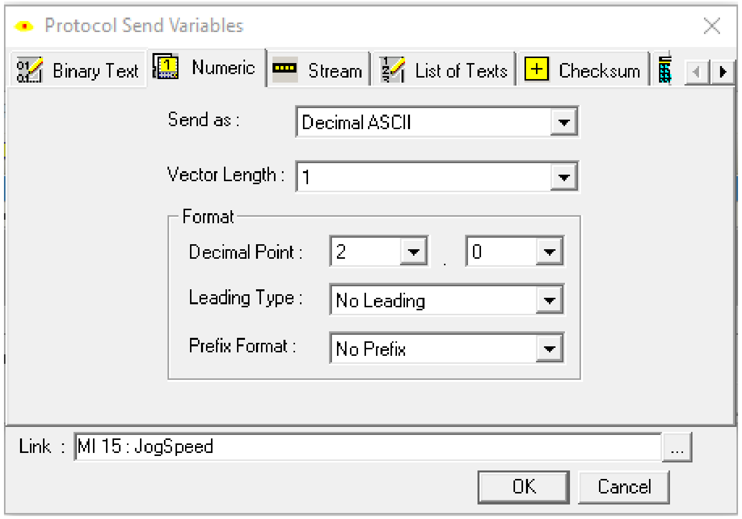
Select Name : TCP/IP_10000

Protocol Messages

0 | M o v e J o i n t s V e l (- 1 1 , 0 , 0 , 0 , 0 , 0) 00

Send Message succeeded: MB 350 : Send Message succeeded

Figure 10 – Variable in a String


Protocol Send Variables

Send as : Decimal ASCII

Vector Length : 1

Format

Decimal Point : 2 . 0

Leading Type : No Leading

Prefix Format : No Prefix

Link : MI 15 : JogSpeed

Figure 11 – Variable Configuration

3.3 Receiving Feedback

The net in Figure 12 exemplifies how to receive feedback through the socket from the Meca500.

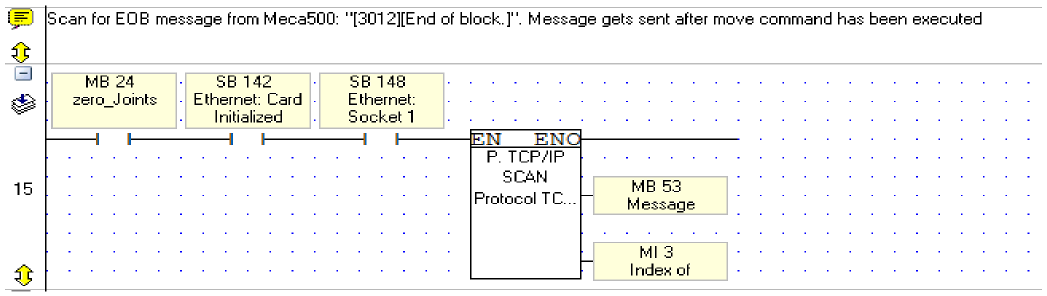


Figure 12 – Scan Protocol TCP/IP Block

If MB24 is set, the monitoring of the socket will be active as SB142 and SB 148 are there only to make sure that the communication was set correctly.

The scan block is used to scan for a *specified* message sent by the Meca500[3]. In this case, the message it is looking for is an EOB ('End of Block') message which indicates that there is no more move command in the robot buffer[4].

Should the scan block be successful it outputs a specified MI index value and sets a MB HIGH. Both of which can be used later in the program to activate other functions or perform other tasks.

3.4 Receiving variables

To receive variables, such as positions or robot status messages, you will need to use the stream type of variable since the messages are variable in both length and format. It is recommended that you place the scanning block for those variables in the main routine. Also, the Unitronics controller exhibits some limitations in handling data. When too much information is received or if the order of the data changes, some message can be received as missing or incomplete.

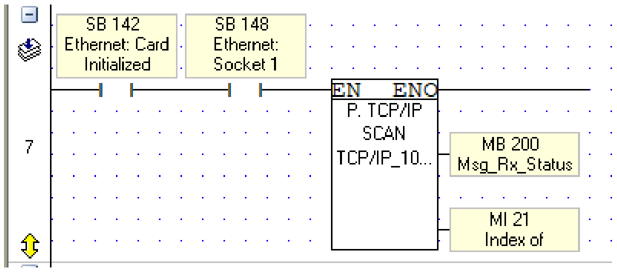
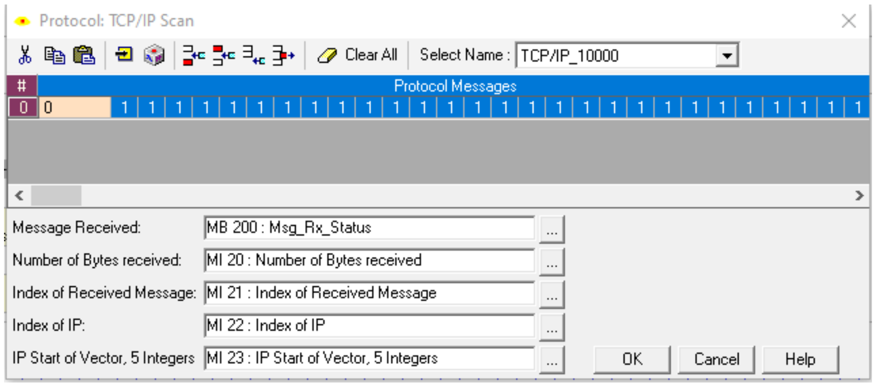


Figure 13 – Scan Block for Variables

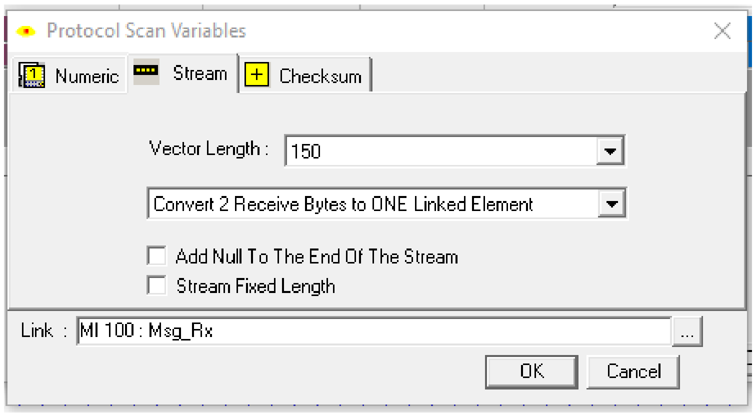
[3] The list of feedback messages the Meca500 outputs can be found in the programming manual.
[4] A null character must be added to the end of the message to be scanned for.

It is recommended that a lot of memory be reserved to handle long strings of data. Monitoring on port 10001 is challenging since there is no way to know what, when or how the data will be sent as the robot's feedback messages are only sent when the status is changed. Therefore, monitoring expected responses on port 10000 is far easier and should be sufficient for simple applications. In that case, reserve memory can be lowered significatively. Putting a timer on the scan block for the expected string would be recommended to avoid being stuck at waiting a message that was lost or not received in the right order.



Message Received: MB 200 : Msg_Rx_Status
Number of Bytes received: MI 20 : Number of Bytes received
Index of Received Message: MI 21 : Index of Received Message
Index of IP: MI 22 : Index of IP
IP Start of Vector, 5 Integers: MI 23 : IP Start of Vector, 5 Integers

Figure 14 – Stream Variable in a String



Vector Length : 150
Convert 2 Receive Bytes to ONE Linked Element
☐ Add Null To The End Of The Stream
☐ Stream Fixed Length
Link : MI 100 : Msg_Rx

Figure 15 – Stream variable Configuration

3.5 Jog Menu

Before using the jog menu, you should have already sent the 'ResetError', 'ActivateRobot' and 'Home' commands to the Meca500. These commands are the basic start commands needed to be executed by the robot prior to use.

You need to create 12 nets that will be exactly the same as the one shown in Figure 16, except for the trigger bit and the 'Send Message' bit.

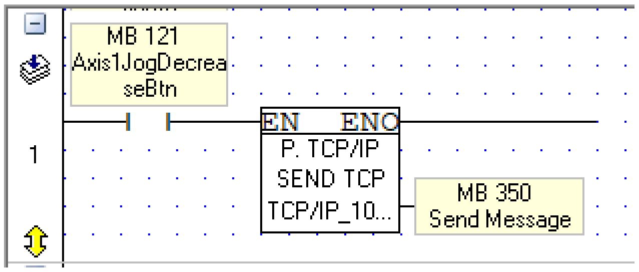


Figure 16 – Jog Net

It's important that each instance of the 'Protocol: TCP/IP Send' block have a different 'Send Message' bit in order to not trigger any subsequent dependent logic in your application and to facilitate debugging in 'Online Test' mode as well.

Put your "Axis1_Decrease" equivalent bit on a direct open contact and connect it to a 'Protocol: TCP/IP Send' block that sends a 'MoveJointsVel(-*11*,0,0,0,0,0)'[5] command.

Make 11 more of these nets, each corresponding to an increase or decrease button unique to each axis.

Holding the plus or minus buttons will send the same command multiple times. The culmination of which will be a continuous motion in one axis.

[5] Here, "11" is a variable and is used to send a variable speed.