

Mitsubishi Motion AOI Solutions
J4TM AOI User Manual

Version 2.000

This document applies to the Mitsubishi Electric Corporation product components and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

MITSUBISHI ELECTRIC CORPORATION PROVIDES THIS DOCUMENT "AS IS," WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

This document could contain technical inaccuracies or typographical errors. Changes are made periodically to the information herein. Mitsubishi Electric Corporation may make improvements and changes at any time to the product(s) and/or program(s) described in this document.

Table of Content

1. RSLOGIX VERSION	5
2. MANAGED SWITCHES.....	5
3. BEST PRACTICES	5
4. STATE AOIS	6
MMSO - MITSUBISHI MOTION AXIS SERVO ON	6
MMSF - MITSUBISHI MOTION AXIS SERVO OFF	6
MMASD - MITSUBISHI MOTION AXIS SHUTDOWN	7
MMASR - MITSUBISHI MOTION AXIS SHUTDOWN RESET	7
MMAFR - MITSUBISHI MOTION AXIS FAULT RESET	7
5. MOTION AOIS	8
MMAS - MITSUBISHI MOTION AXIS STOP	8
MMAJ - MITSUBISHI MOTION AXIS JOG	9
MMAH - MITSUBISHI MOTION AXIS HOME	11
MMAM - MITSUBISHI MOTION AXIS MOVE	12
MMAT - MITSUBISHI MOTION AXIS TORQUE.....	15
6. EVENT AOIS.....	16
MMAW - MITSUBISHI MOTION ARM WATCH & MMDW - MITSUBISHI MOTION DISARM WATCH	16
MMAR - MITSUBISHI MOTION ARM REGISTRATION & MMDR - MITSUBISHI MOTION DISARM REGISTRATION	20
7. CONFIGURATION AOIS	23
MMREADP - MITSUBISHI MOTION READ PARAMETER	23
MMWRITEP - MITSUBISHI MOTION WRITE RAM PARAMETER	23
MMWRITEE - MITSUBISHI MOTION WRITE EEPROM PARAMETERS.....	25
MMCFCG - MITSUBISHI MOTION AXIS CONFIGURATION	26
MMRE - MITSUBISHI MOTION READ ERROR.....	27
8. GEARING AOI	28
MMAG - MITSUBISHI MOTION AXIS GEARING	28
9. POINT TABLE AOI	31
MMAPT - MITSUBISHI MOTION POINT TABLE.	31
POINT TABLE UDT	32
COMMANDS	33
EXAMPLES	33
MMAPTEdit	35
10. SETTING IP ADDRESS	36
11. ROCKWELL PLC CONFIGURATION	39
COPY FILE.....	41
EDS FILES	42
12. IMPORTING AOIS.....	43
13. APPENDICES	46
ERROR CODES	46
CROSS REFERENCE OF CANOPEN OBJECT VS J4 PARAMETERS	48

LED & SERVO STATES	49
SERVO I/O INTERFACE CONNECTIONS	50
MASTER/SLAVE INTERFACE CONNECTIONS FOR GEARING	51
WRITE LOOP 1	52
WRITE LOOP 2	54
INITIALIZE	56
14. MAJOR REVISIONS.....	59
15. REFERENCES	59

1. RSLogix Version

These AOIs need version 20.03 or newer.

These AOIs do not use Motion Groups.

2. Managed Switches

From Rockwell Literature "Ethernet Design Considerations" Publication ENET-RM002C-EN-P - May 2013

As a general rule for unmanaged switches, make sure of the following:

- Your application does not contain I/O traffic or
- Your application has I/O control and the following is true:
 - o The network is not directly connected to the IT network
 - o All nodes on the network are Rockwell Automation devices
 - o There is no potential to overload a device with traffic

3. Best Practices

- Set an RPI between 1.0 ms and 20.0 ms. These AOIs are not embedded instructions. They rely on the RPI to maintain a viable servo update rate. Keep the interval between 1.0 and 20.0 ms. Anything longer and the AOIs may not work properly.
- Each AOI must have a unique instance. E.g. if you have multiple MMAM AOIs, each one must have its own instance, do no share instance names.
- All AOIs use the same Mitsubishi_Servo UDT for that axis. E.g. if you have 2 axes, you should only have 2 Servo UDTs, one for each axis. (An Allen Bradley UDT = Mitsubishi SDT). A single AOI uses only one servo UDT.
- Using an EDS file will allow you to monitor real-time communication between the AOIs and the servos. If you use a generic connection, you will not notice if communication is lost until an AOI fails to execute.
- If communication is lost and regained, some AOIs need to be rebooted (e.g. MMAG re-programs registers for the gearing function during first scan). Either reboot the PLC or go from Run mode to Program mode and back or Initialize the AOIs.
- NOTE: Servo firmware versions before and including B1 do not recognize Merge in the MMAM.
- Do not set a loop continuously reading parameters using MMReadP. If you want to monitor servo settings such as parameters, inputs and outputs - use Explicit messaging (see Appendix). Explicit reading of any servo error thru MMRE can be triggered by Misubishi_Servo.Control bit3 (servo error) or bit7 (servo warning).
- Do not execute MMRE, MMWriteR, MMreaP and MMWriteE at the same time.

4. State AOIs

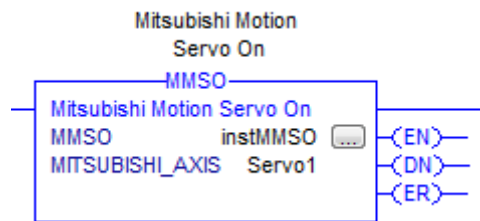
- **True/False or Rising Edge** means the AOI can start with either a true rung or a one-shot. Once started, the logic will continue whether the rung is true or false.
- **Rising Edge** means it will execute on the rising edge and then lock out. There is no continuing logic when the rung is false. With a latched bit, the rung will have to go false, then true to re-trigger the AOI.
- **Latched True** means the AOI only executes when the rung is true.

MMSO - Mitsubishi Motion Axis Servo On

Use the Motion Servo On (MMSO) instruction to enable the servo amplifier

DN on MMSO and MMSF shows state of servo based on servo enabled state. If servo is enabled, MMSO.DN is on at all times.

- MMSO is **Rising Edge**. It enables the drive when the rung makes a false-to-true transition



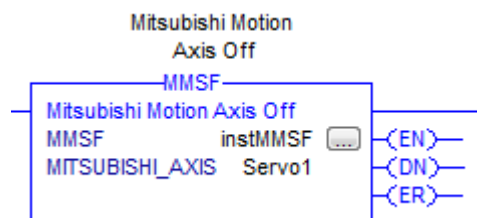
See [Servo States](#)

MMSF - Mitsubishi Motion Axis Servo Off

Use the Motion Servo Off (MMSF) instruction to deactivate the drive output for the specified axis.

DN on MMSO and MMSF shows state of servo based on servo enabled. If servo is not enabled, MMSF.DN is on at all times.

- MMSO is **Rising Edge**. It disables the drive when the rung makes a false-to-true transition

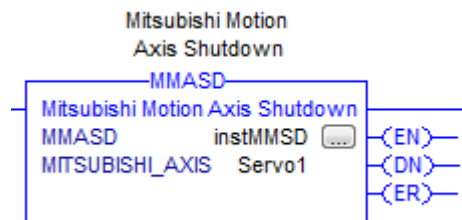


MMASD - Mitsubishi Motion Axis Shutdown

Use the Motion Axis Shutdown (MMASD) instruction to force a specified axis into the Shutdown state. The Shutdown state of an axis is the condition where the drive output is disabled. The axis remains in the Shutdown state until either an Axis or Group Shutdown Reset is executed.

- MMASD is **Rising Edge**. It disables the drive when the rung makes a false-to-true transition

See [Servo States](#)

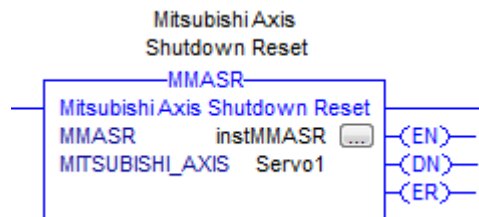


MMASR - Mitsubishi Motion Axis Shutdown Reset

Use the Motion Axis Shutdown Reset (MMASR) instruction to transition an axis from an existing Shutdown state to an Axis Ready state. All faults associated with the specified axis are automatically cleared.

- MMASR is **Rising Edge**. It clears the motion axis shutdown state when the rung makes a false-to-true transition

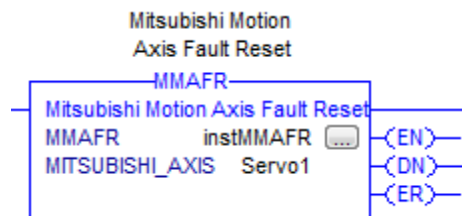
See [Servo States](#)



MMAFR - Mitsubishi Motion Axis Fault Reset

Use the Motion Axis Fault Reset (MMAFR) instruction to clear all motion Errors and warning for an axis. This is the only method for clearing axis motion errors.

- MMAFR is **Rising Edge**. It reset the error when the rung makes a false-to-true transition



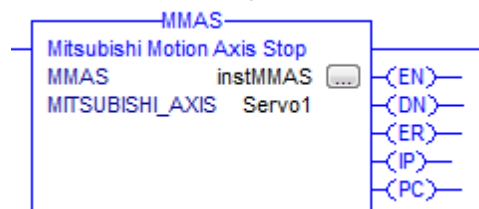
5. Motion AOIs

- **True/False or Rising Edge** means the AOI can start with either a latched true rung or a one-shot. Once started, the logic will continue whether the rung is true or false.
- **Latched True** means the AOI only executes when the rung is true.

MMAS - Mitsubishi Motion Axis Stop

Use the Motion Axis Stop (MAMS) instruction to stop a specific motion process on an axis or to stop the axis completely.

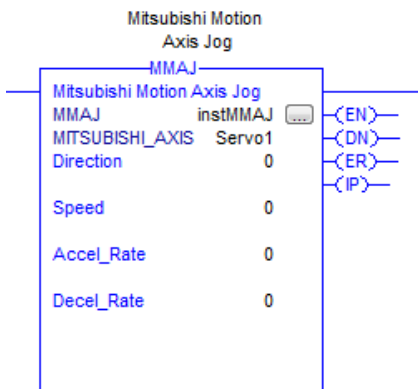
- **MMAS** (Motion Axis Stop) is **Latched True**. Once set, the axis cannot move.
- If you use it with a one-shot, it will stop motion but any motion command can start the servo again.
- When latched and active, the servo does not move and ignores all motion commands,
- If you try to move with **MMAS** active - Error = 1017.
- **MMAS** stops motion and unconditionally clears the target buffer.
 - If you want to be able to resume after stopping an MMAM move, use absolute moves.
- If you trigger a stop when servo is not enabled, you do not get an alarm.



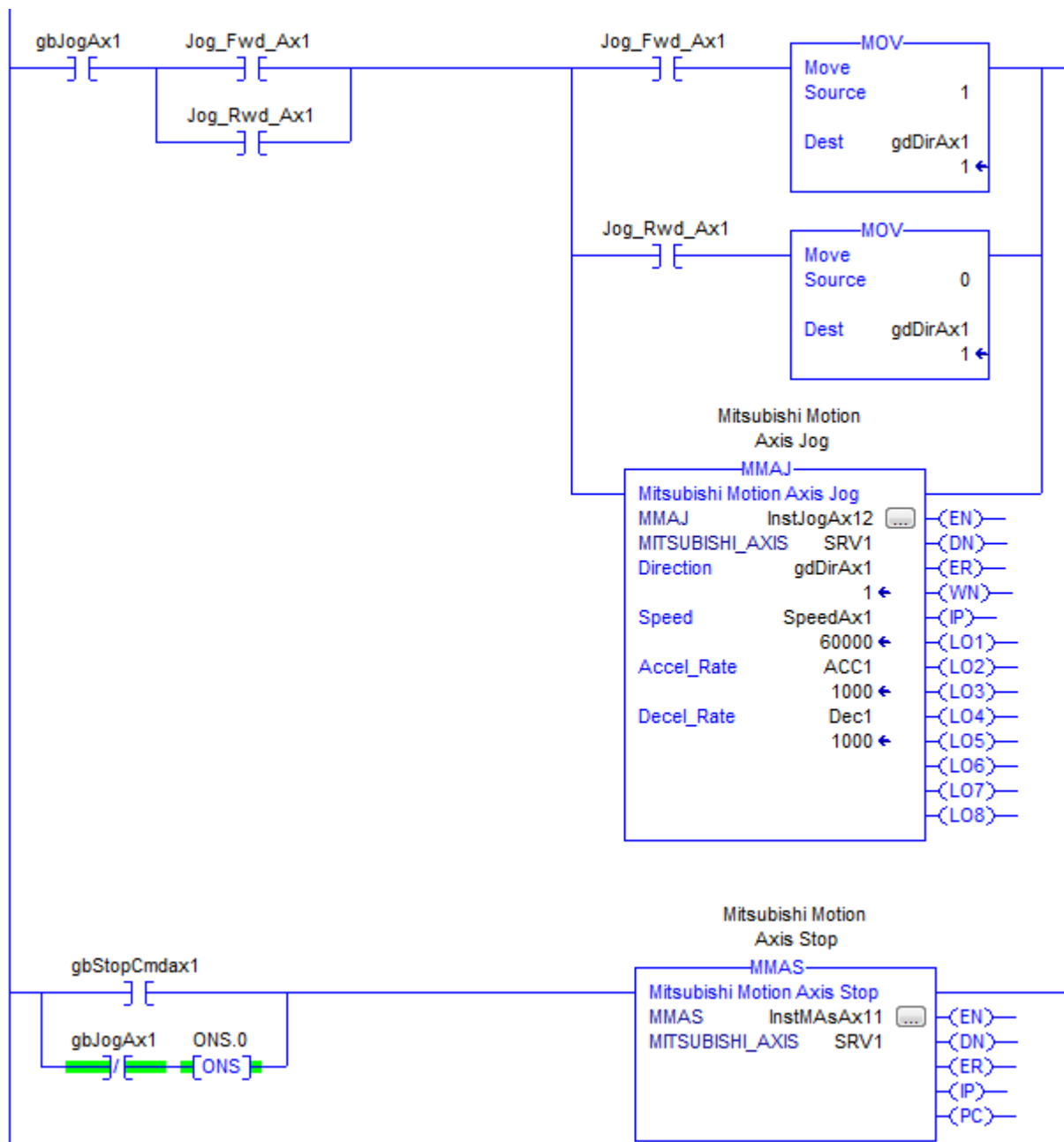
MMAJ - Mitsubishi Motion Axis Jog

Use the Motion Axis Jog (MMAJ) instruction to move an axis at a constant speed until execute MMAS AOI.

- MMAJ (Motion Axis Jog) is **True/False or Rising Edge**.
- Direction is 0 or 1.
- Speed (unit rpm x0.01) cannot be negative
- Multiple Moves - During a jog, a second jog command will merge immediately and will not change the outputs (DN, IP) of the first AOI instant.
- Accel_Rate is value in ms to reach the rated speed
- Decel_Rate is value in ms from rated speed to zero speed.
- To stop, use **MMAS** Mitsubishi Motion Axis Stop.



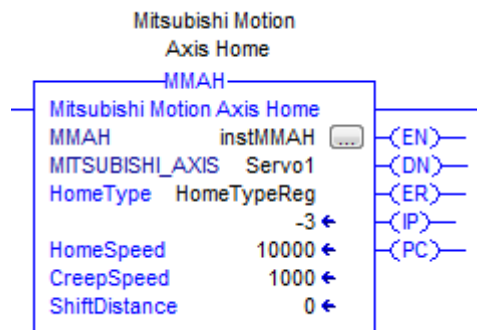
Note once started, Jog will run continuously regardless of the rung state. If you want to control jog thru a pushbutton, program the MMAJ to execute with a oneshot or as long as the button is pressed. Then to stop, program a rung with an MMAS triggered by a one shot when you let go of the button.



MMAH – Mitsubishi Motion Axis Home

Use the Motion Axis Home (MMAH) instruction to home an axis.

- MMAH - Homing is **True/False or Rising Edge**
- Direction is based on Home type



It has the **MMWriteR** AOI embedded inside it for Parameter Mapping – HomeType, HomeSpeed and CreepSpeed.

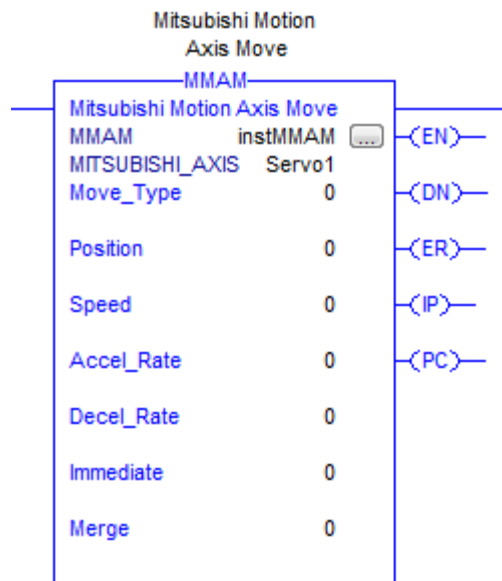
Parameter	Description	Comment
HomeType	Used in all homing routines	See details in user manual
HomeSpeed	Used in all homing routines	unit rpm x 0.01 cannot be negative
CreepSpeed	Used in all homing routines	unit rpm x 0.01 cannot be negative
ShiftDistance	After homing, the servo moves to this shift position and zeros the position register	Polarity has no effect. Absolute value is loaded into J4. (Not implemented yet)

For detailed description of all homing commands, MR-J4-TM SERVO AMPLIFIER INSTRUCTION MANUAL (EtherNet/IP) SH030226

MMAM – Mitsubishi Motion Axis Move

Use the Motion Axis Move (MMAM) instruction to move an axis to a specified position.

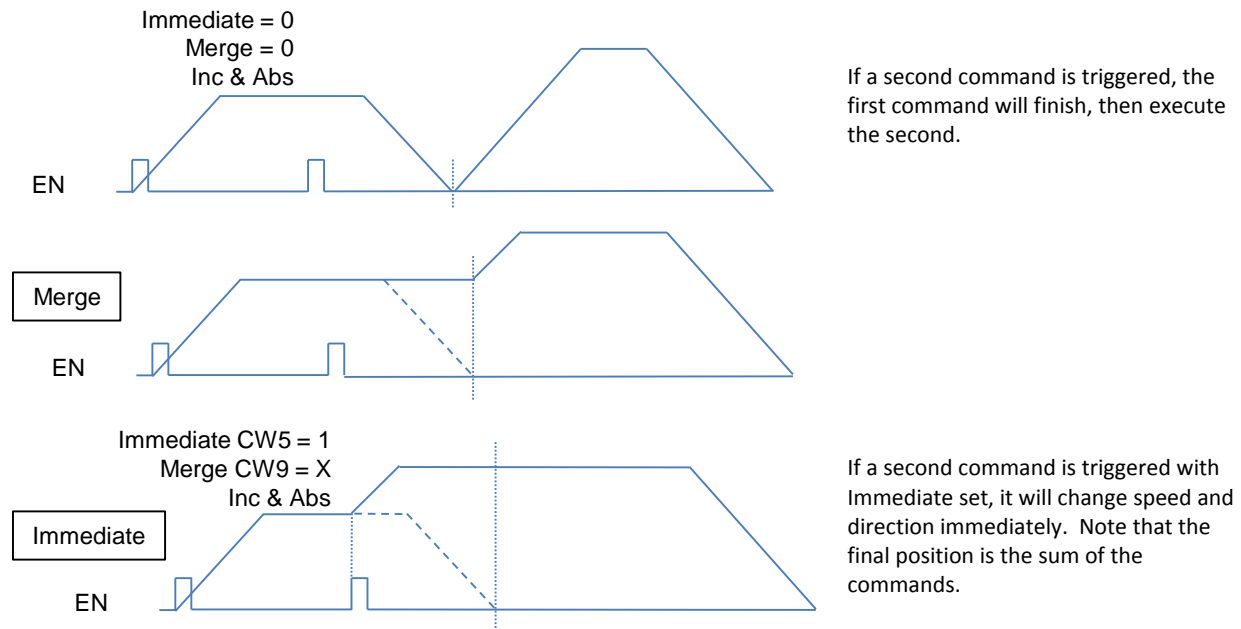
- MMAM (Motion Axis Move) is **True/False or Rising Edge**.
- Move Type: 0 → Absolute and 1 → Incremental
- Position(pulse): Absolute system → Position to move and Incremental system → Distance to move (polarity determines direction)
- Position command range -999999 to 999999
- Speed (rpm x 0.01) cannot be negative
- Multiple Moves using Merge and Immediate
- Accel_Rate is value in ms to reach the rated speed
- Decel_Rate is value in ms from rated speed to zero speed.



NOTE: Servo firmware versions before and including B1 do not have Merge in the MMAM.

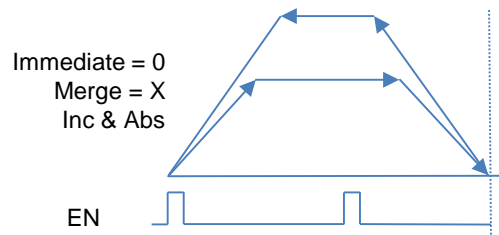
You can merge/blend a second move command using Immediate and Merge inputs. However, note the distance of the second command is always added to the distance of the first command.

MMAM

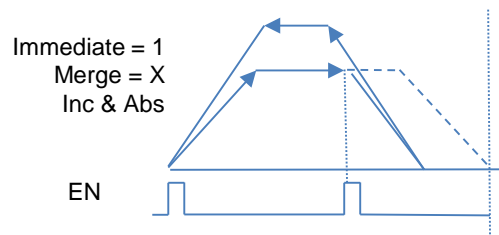


MMAM

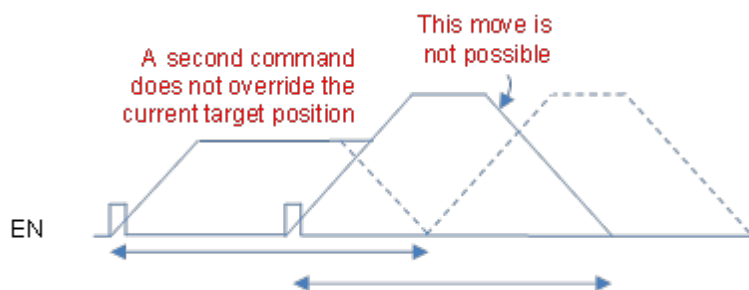
Second move is same distance but negative



If a second command is triggered before the first is complete, the first command will finish, then execute the second.



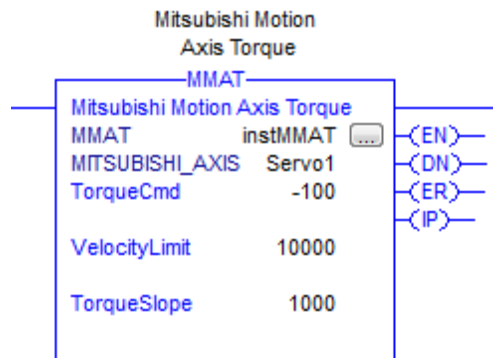
If a second command is triggered with Immediate set before the first is complete, it will change speed and direction immediately. Note that the final position is the sum of the commands.



MMAT - Mitsubishi Motion Axis Torque

Use the Motion Axis Torque (MMAT) instruction to move an axis at a constant torque until you tell it to stop.

- MMAT (Motion Axis Torque) is **True/False or Rising Edge**.
- **VelocityLimit** cannot be negative.
- Direction is determined by TorqueCmd polarity.
- **TorqueCmd** is in units of 0.1% of Rated Torque (1000 => 100% of Rated Torque) **TorqueSlope** is Rated Torque/10/sec
- To stop, use MMAS Mitsubishi Motion Axis Stop.
- Note that stopping torque move with MMAS will leave the servo with 0 torque and will not hold position.

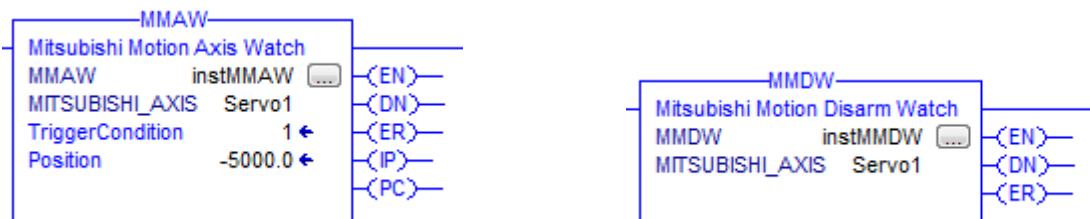


6. Event AOIs

- **True/False or Rising Edge** means the AOI can start with either a true rung or a one-shot. Once started, the logic will continue whether the rung is true or false.
- **Rising Edge** means it will execute on the rising edge and then lock out. There is no continuing logic when the rung is false
- **Latched True** means the AOI only executes when the rung is true.

MMAW - Mitsubishi Motion Arm Watch & MMDW - Mitsubishi Motion Disarm Watch

- **MMAW** is True/False or Rising Edge
- **MMDW** is Rising Edge



- **TriggerCondition** –
 - 0 = Forward - the servo module looks for the actual position to change from less than the watch position to greater than the watch position.
 - 1 = Reverse – the servo module looks for the actual position to change from greater than the watch position to less than the watch position.
- **Position** that triggers PC output
- **PC** - will go high if servo moves in correct direction from outside to inside the position.

If **TriggerCondition** is 0 and actual position is greater than **Position**, **PC** is set. It is reset by another rising edge trigger to MMAW or MMDW.

If **TriggerCondition** is 1 and actual position is less than **Position**, **PC** is set. It is reset by another rising edge trigger to MMAW or MMDW.

In the MITSUBISHI_SERVO UDT:

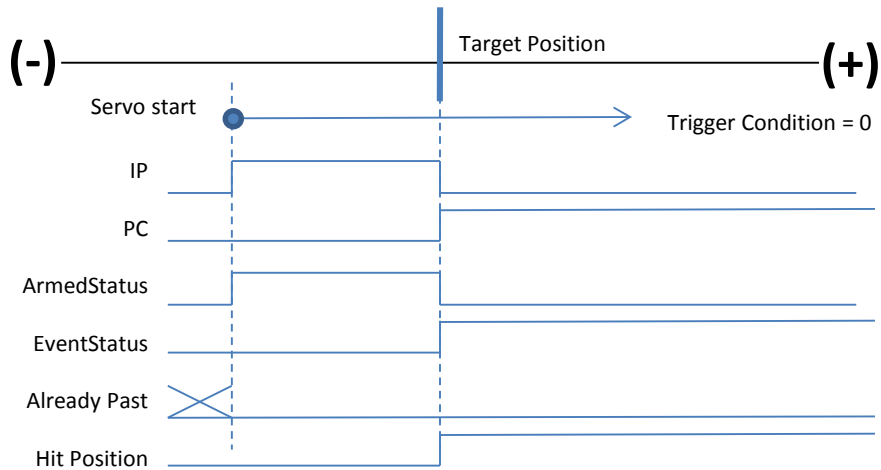
- **WatchEventArmedStatus** is a Boolean set if the Watch logic is active. Reset when event occurs or MMDW.
- **WatchEventStatus** is a Boolean flag set when event occurs. It is reset by another rising edge trigger to MMAW or MMDW.

1. If **TriggerCondition** is 0 and actual position reaches greater than **Position**, **PC** is set. It is reset by another rising edge trigger to MMAW or MMDW.
2. If **TriggerCondition** is 1 and actual position reaches less than **Position**, **PC** is set. It is reset by another rising edge trigger to MMAW or MMDW.

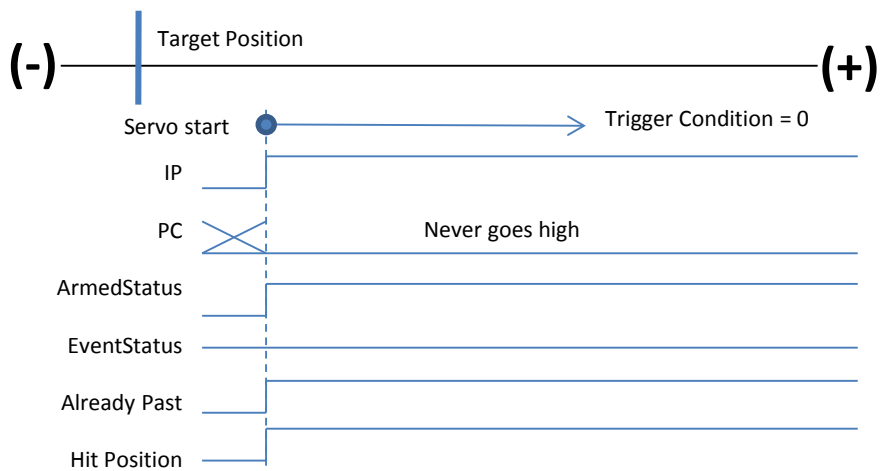
3. If the AOI is executed with servo ALREADY past the programmed position (heading away from the programmed position), it will set the MMAW instance dot operator - <instMMAW>. **AlreadyPast** and **PC** will never be set.

To reset the watch you have to re-trigger MMAW. MMDW only turns off the watch and clears the outputs of the MMAW, it does not re-trigger anything.

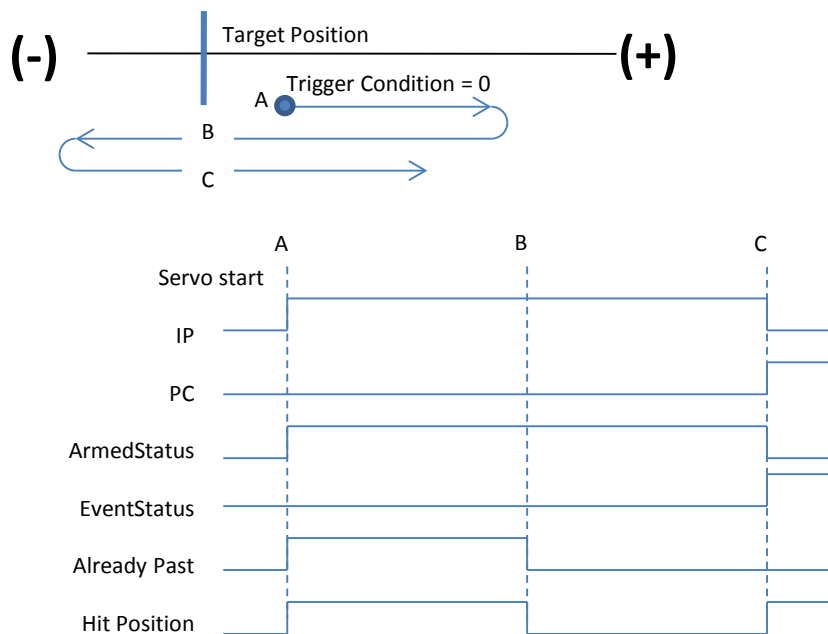
Example 1



Example 2



Example 3



MMAR - Mitsubishi Motion Arm Registration & MMDR - Mitsubishi Motion Disarm Registration

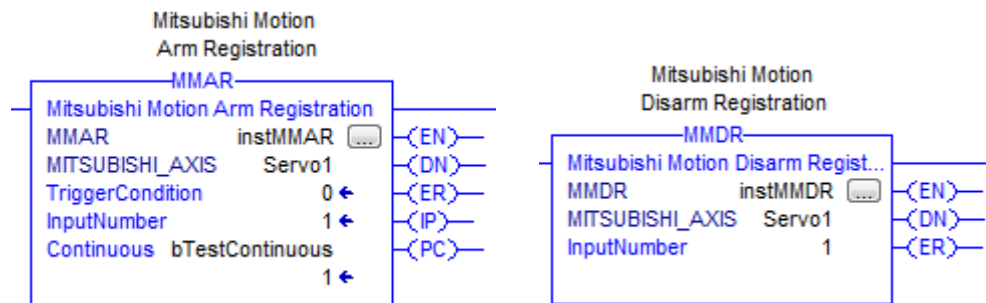
- MMAR is **True/False** or **Rising Edge**
- MMDR is **Rising Edge**
- If you use non continuous (= 0) you need to Retrigger the rung to reset the PC outputs.
- If you change any setting, you need to re-initialize the AOI. <MMAR.Initialize>

Cycle the PLC at least once or trigger <MMAR instance>.Initialize after adding MMAR to the program or changing any setting. MMAR reprograms an internal register for Registration functions.

When the Touch Probe input is asserted, the actual position at that moment is captured and stored in Registration registers.

The Touch Probe inputs are in CN3 pins 1 and 10.

The captured positions are in the AOI instance – x.**A_TouchProbeValue** and x.**B_TouchProbeValue**.



Input	Range	Description
Trigger Condition	= 0 trigger on positive edge = 1 trigger on negative edge	
InputNumber	0, 1 or 2	Which probe to use. 0 = disable that channel
Continuous	= 0 Once captured, AOI must be toggled to re-arm = 1 AOI will automatically re-arm after every probe input assertion.	If AOI is not Continuous , PC (Process Complete) will come on with the first probe input assertion. PC will stay on and AOI will not re-arm until AOI enable is toggled. If AOI is set Continuous , PC is a one-shot for each probe input assertion.

RegEventArmedStatus in the MITSUBISHI_SERVO UDT is set when the AOI is enabled in Logic. It is reset when...

- after a capture if not in Continuous mode.
- with MMDR
- with an error.

RegEventStatus in SDT goes low when the AOI is enabled. It goes high when a capture occurs. It is reset when...

- RegEventArmedStatus is toggled

Output	Continuous	Description
A_PC B_PC	= 0 Latched	Latched when position is captured. Stays set until the AOI is toggled or initialized
	= 1 Continuous update	Pulsed output for 1 scan only when position is captured.

There are two probe inputs 1 & 2. These are independent of the two Channels A & B. You assign the inputs to any channel.

Example 1

Channel A is Touch Probe input 2, falling edge and continuous.

Channel B is Touch Probe input 2, rising edge and continuous.

Example 2

Channel A is Touch Probe input 1, rising edge and continuous.

Channel B is Touch Probe input 2, rising edge and continuous.

Example 3

Channel A is Touch Probe disabled (= 0).

Channel B is Touch Probe input 2, rising edge and latched.

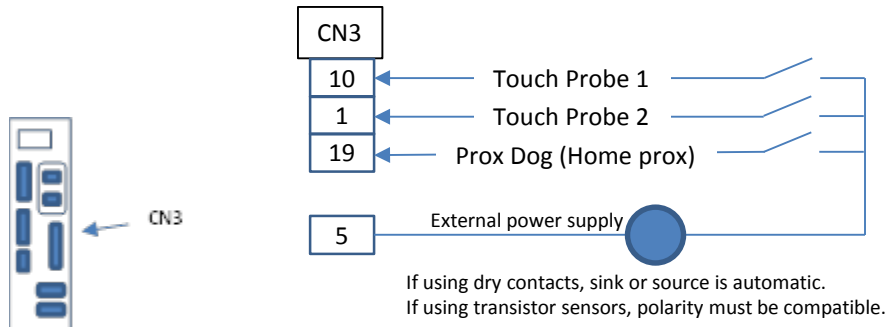
Example 4

Channel A is Touch Probe input 1, rising edge and ***latched***.

Channel B is Touch Probe input 1, falling edge and ***continuous*** <<< NOT ALLOWED.

You cannot have both latched and continuous on the same input.

The **MMAR** AOI will not have window registration. The registration function in the servo works all the time with no limitations. Rather than embed windowing code inside the AOI, it is left to the user to write their own windowing logic if desired.



7. Configuration AOIs

- **True/False or Rising Edge** means the AOI can start with either a true rung or a one-shot. Once started, the logic will continue whether the rung is true or false.
- **Rising Edge** means it will execute on the rising edge and then lock out. There is no continuing logic when the rung is false
- **Latched True** means the AOI only executes when the rung is true.

MMReadP - Mitsubishi Motion Read Parameter

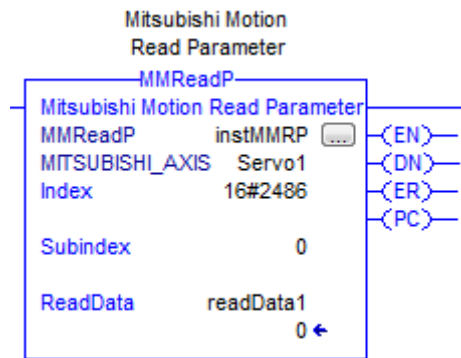
- MMReadP is **True/False or Rising Edge**

This is a single read AOI that uses CANOpen objects.

To convert J4 parameter addresses to CANOpen see [Cross Reference of CANOpen Parameters](#)

E.g. using the mapping Cross Reference, Creep Speed PT06 converts to 16#2486h sub-index 0.

The original address would be CANOpen register 6099h sub index 2.



MMWriteR - Mitsubishi Motion Write RAM Parameter

- MMWriteR is **True/False or Rising Edge**

This is an auxiliary AOI for writing parameters with the J4 Servo. MMWriteR writes to the RAM memory. When you cycle power, these values are not retained.

To permanently load into memory EEPROM, see [MMWriteE](#).

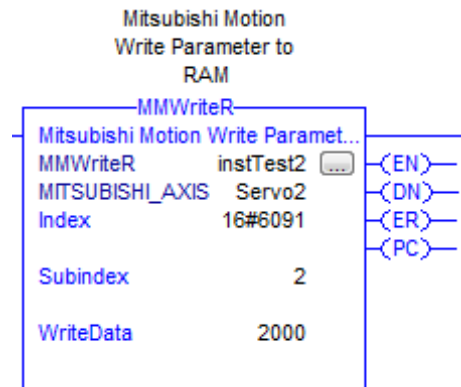
You can program a loop to do multiple writes. See [Write Loop examples](#) in Tables & Lists

This is a single write AOI that uses CANOpen addresses.

This example writes to CANOpen register 6091h sub index 2 which is electronic gear denominator (scaling).

To convert J4 parameter addresses to CANOpen see [Cross Reference of CANOpen Parameters](#).

Using the mapping Cross Reference PA07 converts to 16#2007h sub-index 0.

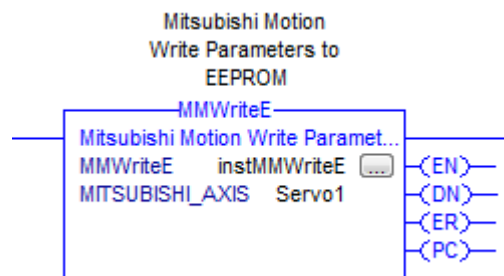


MMWriteE - Mitsubishi Motion Write EEPROM Parameters

- MMWriteE is True/False or Rising Edge
- **You do not enter parameters in this AOI.** It will save what is already in RAM at the time of execution.

MMWriteE writes all the RAM settings to EEPROM.

MMWriteR only writes to RAM – when the servo reboots, the values in the EEPROM load to servo parameters



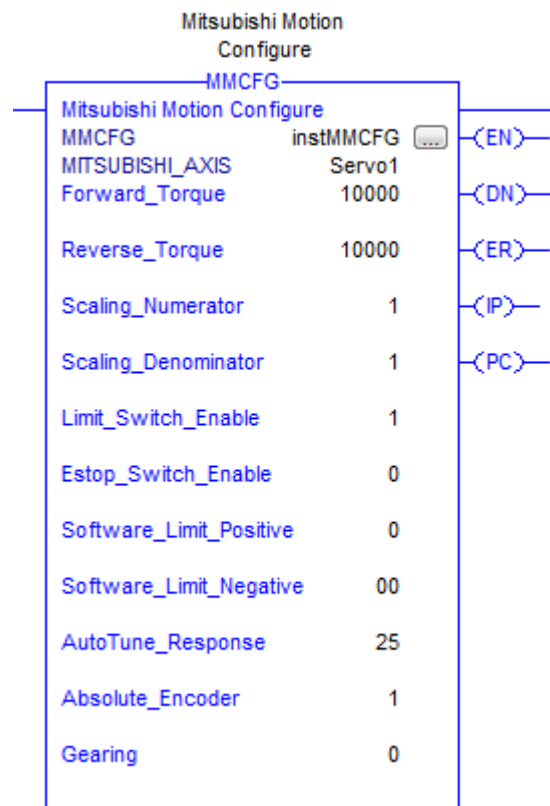
Note: This can take up to 10 seconds - it stores all the parameters from RAM to EEPROM. Do not turn off power to recycle power for at least 10 seconds

MMCFG – Mitsubishi Motion Axis Configuration

- **MMCFG** is True/False or Rising Edge
- An embedded **MMWriteE** is automatically executed.
 - Note: This can take up to 10 seconds - it stores all the parameters from RAM to EEPROM.
Do not turn off power to recycle power for at least 10 seconds
- You must cycle power to the servo after executing

These are the most common user settings.

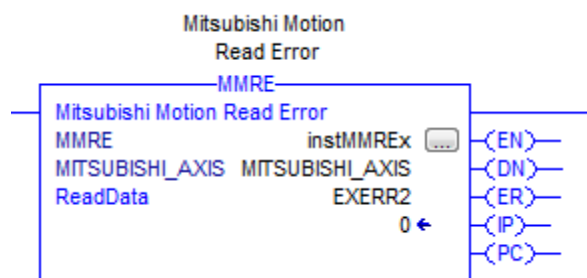
- Torque –
 - -1 = maintains existing value
 - 0 – 10,000 (1000.0%)
- Scaling Numerator/Denominator –
 - -1 = maintains existing value
 - 0 to 16777215
- Limit Switch Enable –
 - -1 = maintains existing value
 - 0 = disabled, no limit switches (0C00)
 - 1 = Positive limit switch on (0800)
 - 2 = Negative limit switch on (0400)
 - 3 = Both limit switches on (0000)
- Estop Switch Enable –
 - -1 = maintains existing value
 - 0 = disabled, no Estop switch (2100)
 - 1 = Estop switch on, no deceleration (0000)
 - 2 = Estop switch on, deceleration (2000)
- Software Limit –
 - Full range of DINT
 - If Positive value = Negative value, maintains existing values
- AutoTune Response –
 - -1 = maintains existing value
 - Range is 1 to 40
- Absolute Encoder –
 - -1 = maintains existing value
 - 0 = no battery backup
 - 1 = battery backup
- Gearing (**slave servo only**) –
 - -1 = maintains existing value
 - 0 = no gearing
 - 1 = gearing (**only use on slave servo**)



Recycle Power after executing this AOI and PC is on.

- MMRE - Mitsubishi Motion Read Error**

In the event of a Servo Error, MMRE will output the servo error into EXERR and the parent AOI will move Error Code 1020 into ERROR.



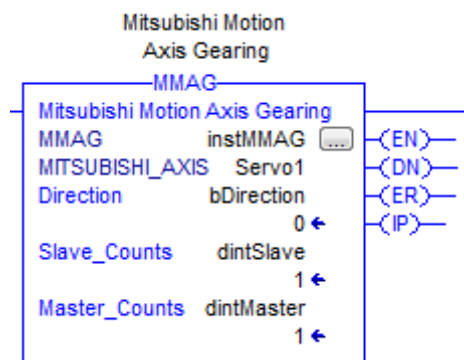
8. Gearing AOI

MMAG - Mitsubishi Motion Axis Gearing

The slave axis will follow the master axis. The master encoder counts are ported to the slave.

- *All MMAG commands are for the slave (geared) axis.* You do not program anything for the master servo.
- **MMAG is Latched True.** When active, the slave will follow. When not active the slave drops out.
- **Direction** = 0 slave moves in same direction. **Direction** = 1 slave moves in opposite direction.
- Slave Counts/Master Counts

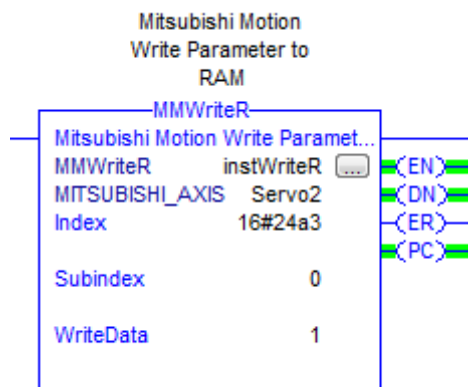
Cycle the PLC at least once after adding MMAG to the program. MMAG reprograms an internal register during first scan for Gearing functions.



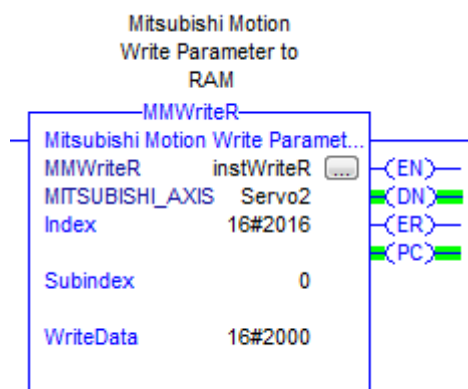
To enable Gearing, you can either use MMCFG or you can manually set the parameters yourself.

Manually, you enter 1 in PT35 and 2000 in PA22 for the slave axis. You can do this using MR Configurator or by using MMWriter.

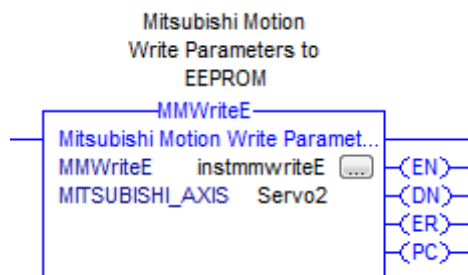
Using the mapping to J4 parameters, PT35 = 24A3h



Using the mapping to J4 parameters, PA22 = 2016h



After you write both values to RAM, write them to EEPROM, then **cycle power to the servo**.



For cabling between Master and Slave, see Appendices.

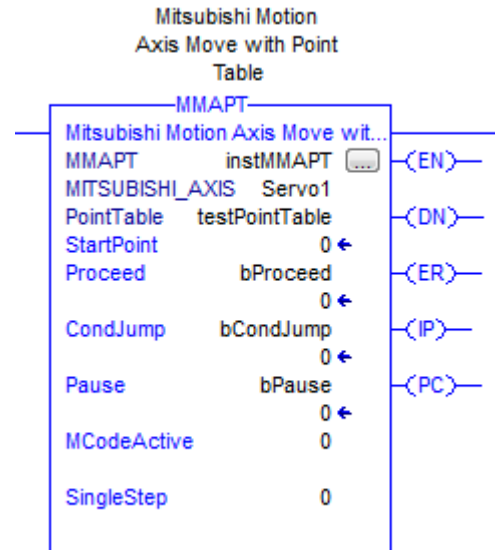
Remember, only the slave gets the parameters. If you try to enable gearing on the any axes you will get error 70.3 on the axes if the gearing cable is not there.

The Servo instance will be in Gearmode when MMAG AOI is true (EN = 1). If you drop the PLC out of run mode while gearing (e.g. go to program mode), then back to run mode, the Servo itself will drop out of gearing but the AOI will still be in Gearmode. You will have to toggle the AOI to reset the servo functionality. Also, if you try to jog or move on the slave axis after toggling run mode, you will get an error that you are still in Gearmode until you turn off the MMAG rung.

9. Point Table AOI

MMAPT - Mitsubishi Motion Point Table.

- **MMAPT** is Latched True



Point tables are an array of simple moves. They are not merged or blended. At this time, the point tables are set at 20. An index consists of an absolute or relative move, with a command at the end of the move (continue, timer, jump, wait, etc).

The MMAPT is not transitional, i.e. if not active, the point table stops at the end of the current move and command.

If you turn off the rung, the index will stop at the end of the index (after move and command). However if you re-activate it, the Point table resets.

- If you want to pause, use the Pause input
- If you want to single step, use the Pause then Single Step input.

Inputs

- **StartPoint** is which index to start with, typically 0.
- **Proceed** is an input used with 3 functions
 - To override **TimerOverride (4)** command. If Proceed goes high during the timer, it will stop the timer and move to the next index. If Proceed does not go high during the timer, the timer will time out normally and move to the next index.
 - To override **Wait (5)** command.
 - To override **Pause** Input. **Pause** will stop the table at any time. You resume by turning **Proceed** on (If **Pause** is not active). This needs a rising edge. If **Proceed** is already on before a **Pause**, it will need to be toggled. (An MMAS will clear the buffers, this does not)

Proceed a rising edge trigger.

- **CondJump** is an input used for a Conditional Jump command. If active it will jump to the index loaded in Value. If not active, the next index will be executed. **CondJump** is active as long as it is true.
- If **Pause** turns on during a move, it will stop immediately. When the Pause input goes low, and **Proceed** is toggled, the command will finish and the table will continue.

Remember that some commands are single scan operations.

- **MCodeActive** is an input to specify if the M code is loaded at the beginning (=0) or the end (=1) of the index, after both position and command. If loaded at the end, the initial index MCode value will be 0.
- **SingleStep** is an input that causes the AOI to stop at the end of each index (move and command). You press Proceed to execute the next index.
 - If SingleStep is set before executing MMAPT, you will have to toggle Proceed to execute the first index.
 - If SingleStep is set before the end of an index (move and command), the index will stop at the end.
 - If SingleStep is reset before the end of an index, the index will still stop at the end. Proceed will clear it.
 - If the command is a 5 = Wait until Proceed, a single Proceed will go to the next index.

If you turn off the rung, the index will stop at the end of the move and command. However if you re-activate it, the Point table resets.

Outputs

- **Mcode** is a user defined number. It is loaded into the Mcode register at the beginning or end of the move depending on the **MCodeActive** input. It is a DINT so you can put -2,147,483,648 to +2,147,483,647. You can have duplicates.
- **CurrentIndex** shows the index number that is active.
- Transitions can be monitored by with the MCodes.

Point Table UDT

The Point Table uses an array of UDTs.

TypeMove	DINT	Decimal	0=Absolute 1=Incremental 2=No Move	Read/Write
PositionData	DINT	Decimal		Read/Write
SpeedData	DINT	Decimal		Read/Write
AccelTime	DINT	Decimal		Read/Write
DecelTime	DINT	Decimal		Read/Write
M_Code	DINT	Decimal	Active at start or end of step (depends on MCodeActive input)	Read/Write
Command	DINT	Decimal	Commands are executed at the end of a move	Read/Write
Value	DINT	Decimal	Depends on the command	Read/Write

- If you want the line to be non-motion, set **TypeMove** = 2, the **Command** will still execute.
- If you want the line to be a complete NOP, set **TypeMove** = 2 and the **Command** = 1 (continue).
- **Command** and its **Value** are active at the end of a move.

- **Mcode** is simply a user selected number to identify each point. You can match two different point tables even if the actual indexes are not the same. You can have duplicates. This number is loaded into the output register at the start of the step. It is a DINT so you can put -2,147,483,648 to +2,147,483,647.
- **Wait** until **Proceed** Input is active at the end of a move, then execute next index.
- **Timer Override** is a Timer that can be overridden by the **Proceed** input. If the **Proceed** input is not toggled, the timer will time out normally and the next index will execute.
- **Jump** will jump to the index in **Value**
- **ConditionalJump** will jump to the index in Value if the **CondJump** input is high. If not, it will go to the next index.

Commands

Command		Value	Description
0	End	<don't care>	Servo will stop – end of point table
1	Continue	<don't care>	Execute next step
2	Timer	Timer duration in milliseconds	Start timer at end of move
3	Jump	Destination index	Jump to index at end of move
4	TimerOverride (Proceed)	Timer duration in milliseconds	Start timer at end of move until timer times out or Proceed is active
5	Wait until Proceed	<don't care>	Stop at end of move and wait for Proceed input
6	ConditionalJump	if CondJump = 1, Destination index if CondJump = 0, <don't care>	At the end of the move, if CondJump input = 1 jump to destination index if CondJump input = 0 move to next index

Examples

(MCodeActive = 0)

Index	M_code	TypeMove	Position	Speed	Command	Value
3	4	0 (abs)	1000	10000	1 (cont)	<don't care>

The servo will output an Mcode of 4, do an absolute move to position 1000 at a speed of 10000. Then it will continue to next index.

(MCodeActive = 0)

Index	M_code	TypeMove	Position	Speed	Command	Value
5	12	1 (rel)	1500	20000	2 (timer)	3000

The servo will output an Mcode of 12, do an incremental move of 1500 counts at a speed of 20000, then wait 3 seconds before executing the next step.

(MCodeActive = 1)

Index	TypeMove	Position	Speed	Command	Value	M_code
3	0 (abs)	21000	10000	6 (condjump)	10	4
4	0 (abs)	30000	20000	1 (cont)	<don't care>	5

10	1 (rel)	2000	20000	1 (cont)	<don't care>	17

The servo will do an absolute move to 21000 at a speed of 10000. When it arrives it will output an Mcode of 4, then perform a conditional jump. If the CondJump input is high, the next point will be index 10. If the CondJump input is low, index 4 will be executed.

(MCodeActive = 1)

Index	TypeMove	Position	Speed	Command	Value	M_code
15	0 (abs)	34000	20000	5 (wait)	<don't care>	11

The servo will do an absolute move to 34000 counts at a speed of 20000, then will stop and wait until the Proceed input goes high. When the Proceed input goes high, it will output an Mcode of 11 and continue to the next index.

(MCodeActive = 1)

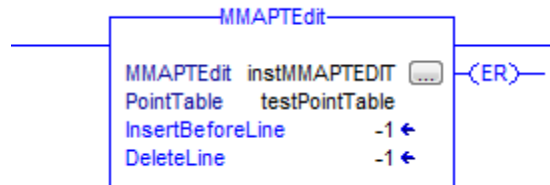
Index	TypeMove	Position	Speed	Command	Value	M_code
15	0 (abs)	34000	20000	1 (cont)	<don't care>	11
16	3 (no move)	<don't care>	<don't care>	2 (timer)	5000	12

The servo will do an absolute move to 34000 counts at a speed of 20000. When it arrives, it will output an MCode of 11. The next index is a non-motion index and will simply start a timer for 5 seconds. When it times out, it will output 12 and move to the next index.

MMAPTEdit

MMAPTEdit is an editing tool for inserting lines or deleting lines in a point table. Otherwise you will have to copy and paste.

- MMAPTEdit is Rising Edge only



Save after an edit.

- AOI is rising edge, to avoid multiple executions. You must toggle off to reset it.
- Delete or Insert function disabled by (-1)
- If both inputs are (-1) nothing is done
- For a 30 element array, ranges are 0 to 29.
- If **both** inputs have values of 0 or greater, you will get an error.
- If any input value is 30 or greater, you will get an error.

DeleteLine

- After a valid Delete, the Delete line number is changed to -1 for safety.
- If you delete a line, any jump label (Command 3 or 6) greater than DeleteLine will be decremented
 - If you delete a line, and there is a jump to the old line, the jump line number will not be modified.

E.g. Index 8 has a jump to index 3. If you delete index 3, the jump from index 8 still goes to index 3

InsertBeforeLine

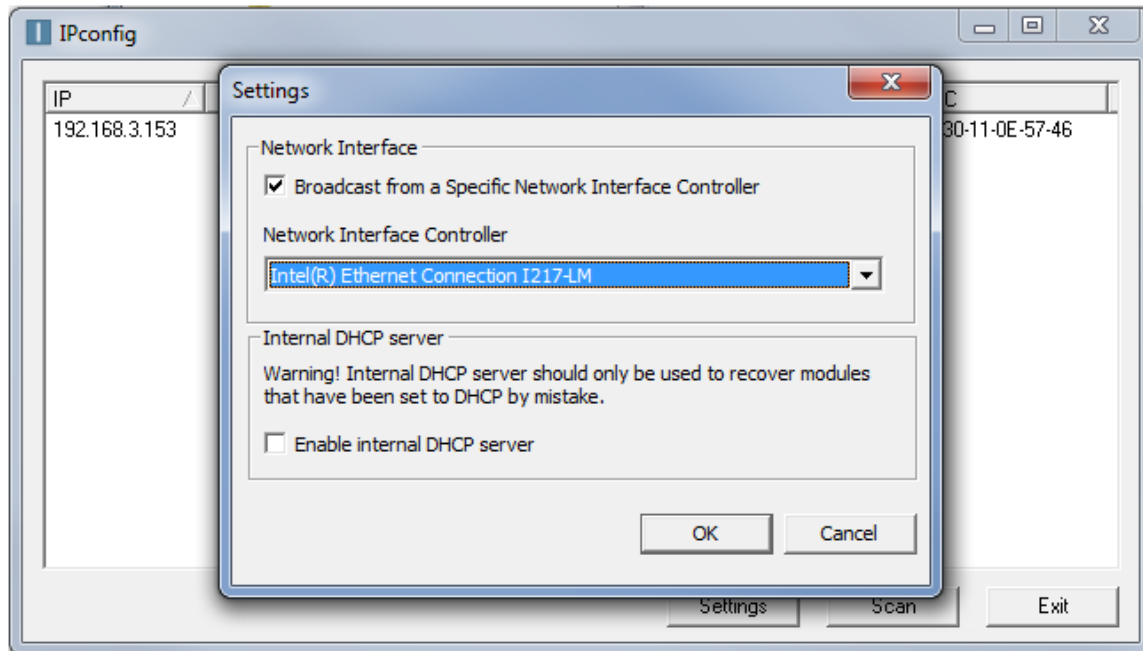
- After a valid Insert, the Insert line number is changed to -1 for safety.
- An INSERT will insert a “Do Nothing” index with a MoveType of 2 (no motion), a Command of 1 (continue), an MCode of -12345. All other elements will be 0.
- E.g. If you insert before 5, the new “Do Nothing” index will be 5. The original index 5 will then be 6.
- After a valid Insert, the last index of the array is lost.
- If you insert a line, any jump label (Command 3 or 6) greater than or equal to InsertBeforeLine will be incremented
 - If you insert a line, and there is a jump to the first moved line, the jump line number will be modified.

E.g. Index 8 has a jump to index 3. If you insert a line before index 3, the jump from index 8 will go to index 4.

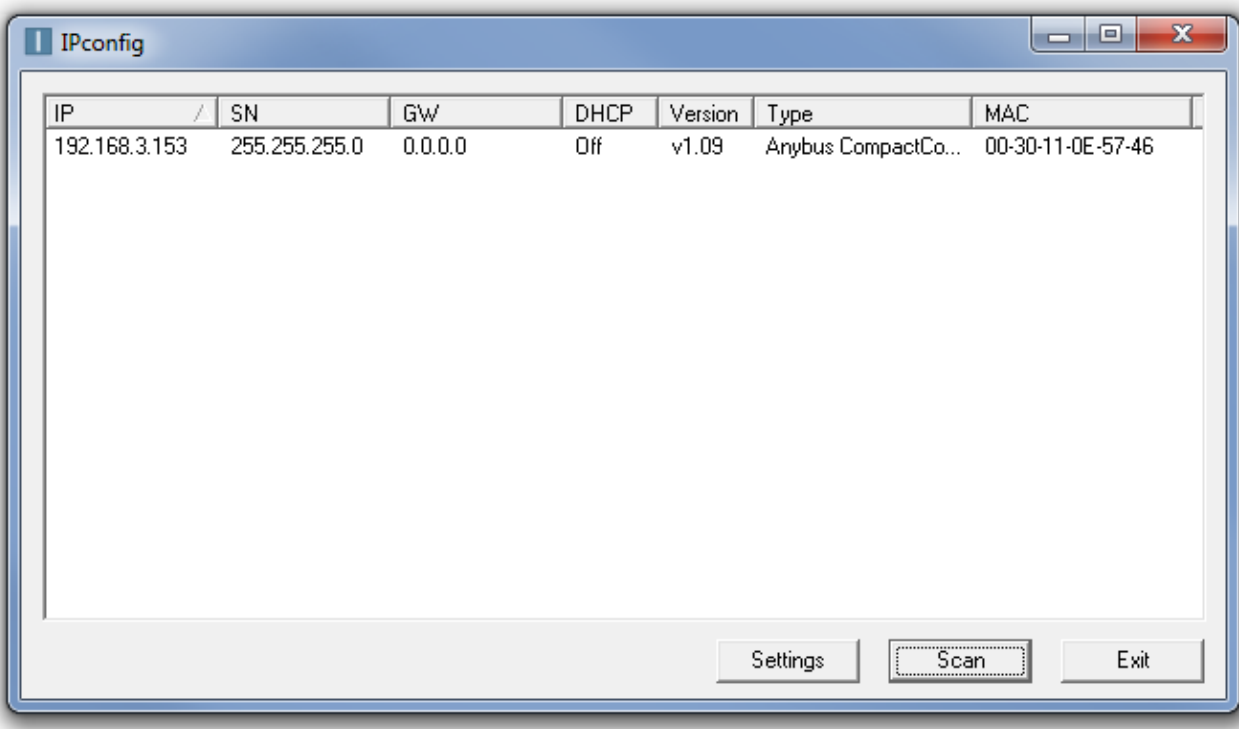
10. Setting IP Address

To change the IP address of the servo amplifier Anybus module, use IPConfig tools from HMS Anybus

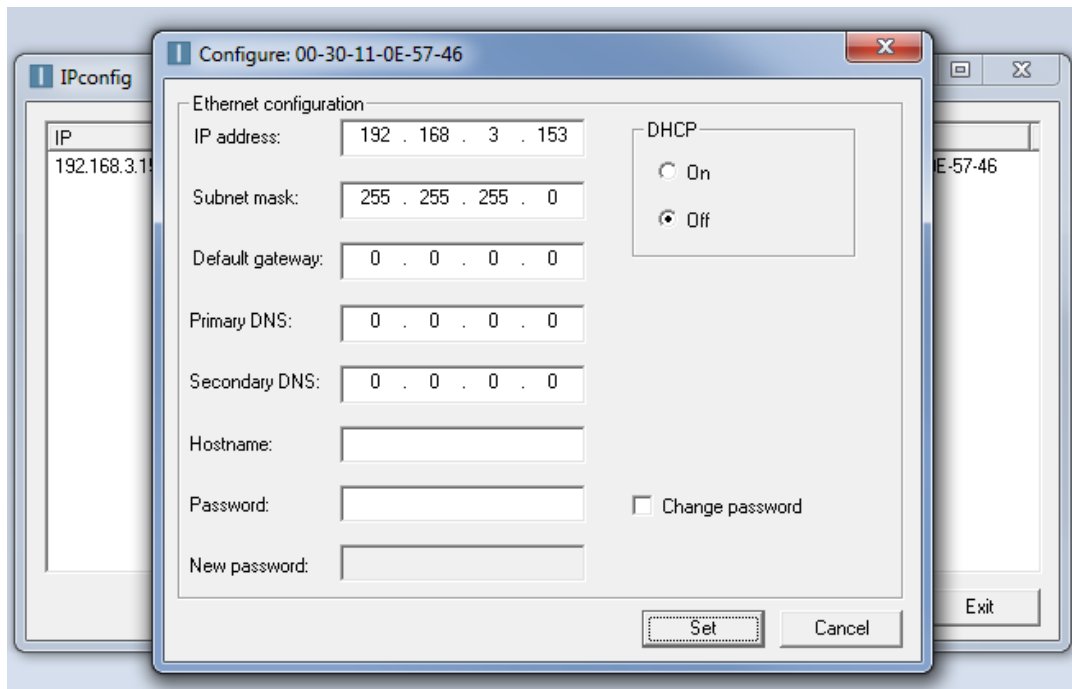
- 1) Launch the **Ipconfig** tools
- 2) Click on settings and make sure you are not wireless.



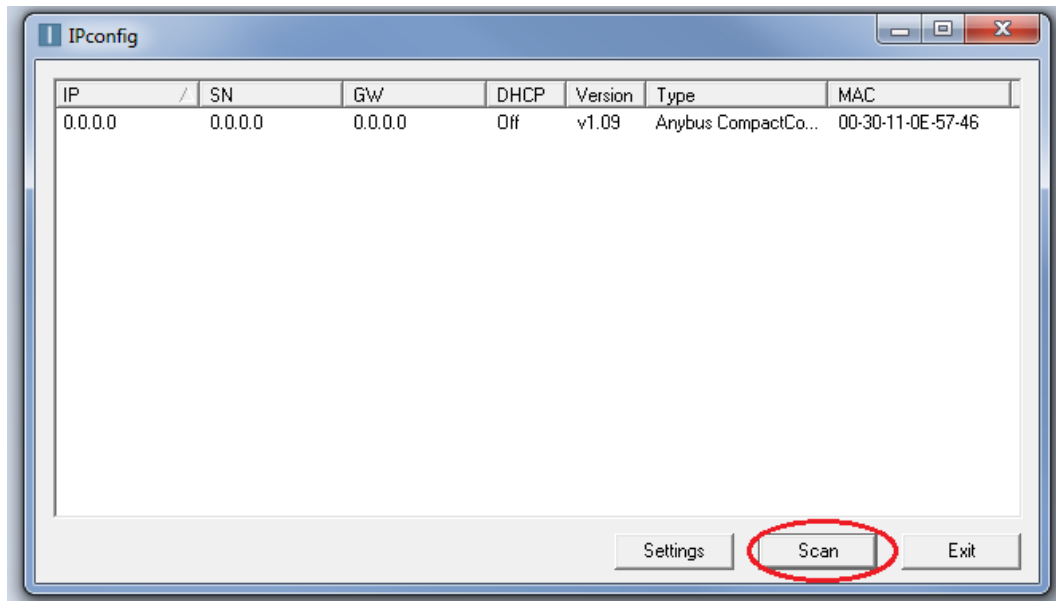
- 3) Click on **Scan** to find Anybus module. It will find all Anybus modules regardless of octet value.



- 4) To change setting, first double click on the entry to get to settings and Change the IP address here.



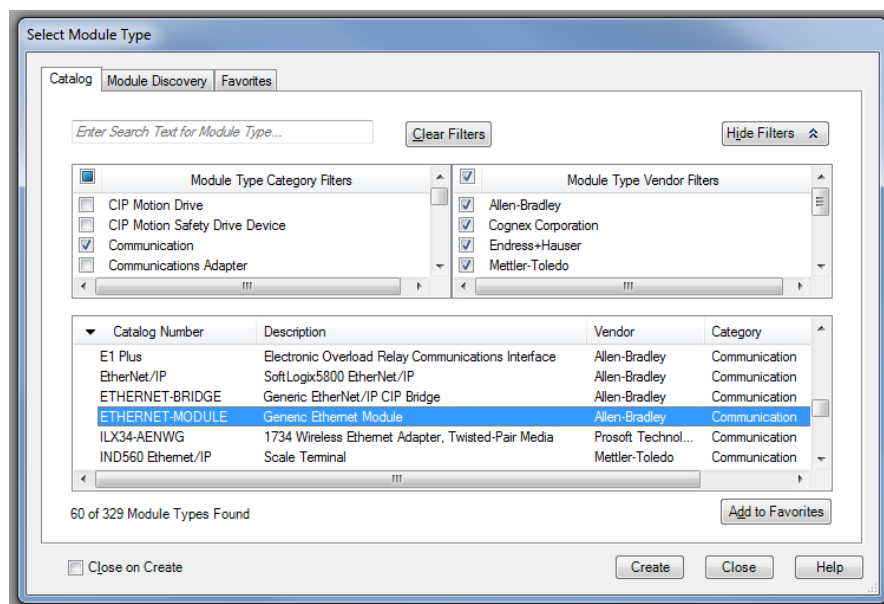
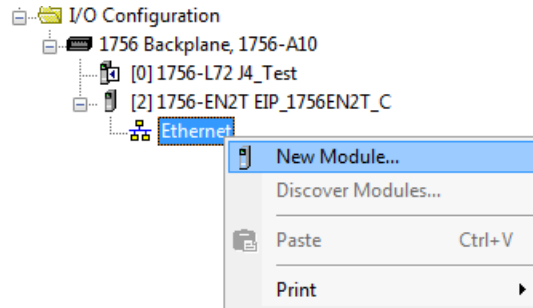
- 5) If you change anything, entry will go blank, click on **Scan** again to verify change



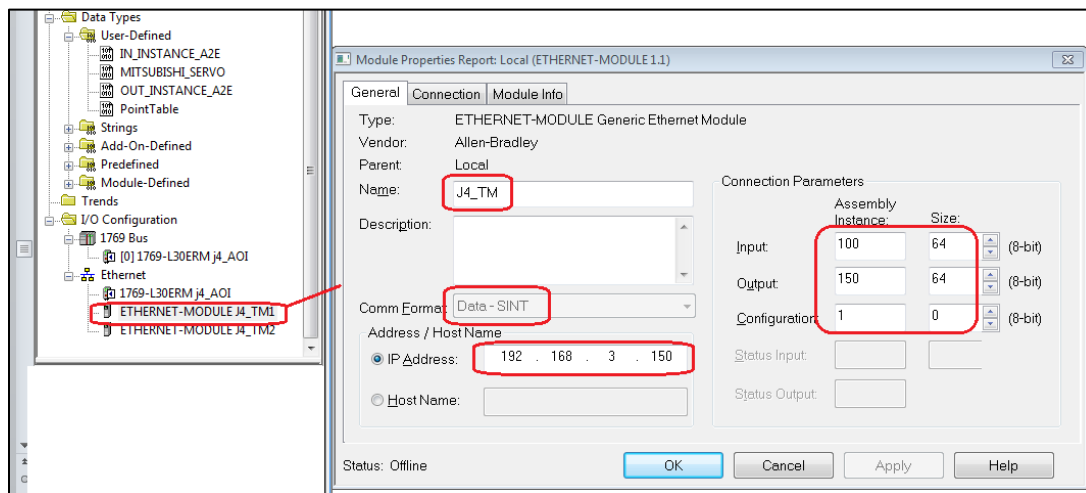
11. Rockwell PLC configuration

The configuration steps for a CompactLogix project are described this section. These steps are used to communicate with a servo amplifier. It is assumed that the user has basic knowledge in using RSLogix5000 software to perform the basic configuration steps.

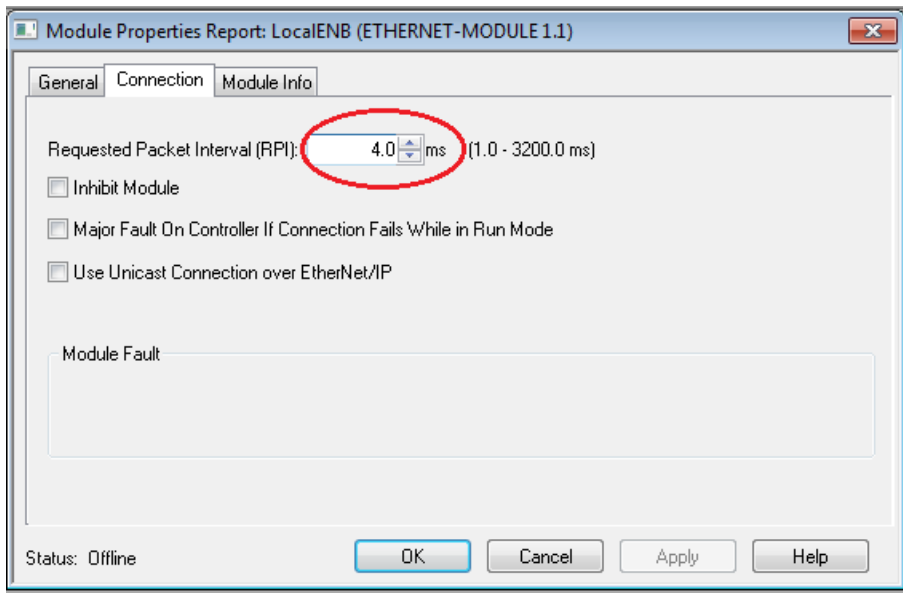
- 1) Create a new project in the RSLogix5000 using the proper revision level as the ControlLogix controller. In this example, the revision level is 20.
- 2) Under I/O configuration right click on the 1756 EN2T and Ethernet icon and and choose "New Module..."



- 3) In the "Select Module" pop-up window, choose the "Communications" and then select "Generic Ethernet Module".
- 4) Enter the proper Name, Communication Format, IP Address and Connection Parameters and the Input and Output instances with their sizes. Be sure to specify the Comm Format is a SINT.



5) The second tab, make sure the RPI interval is between 1.0 – 20.0



These AOIs are not embedded instructions. They rely on the RPI to maintain a viable servo update rate. Keep the interval between 1.0 and 20.0 ms.

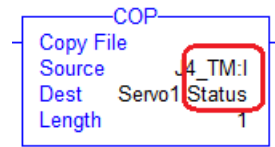
If the RPI interval is too long, the AOIs may not function properly.

COPY File

At the top of the program file put the COP (read from <servo>:I) instruction. Use the Generic Ethernet module as the source and put the file into the MITSUBISHI_SERVO UDT in the field **Status**.

For this example:

the Generic Ethernet module is **J4_TM**

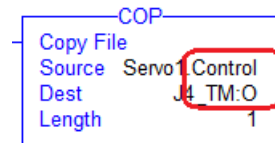


The Mitsubishi servo UDT is **Servo1** and you must have Servo1.**Status**

At the bottom of the program file, put the COP (write to <servo>:O) instruction. Use the MITSUBISHI_SERVO UDT field **Control** and write it to the Generic Ethernet Module.

For this example:

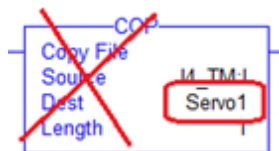
the Generic Ethernet module is **J4_TM**



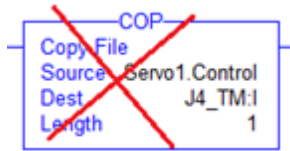
The Mitsubishi servo UDT is **Servo1** and you must have Servo1.**Control**

Common Errors

Note Servo1 does not have xxx.Status



Note here, the Control word (which is an **output**) is incorrectly matched with the module **Input**



EDS Files

Refer to the Technical Bulletin in the KnowledgeBase

[“TB-How to use the EDS file of MR-J4-TM in RSLogix 5000 platform”](#)

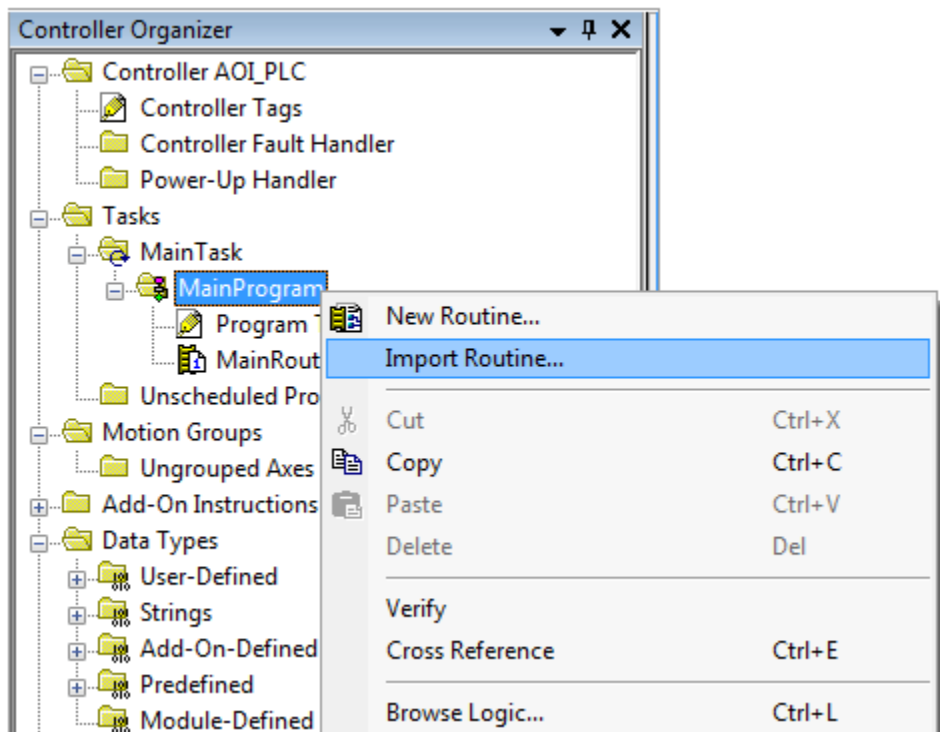
12. Importing AOIs

There are 2 ways to import the AOIs:

- Import all of them at once using Library_X.L5X
- Import them individually from the AOI ZIP file.

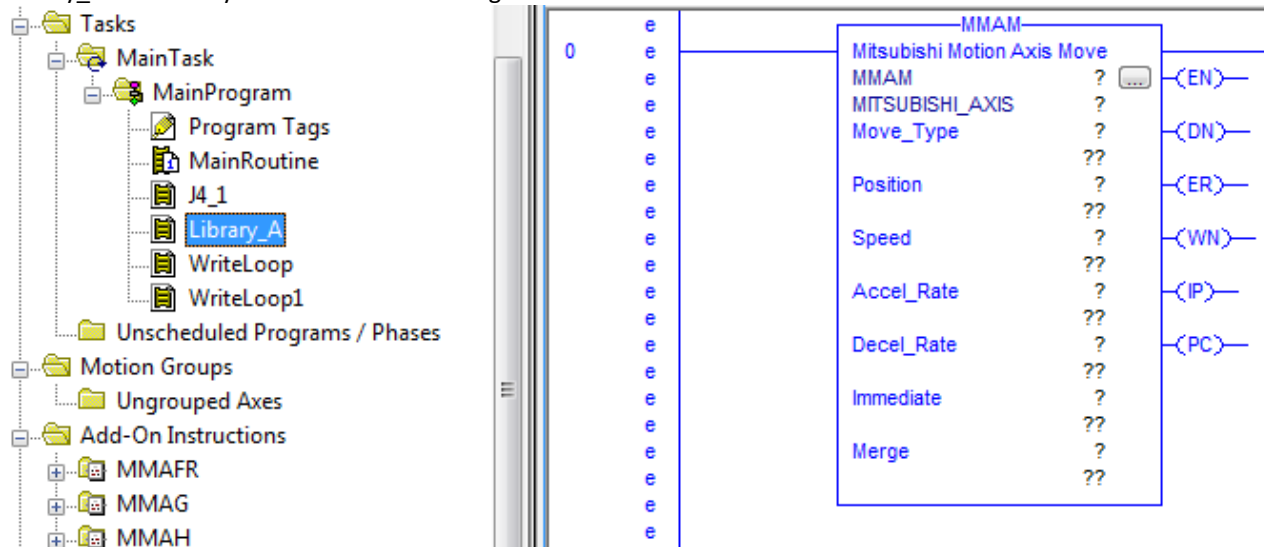
Library_X.L5X

This is a dummy routine that has all the AOIs on several rungs. The AOIs are not configured; they have no instances and no values. When you import this library, it loads the AOIs in the tree and also loads all the support files you will need.

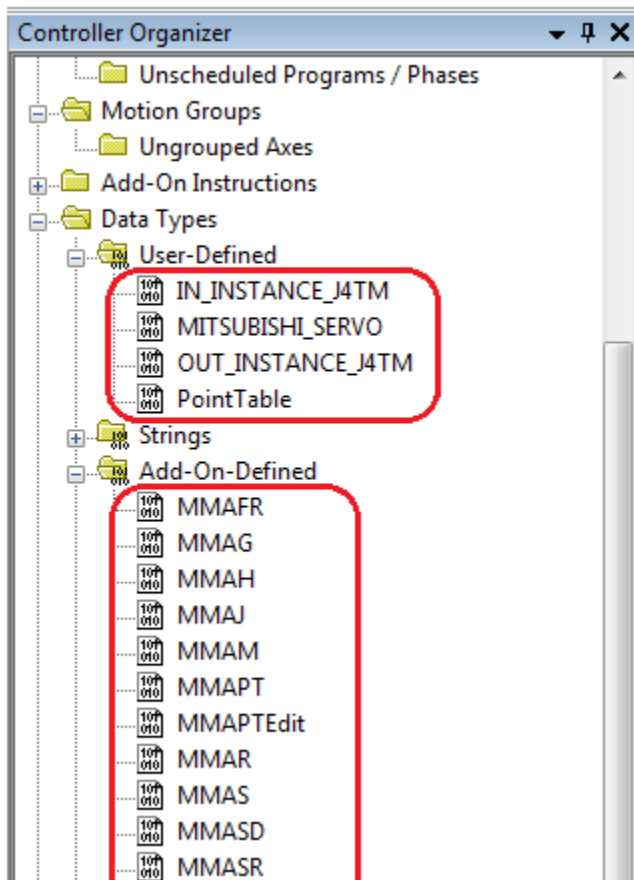


Import Routine, NOT the Import Task.

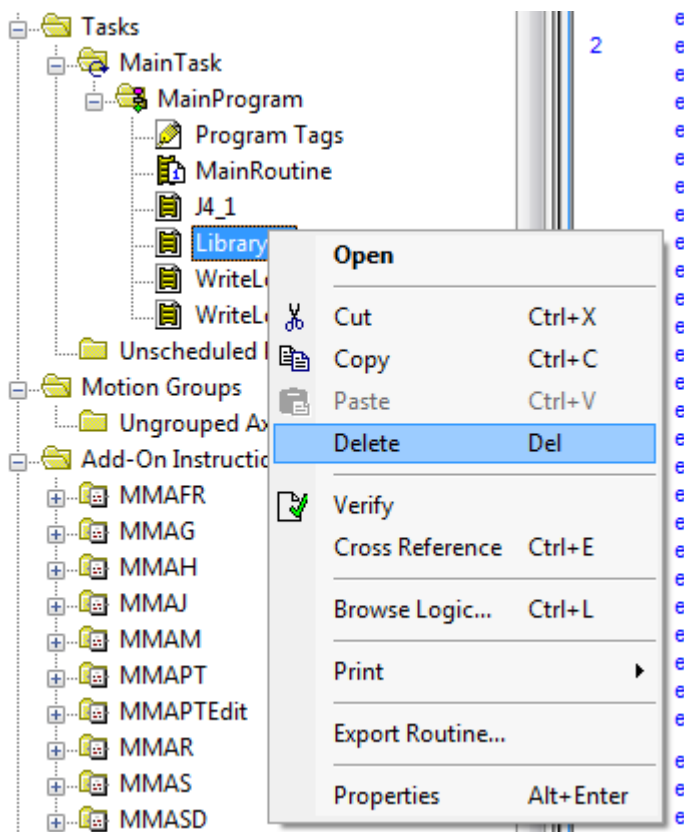
Library_X is a dummy file with all the unconfigured AOIs.



When it is imported, it brings all the support files as well as the AOIs.



Once imported, you can delete Library subroutine. The Add-On Instructions will remain.



13. Appendices

Error Codes

Error codes have Extended Errors EXERR to further define the fault.

Error Code	EXERR	Description
1013	0	Invalid Axis
	1	Speed out of range (MMAM) -999999 <=> 999999
	2	Invalid Move Type – Absolute/Incremental (MMAM, MMAPT)
	3	Invalid Position
	4	Invalid Speed out of range (MMAJ, MMAM, MMAH)
	5	Invalid Acceleration rate (MMAJ, MMAM)
	6	Invalid Deceleration rate (MMAJ, MMAM)
	7	Invalid Home Type (MMAH)
	8	Invalid Torque Slope
	9	Invalid Torque Limit (MMAT)
	10	Invalid Speed = 0 (MMAM, MMAJ)
	11	Invalid Home Speed (MMAH)
	12	Invalid Home Creep Speed (MMAH)
	13	Invalid Parameter Group Invalid (MMWP, MMRP)
	14	Invalid Speed cannot be negative (MMAT)
	15	Invalid Parameter Invalid (MMWP) E.g. < 0
	16	Invalid input number (MMAR)
	17	Invalid AutoTune Response (MMCFCG)
	18	Invalid Scaling Numerator (MMCFCG)
	19	Invalid Scaling Denominator (MMCFCG)
	20	Invalid Limit Switch Enable (MMCFCG)
	21	Invalid Estop Switch Enable (MMCFCG)
	22	
	23	
	24	Invalid Forward Torque Limit (MMCFCG)
	25	Invalid Reverse Torque Limit (MMCFCG)
	26	A single probe cannot be both continuous and latched (MMAR)
	27	Invalid Start Point (MMAPT)
	28	Invalid Velocity Limit (MMAT)
	29	Triggered MMAR while MMDR is active
	30	Invalid Gearing value - must be -1,0,1 (MMCFCG)
	31	Invalid Absolute Encoder value - must be -1,0,1 – (MMCFCG)

Error Code	EXERR	Description
1014	AOI ID ¹	Not Homed (MMAM, MMAPT, MMAW)
1015	<error code from servo>	Servo in Warning condition SW.7
1016	AOI ID ¹	Cannot change modes unless servo is stopped/done e.g. attempt MMAM if in Jog (EXERR = 1) e.g. attempt MMAJ if Homing (EXERR = 4) e.g. attempt MMAH if in Gearing (EXERR = 6)
1017	AOI ID ¹	MMAS is on, cannot try to move anything.
1018	AOI ID ¹	Cannot trigger gearing if in JogMode, TorqueMode or Homing. EXERR is what mode you are in
1020	<error code from servo>	Servo in Error condition SW.3
1021	AOI ID ¹	Servo is disabled and a move instruction was attempted – servo may be shutdown, not enabled
1022	<Which parameter>	EXERR value 16#10 (16) = Illegal code number. • Check Parameter InstructionCode2 E.g. You wrote “6083” instead of “16#6083”, Subindex out of range
1023	<Which parameter>	16#20 (32) = Unreachable parameter. Check WriteData –
1024	<Which parameter>	16#30 (48) = Out of range Address out of range or value out of range E.g. For writing data I out 16#6064. I should have put 16#6064_0000
1025	<index line>	An invalid entry in the Point Table UDT. EXERR gives the index number
1026	AOI ID ¹	CommBusy Watchdog – CommBusy on too long, no communication with J4
1027	0	Gearing flags not responding. Servo may have lost reprogrammed Gearing registers. Initialize or recycle power to PLC.
1028	AOI ID ¹	Attempted motion while AOI is in Initialize state
1050	1	Gearing dropped out (2D16.14 went low)

¹ In the Mitsubishi_Servo UDT, the following are placed in the extended register EXERR to indicate which AOI instruction set this error

(e.g. 1017, 1021) or which mode/AOI is active (e.g. 1016)

- 1 = MMAJ - Jog
- 2 = MMAM - Move/Position
- 3 = MMAT - Torque
- 4 = MMAH - Home
- 5 = MMAPT – Point Table
- 6 = MMAG – Gearing
- 7 = MMAW - Watch
- 8 = MMCFG – Configure Servo
- 9 = MMAR – Registration
- 10 = MMAFR – Fault Reset

Cross Reference of CANOpen Object vs J4 Parameters

You can access the servo parameters two ways. There is the original CANOpen address and there is a mapping table so that J4 parameters have an equivalent CANOpen address.

J4	CANOpen	Note: J4 parameters are decimal CANOpen parameters are hexadecimal PA14 <> 2014h PA14 = 200Eh Note: Parameters starts at 1, not 0. E.g. there is no PA00.
PA_	2001h to 2020h	
PB_	2081h to 20C0h	
PC_	2101h to 2150h	
PD_	2181h to 21B0h	
PE_	2201h to 2240h	
PF_	2281h to 22C0h	
PL_	2401h to 2430h	
PT_	2481h to 24D0h	
PN_	2581h to 25A0h	

For example the CANOpen Home object is 6099h. Sub-index 1 is the homing speed and sub-index 2 is the creep speed.

Index	Sub	Name	Data Type	Access	PDO Mapping
6099h	0	Homing speeds	UNSIGNED8	ro	Impossible
	1	Speed during search for switch	UNSIGNED32	rw	Possible
	2	Speed during search for zero			

In the J4 parameters PT05 is the homing speed and PT06 is the creep speed.

Positioning control					Selected Items Write	As
No.	Abbr.	Name	Units	Setting range		
PT01	**CTY	For manufacturer setting		0200-0300		
PT02	*TOP1	For manufacturer setting		0000-0001		
PT03	*FTY	For manufacturer setting		0000-0300		
PT04	*ZTY	For manufacturer setting		0000-0300		
PT05	ZRF	Home position return speed	r/min	0.00-167772.15		
PT06	CRF	Creep speed	r/min	0.00-167772.15		

Using the table above, PT05 is 2485h and PT06 is 2486h. **With the mapping table, all sub-indexes are 0.**

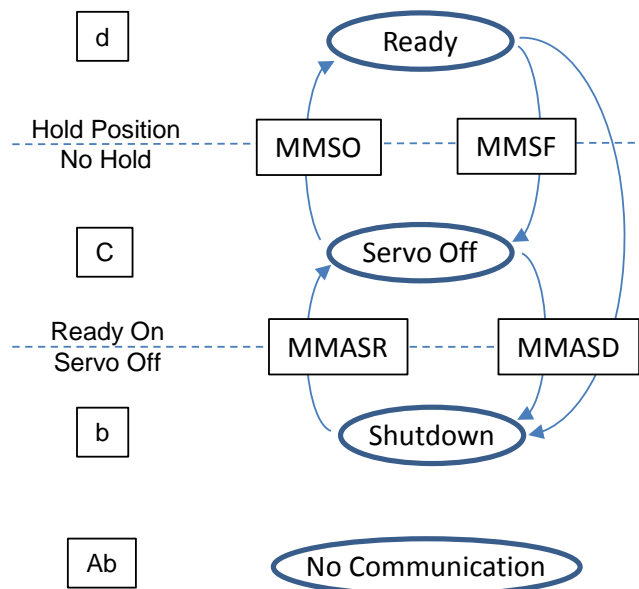
Therefore you can address the Home creep speed 2 ways

- using index = 6099h and sub-index = 2
- Using index = PT06 > 2486h and sub-index = 0.

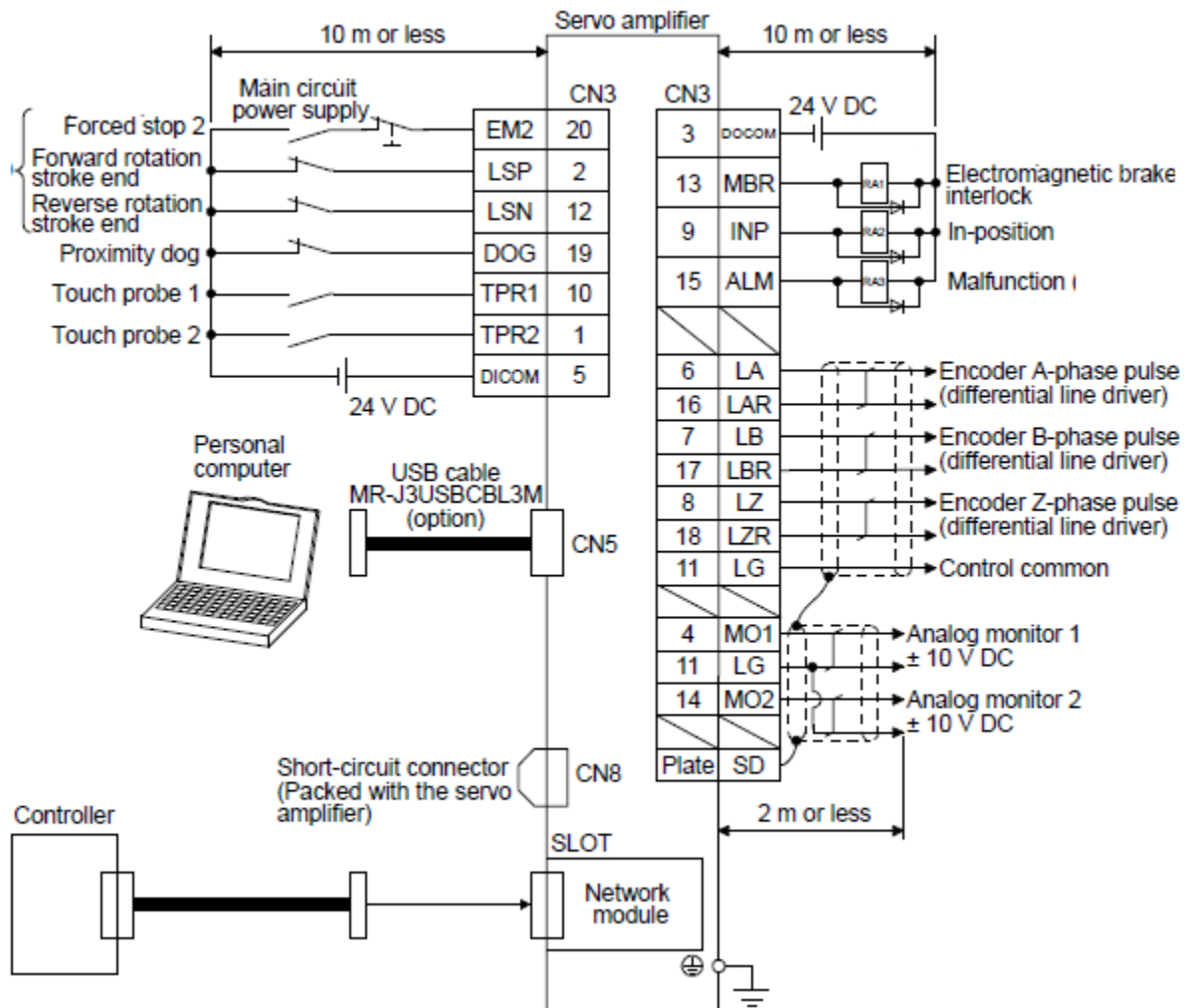
LED & Servo States

d	"d" : Indicates ready-on and servo-on status MMSO Servo On - enables the servo to "d"
C	"C": Indicates ready-on and servo-off status. MMSF Servo Off – <ul style="list-style-type: none"> disables the servo to "C" from "d" If at "C" or "b", no effect. MMASR Shutdown Reset – resets <servo>.ShutdownStatus flag <ul style="list-style-type: none"> If at "b" enables the servo to "C" If at "C" or "d", no effect.
b	"b" : Indicates ready-off and servo-off status. MMASD Shutdown - electronically disables the servo to "b". Must trigger MMASR to reset it internally. Uses <servo>.ShutdownStatus flag.
Ab	No communication between J4 and PLC. Ping both IP addresses for J4 Anybus and PLC Generic Ethernet module. If they ping, then the problem is in the PLC program.

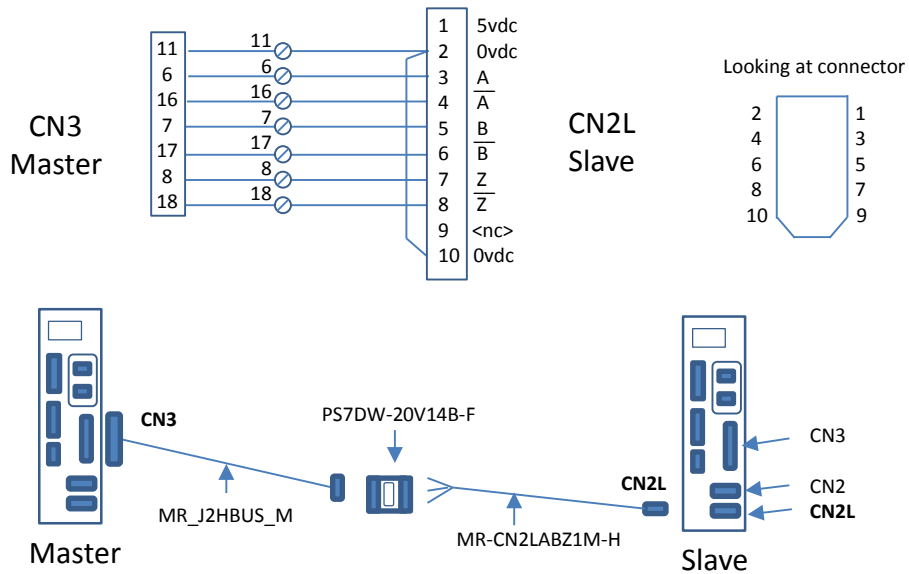
(Power On state based on state at Power Off)



Servo I/O interface Connections



Master/Slave Interface Connections for Gearing



The cable from DIN rail mounted terminal block to CN3

MR-J2HBUS__M	05, 1, 3, 5
--------------	-------------

e.g.

MR-J2HBUS05M for 0.5 meters

The DIN rail mounted terminal block is

PS7DW-20V14B-F

This part number is CN2L connector at one end and flying leads at the other.

MR-CN2LABZ1M-H

Write Loop 1

In the following example, 3 parameters are written to the servo:

16#2009 – Auto tuning parameter PA09

16#200A – Forward torque limit PA11

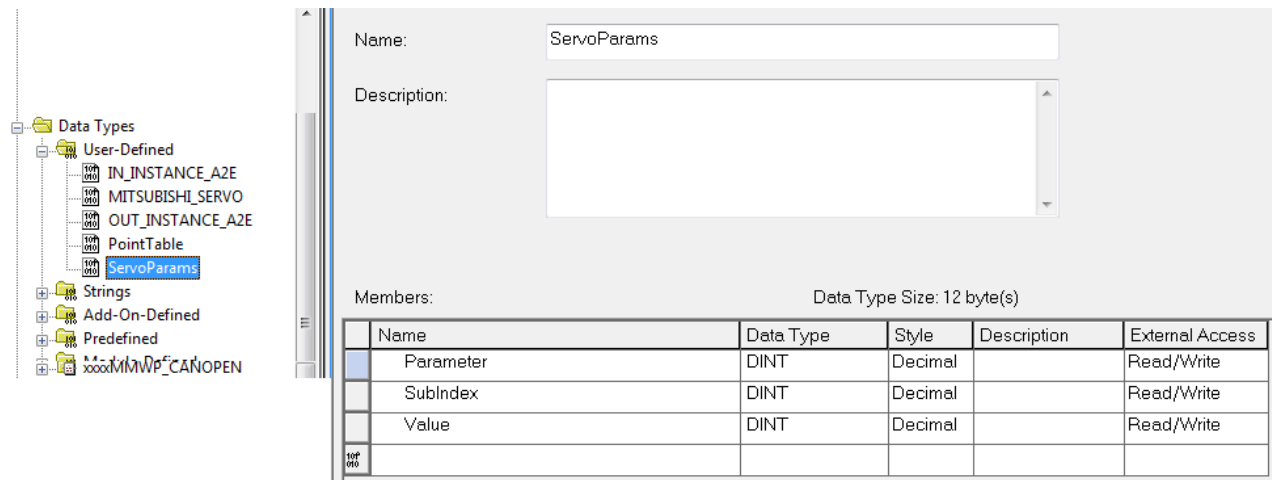
16#200B – Reverse torque limit PA12

You set **bWrite** high and it will reset itself after the loop is done. You can increase the number of parameters, you simply update the **GEQ** to turn off the loop.



Write Loop 2

For this example, the parameters are stored as an array. The ladder logic sequences thru the array to the end or if it encounters a 0 parameter.



The screenshot shows the 'Data Types' tree on the left with 'ServoParams' selected. The main window displays the definition for 'ServoParams'.

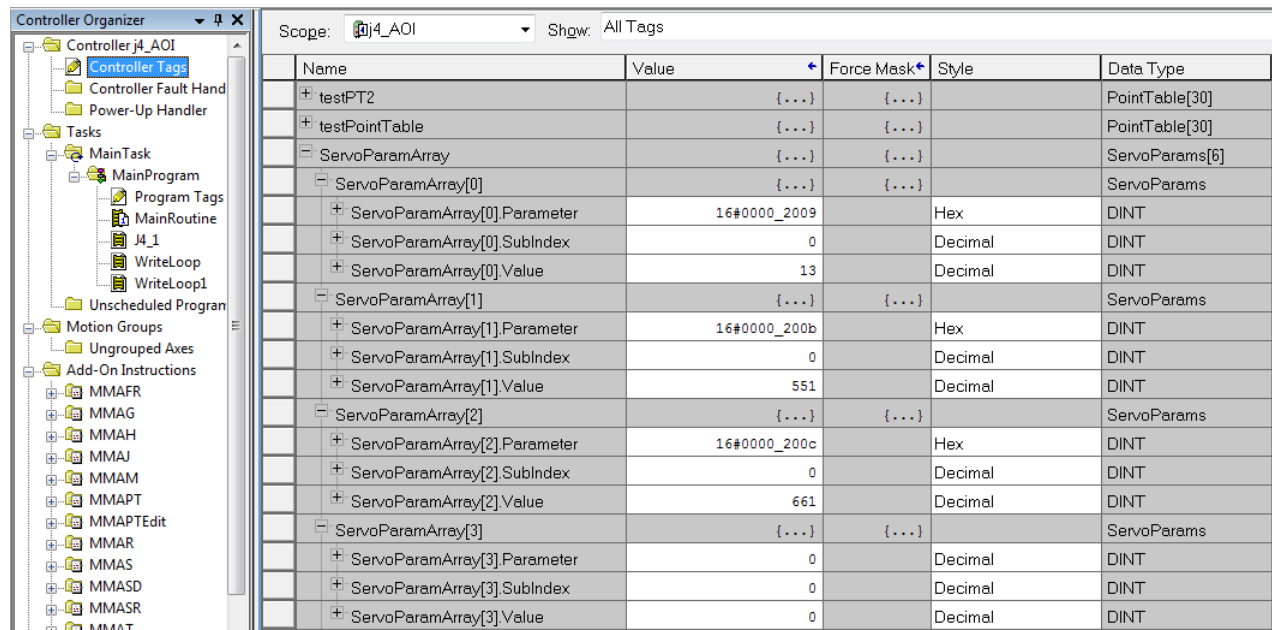
Name: ServoParams

Description:

Members:

Name	Data Type	Style	Description	External Access
Parameter	DINT	Decimal		Read/Write
SubIndex	DINT	Decimal		Read/Write
Value	DINT	Decimal		Read/Write

Data Type Size: 12 byte(s)



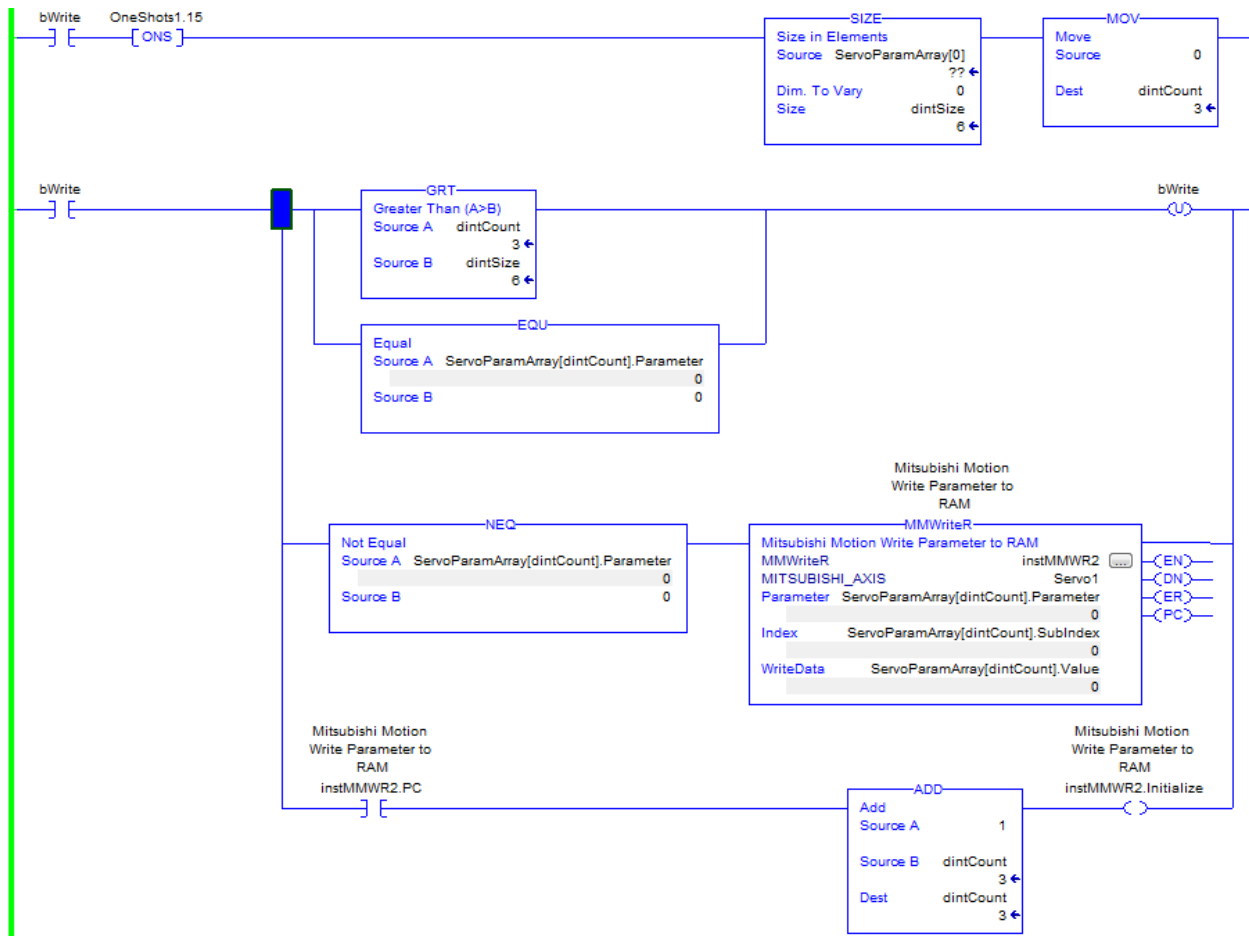
The screenshot shows the 'Controller Organizer' on the left and a list of tags in the main window. The scope is set to 'J4_AOI'.

Controller Organizer:

- Controller J4_AOI
 - Controller Tags
 - Controller Fault Hand
 - Power-Up Handler
 - Tasks
 - MainTask
 - MainProgram
 - Program Tags
 - MainRoutine
 - J4_1
 - WriteLoop
 - WriteLoop1
 - Unscheduled Program
 - Motion Groups
 - Ungrouped Axes
 - Add-On Instructions
 - MMAFR
 - MMAG
 - MMAH
 - MMAJ
 - MMAM
 - MMAPT
 - MMAPTEdit
 - MMAR
 - MMAS
 - MMASD
 - MMASR
 - MMAT

Tags List:

Name	Value	Force Mask	Style	Data Type
testPT2	{...}	{...}		PointTable[30]
testPointTable	{...}	{...}		PointTable[30]
ServoParamArray	{...}	{...}		ServoParams[6]
ServoParamArray[0]	{...}	{...}		ServoParams
ServoParamArray[0].Parameter	16#0000_2009		Hex	DINT
ServoParamArray[0].SubIndex	0		Decimal	DINT
ServoParamArray[0].Value	13		Decimal	DINT
ServoParamArray[1]	{...}	{...}		ServoParams
ServoParamArray[1].Parameter	16#0000_200b		Hex	DINT
ServoParamArray[1].SubIndex	0		Decimal	DINT
ServoParamArray[1].Value	551		Decimal	DINT
ServoParamArray[2]	{...}	{...}		ServoParams
ServoParamArray[2].Parameter	16#0000_200c		Hex	DINT
ServoParamArray[2].SubIndex	0		Decimal	DINT
ServoParamArray[2].Value	661		Decimal	DINT
ServoParamArray[3]	{...}	{...}		ServoParams
ServoParamArray[3].Parameter	0		Decimal	DINT
ServoParamArray[3].SubIndex	0		Decimal	DINT
ServoParamArray[3].Value	0		Decimal	DINT



Initialize

Most of the AOIs have an Initialize input that is found under the DOT operator. E.g. instMMAJ.Initialize.

- Clears the internal state flags and the outputs (IP, ER, PC, etc).
- Clears all error messages (except the following)
- With motion AOIs (MMAH, MMAM, MMAJ, MMAT) it will set an error if you have the motion rung energized or if you try to trigger a motion AOI when Initialized is asserted. Error 1028.
- ***Initialize DOES NOT STOP MOTION. It only initializes the AOI state.***
- Once set, it will lock out the AOI until Initialize is reset and the motion rung is NOT active (in the case of motion AOIs MMAH, MMAM, MMAJ, MMAT).

Explicit MSG



Scope: Test

Show: All Tags

	Name	Alias	Base T	Data Type	Description
	ReadFromAms			MESSAGE	

Message Configuration - ReadFromAms

Configuration | Communication | Tag

Message Type: CIP Generic

Service Type: Get Attribute Single

Service Code: e (Hex) Class: 64 (Hex) Instance: 11023 Attribute: 0 (Hex)

Source Element: [] Source Length: [] Destination Element: F []

☒ Enable ☐ Enable Waiting ☐ Start ☒ Done Done Len

☐ Error Code: Extended Error Code: ☐ Timeo

Error Path:

Error Text:

OK Cancel Apply Help

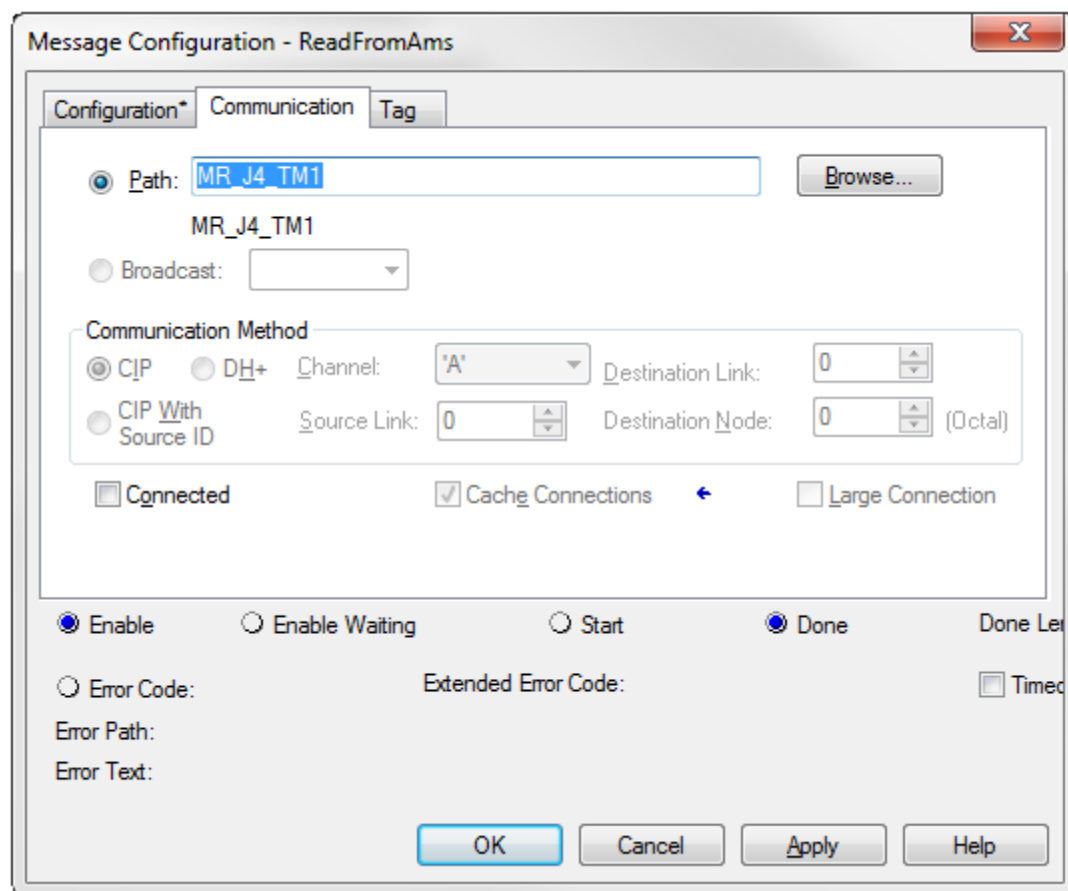
Note the Instance is NOT hex. 11023 = 2B0F Hex

- Message Type = CIP Generic
- Service Type = Get Single Attribute (automatically sets Service Code)
- Service Code (HEX) = "e" - Get Single Attribute
- Class (HEX) = 64 - this is the DS402 Drive Object

(13) Bus voltage (2B0Fh)

Ins ID	Attr ID	Access	Name	Data Type
2B0Fh	0	Get	Bus voltage	UINT

- Instance ID (Decimal) = the object **IN DECIMAL** - 11023 = 2B0F Hex. This is Bus Voltage.
- Attribute ID - the index within the Instance ID - see documentation. Typically 0 for single INT. Some items have indexes 0, 1, 2, ...



Message Configuration - ReadFromAms

Configuration* Communication Tag

Path: Browse...

MR_J4_TM1

Broadcast:

Communication Method

☒ CIP ☐ DH+ Channel: Destination Link:

☐ CIP With Source ID Source Link: Destination Node: (Octal)

☐ Connected ☒ Cache Connections ☐ Large Connection

☒ Enable ☐ Enable Waiting ☐ Start ☒ Done Done Len

☐ Error Code: Extended Error Code: ☐ Timec

Error Path:

Error Text:

OK Cancel Apply Help

14. Major Revisions

Date	Version	Description
June 23, 2017	2.000	Document created and Released

15. References

Company	Document No.	Document Name	Version	Date
Mitsubishi	SH(NA)030226-A	MR-J4-_TM_ SERVO AMPLIFIER INSTRUCTION MANUAL (EtherNet/IP)	A	10/22/2014
	R72-164-SLSASG-001	J4TM AOI User Manual	1	01/10/2015
	SH(NA)030109	MELSERVO-J4 Servo amplifier INSTRUCTION MANUAL (TROUBLE SHOOTING)	*	12/18/2015
HMS		Driver manual Anybus CompactCom 40 motion driver	1.11	10/9/2014
		EtherNet/IP Extension Anybus CompactCom 40 motion driver	1.03	10/7/2014

In the **KnowledgeBase**, either enter the Document No. or the text in Document Name to find the manual.