



Dynamic Motor Motion
Technology Corporation

DYN5 Series

AC SERVO DRIVE

Modbus TCP/IP Specification

MANUAL CODE: DYN5-MBTCP-SL425-10A

REVISION: 10A

RELEASE DATE: October 2021

This manual must be kept accessible for the user or operator.
Copyright © 2021 DMM Technology Corp.

Document Layout Dimension:
Letter 8.5 x 11 inches (215.9 x 279.4 mm)

■ Safety Notice ■



The user or operator should read through this manual completely before installation, testing, operation, or inspection of the equipment. The DYN5 series AC Servo Drive should be operated under correct circumstances and conditions. Bodily harm or damage to equipment and system may result if specifications outlined in this document are not followed. Take extra precaution when the above warning convention is used for certain critical specifications.

■ Notations Used ■

Unless otherwise noted, all specification and units of measurement used in this manual are in Metric standard units:

Mass: Kilogram [kg]

Length: Millimeter [mm]

Time: Seconds [s]

Temperature: Celsius [°C]

Table of Contents

■ Safety Notice ■	2
■ Notations Used ■	2
Table of Contents	3
Section 1. Overview	4
Section 2. Network Connection	5
Section 3. Basic Setup Instructions	6
3.1 Servo Drive Setup	7
3.2 Ping (ICMP) Test	8
3.3 Modbus Poll Test	9
Section 4. Modbus Register Specifications	11
4.1 Modbus Register Overview	11
4.2 Modbus Register Outline	12
4.3 Modbus Register Details	13
Section 5. PLC Communication Example	22
5.1 Initial Communication - Read from register 0xFFFE Diagnostic Counter	23
5.2 Speed Command and Speed Feedback Example	25
5.3 Profile Relative Position Command Example	28
Section 6. Servo Drive Communication Response Time	31
Appendix A - DMMDRV5 Communication Setup	32
Appendix B - Profile Position Command Trajectory Calculator	33
Appendix C - DTPU Dynamic Target Position Update Specification	34
Warranty and Liability	36
Manual Disclaimer	37
Manual Revisions	37

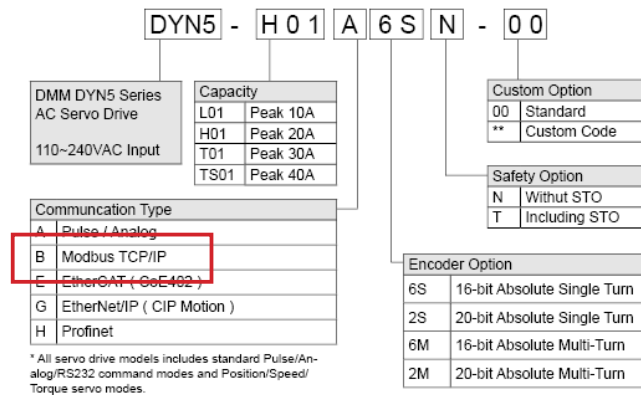
Section 1. Overview

The DYN5 servo drive optioned with Modbus TCP/IP communication allows the servo drive to be fully operated on a Modbus TCP/IP network. The DYN5 servo drive acts as Server (Slave) device and exchanges data with a Client (Master) device. Through this network, the Client has full access and control of the servo drive including:

- Reading and Writing servo drive parameters
- Reading servo motor Position/Speed/Current/Torque
- Reading servo drive Status
- Sending Position/Speed/Torque command

◆ Compatible Model Numbers

The DYN5 servo drive with the below model numbers are optioned with Modbus TCP/IP communication. Note that the servo drive optioned with Modbus TCP/IP still has base Pulse/Analog/RS232 capability if the user wishes to use these control methods.



For additional information regarding Modbus TCP/IP network and protocol specification, please refer to Modbus Organization at www.modbus.org.

◆ Basic Specification

Interface	10/100 Base-T Ethernet IEEE 802.3
Hardware Interface	RJ45 Recommended CAT5 or higher cable with braid shielding
Communication Speed	10/100 Mbps - Drive Auto Detect
IP Addressing	Static – Set in DMMDRV5 program

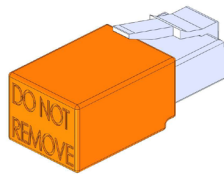
Section 2. Network Connection

For general DYN5 servo drive operation and wiring instructions, refer to DYN5 servo drive instruction manual Manual# DYN5MS-ZM1.

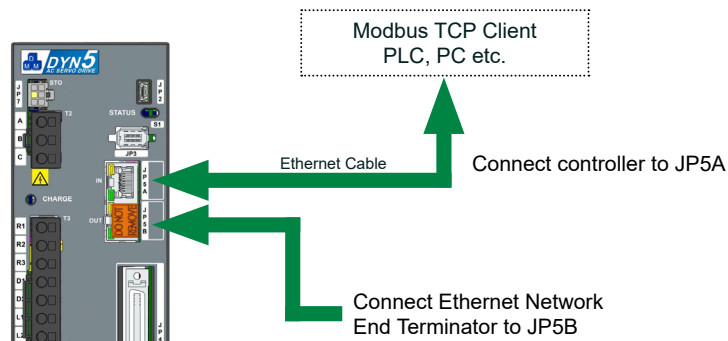
Connector JP5A and JP5B are used for the Modbus TCP/IP interface. Both connectors are standard RJ45.

The DYN5 servo drive supports daisy-chain topology, JP5A is used for input, JP5B is used for output. The last servo drive on the chain should have JP5B terminated with the JP5 Terminator (Part# CN5-JP5-TMD1). If only 1 servo drive is on the network, the terminator still needs to be connected to JP5B, with JP5A connected to the Client (Master).

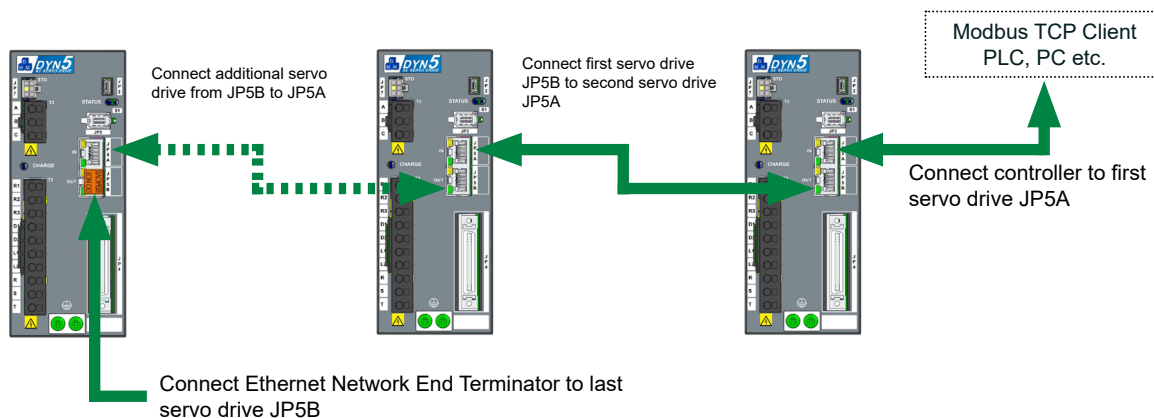
- ◆ Part# CN5-JP5-TMD1 - DYN5 JP5 Ethernet Network End Terminator



- ◆ Single Servo Drive Network Connection



- ◆ Multiple Servo Drive Network Connection



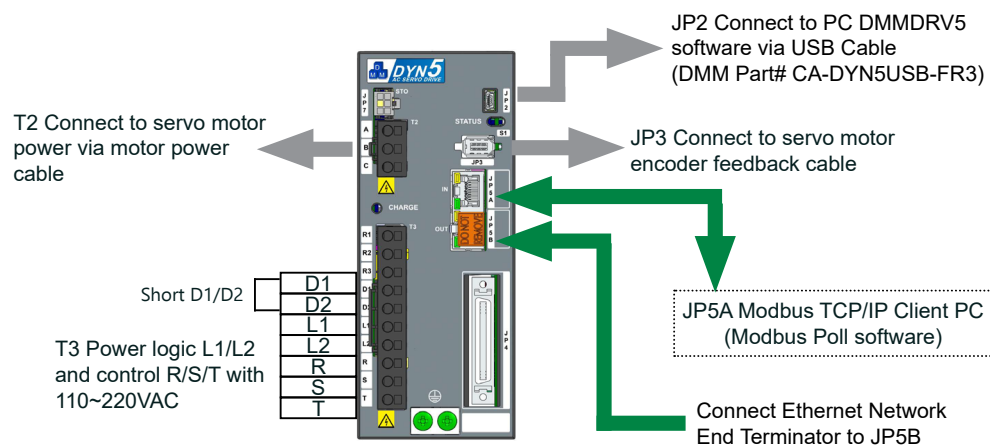
Section 3. Basic Setup Instructions

Follow below instructions to setup a Modbus TCP/IP communication between the DYN5 servo drive and a PC Modbus TCP/IP Client. These instructions are the same for all DYN5 servo drive models with Modbus TCP/IP option.

For general DYN5 servo drive operation and wiring instructions, refer to DYN5 servo drive instruction manual Manual# DYN5MS-ZM1.

(A) Wiring and Connections

Follow below diagram for basic minimum wiring for Modbus TCP/IP communication testing. Note the below diagram does not include any reference for EMI, grounding or safety. Refer to DYN5 servo drive instruction manual for these considerations.



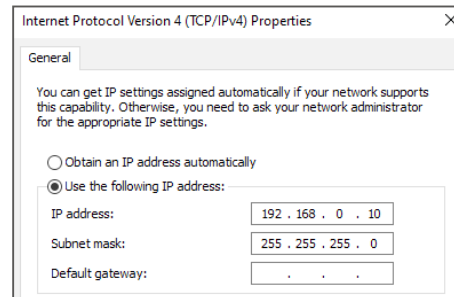
(B) Parts Needed

- DYN5 AC Servo Drive with Modbus TCP/IP option
- DXT/DST/DHT AC Servo Motor
- Encoder Feedback Cable to connect servo motor to servo drive
- Motor Power Cable to connect servo motor to servo drive
- Mini-USB to USB-A cable to connect servo drive to PC DMMDRV5 software (DMM Part# CA-DYN5USB-FR3)
- Ethernet cable to connect servo drive JP5A to PC for Modbus TCP/IP communication
- Part# CN5-JP5-TMD1 - DYN5 JP5 Ethernet Network End Terminator
- PC computer with DMMDRV5 software
- PC computer with Modbus Poll software (www.modbustools.com)

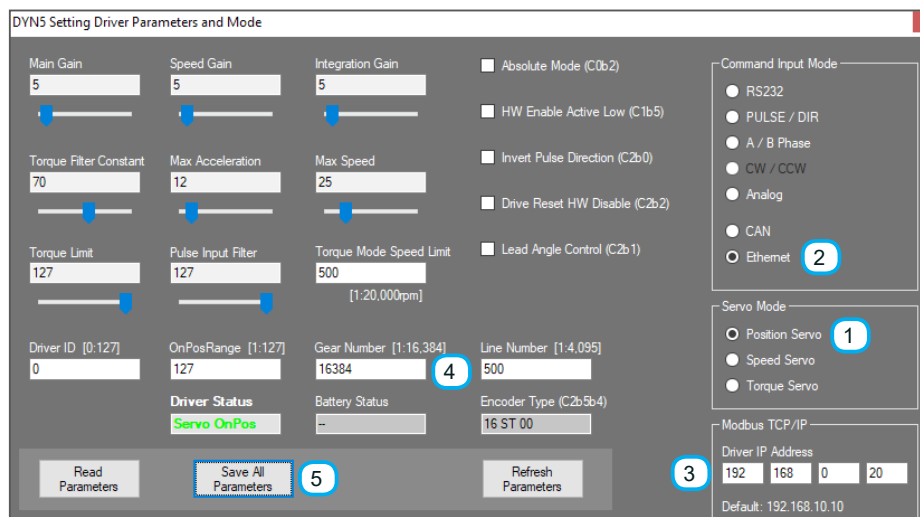
*** Note all ethernet cables used should be Straight Through type and not Crossover type cables**

3.1 Servo Drive Setup

- (1) Download and install DMMDRV5 software.
- (2) Download and install Modbus Poll software (www.modbustools.com). Modbus Poll is a Modbus TCP/IP master (Client) simulator software for PC computer.
- (3) Check PC ethernet port TCP/IPv4 settings. This is needed to set and match with the servo drive IP address. For this example, the PC IP address is set to 192.168.0.10 and subnet mask 255.255.255.0



- (4) Connect all components to servo drive as above diagram (A) Wiring and Connections. Apply power to servo drive.
- (5) Establish connection between DMMDRV5 program and DYN5 servo drive. See Appendix A for connection setup instructions.
- (6) Open the Servo Setting module. Select Position Servo Mode **1**. Select Ethernet Command Input Mode (Modbus TCP/IP) **2**. Note no matter the operation command from Modbus TCP/IP, the servo drive should be set into Position Servo Mode here. The servo drive will internally switch between Position/Speed/Torque servo modes according to the command received from Modbus TCP/IP. Set the servo drive IP address within the network of the PC IP address **3**. In this example, the servo drive IP address is set to 192.168.0.20. Set GEAR_NUM to 16384 **4**.



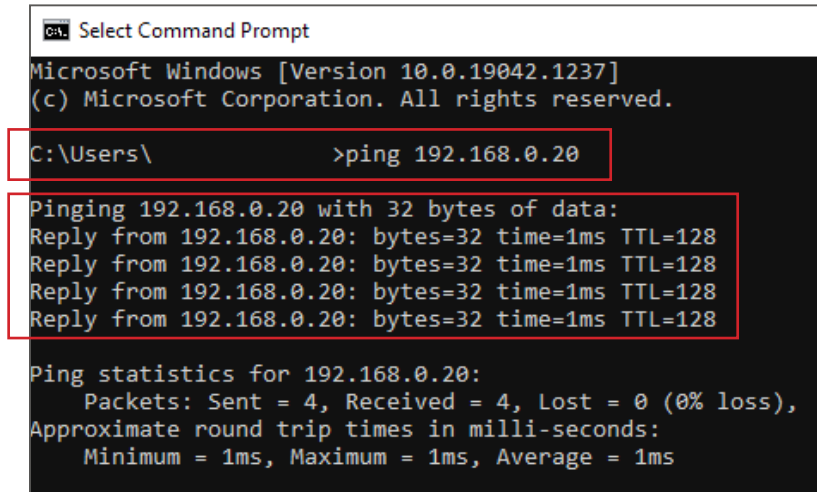
- (7) Click Save All to save new settings into servo drive **5**. Power cycle servo drive - remove power from both L1/L2 and R/S/T. Wait 30 seconds, then apply power again.

The DYN5 servo drive is now correctly setup for Modbus TCP/IP.

3.2 Ping (ICMP) Test

Check servo drive ethernet connectivity using Ping test on PC.

- (1) Open the PC Command Prompt.
- (2) Type "ping AAA.AAA.AAA.AAA" where AAA.AAA.AAA.AAA is the servo drive IP address set in Section 3.1 Step. 6.
- (3) Check that a reply is received back from the servo drive. This confirms the servo drive is in Ethernet mode and IP address is set correctly.



```
Microsoft Windows [Version 10.0.19042.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ >ping 192.168.0.20

Pinging 192.168.0.20 with 32 bytes of data:
Reply from 192.168.0.20: bytes=32 time=1ms TTL=128
Reply from 192.168.0.20: bytes=32 time=1ms TTL=128
Reply from 192.168.0.20: bytes=32 time=1ms TTL=128
Reply from 192.168.0.20: bytes=32 time=1ms TTL=128

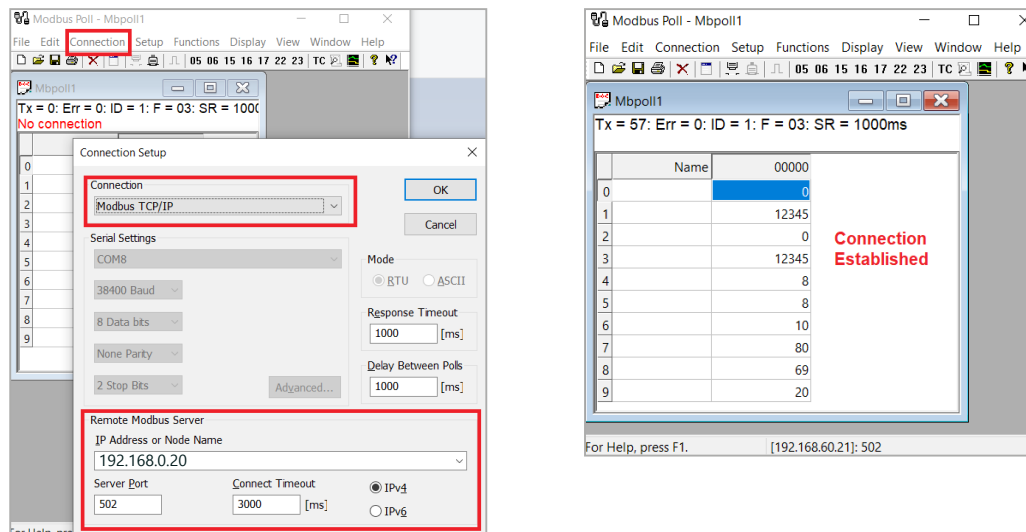
Ping statistics for 192.168.0.20:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 1ms, Average = 1ms
```

- (4) If reply is not received from servo drive, check all connections and settings in Section 3.1.

3.3 Modbus Poll Test

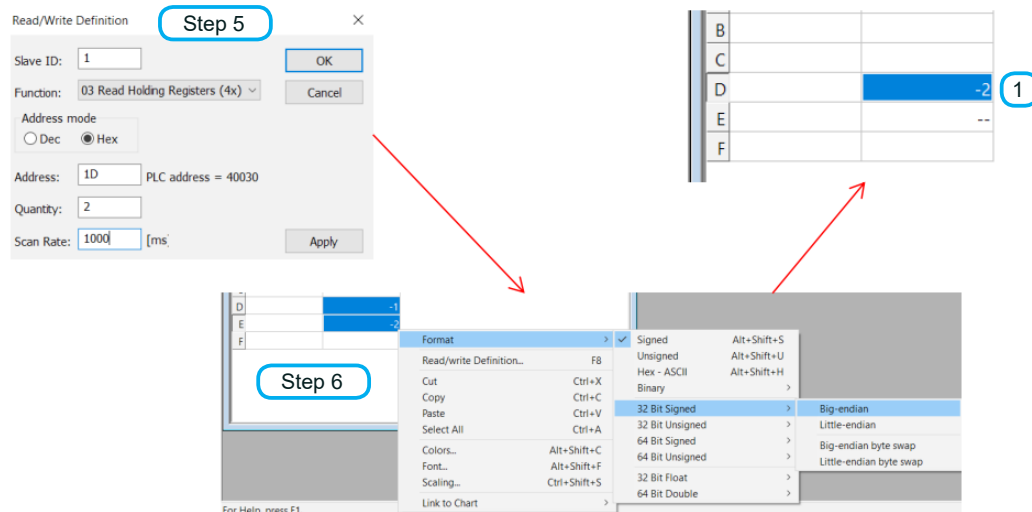
Check Modbus TCP/IP connectivity using Modbus Poll software. Modbus Poll software turns the PC into Modbus Client (master) used to test servo drive communication.

- (1) Open the Modbus Poll software.
- (2) Click Connection->Connect. Select Modbus TCP/IP connection. Input servo drive IP address. Make sure server port is set to 502 for standard Modbus TCP port. Make sure IP protocol is set to IPV4. Click OK. Check that connection is established and registers are read from servo drive.



- (4) See Section 4. for servo drive Modbus Register specification. The default Modbus Poll program reads 10 registers from address 0x00 to 0x09. The registers are read once a second and the “Tx” counter on top increments each time a read is successful.

(5) (Relative Profile Position command example) Set Modbus Poll so it reads from register containing servo motor position. Click Setup->Read/Write Definition. 32-bit servo motor position is contained in registers 0x1D and 0x1E. Set to below setting and click OK. Note Slave ID number does not matter.

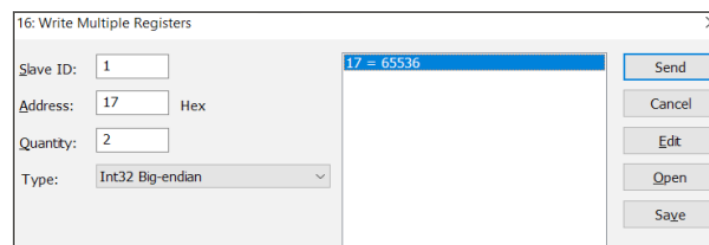


(6) Select both 0x1D and 0x1E register displays and right click, select Format->32-bit signed->big endian. This will automatically combine the two 16-bit Modbus registers into one 32-bit signed data showing full motor position on 0x1D display location. The position displayed should be 0 or close to 0 as the motor position drifts slightly at steady state (1). If the servo drive is set to Absolute mode, the readout position will be single-turn or multi-turn absolute position.

(7) Recall in Section 3.1, Step 6, GEAR_NUM parameter was set to 16384. This sets the position command ratio so that a profile position command of 65536 will move the motor 1 revolution (for motors equipped with 16-bit resolution encoders). See Appendix B for Profile Position Command trajectory details.

(8) Send a command to move the motor 1 revolution. Select Functions->16 Write Registers. Relative Profile position command is register 0x17 and 0x18.

(9) Set to settings as below and Click Send to send a position command of 65536 and check that the motor moves exactly 1 revolution Clockwise. The position readout from 0x0D (Step 6) should now read 65536 to indicate the motor has rotated exactly 1 revolution.



Section 4. Modbus Register Specifications

4.1 Modbus Register Overview

All DYN5 modbus registers are standard 16 bits wide Modbus Holding Registers.

- ◆ The DYN5 servo drive supports the following Modbus Function codes

Function Code	Function Name
03	Read Holding Registers (Single + Multiple)
06	Write Single Holding Register
16	Write Multiple Holding Registers

- ◆ The DYN5 servo drive supports the following Modbus Exception codes

Exception Code	Exception Name	Exception Code Cause
01	Illegal Function	Sending Function code not 03, 06 or 16
02	Illegal Data Address	Attempt to access register that does not exist
03	Illegal Data Value	Write data into a register outside allowed range

- ◆ General Notes regarding DYN5 servo drive Modbus Register behaviour

- Reading from Write-Only or non-existent registers will return 0d12345
- Writing to Read-Only registers or non-existent registers will have no effect, but will not throw exception reply
- Reading from register number 0x27 or larger, or not 0xFFFFE will throw 02 exception
- Writing to register number 0x27 or larger, or not 0xFFFFE will throw 02 exception
- If command 03 Read Multiple Register exceeds last register 0x27, it will be accepted but data will return 0d12345 for non-existent registers higher than 0x27
- If command 16 write multiple register exceeds last register 0x27, it will be accepted but write data will have no effect on non-existent registers higher than 0x27
- Command 16 Max write length = 40 registers at once. If write more than 40, drive throws 02 exception
- Command 03 Max Read length = 40 registers at once. If read more than 40, drive throws 02 exception
- Profile position command value, should be within [-134,217,728 : 134,217,727], else will not run command but will not throw exception

4.2 Modbus Register Outline

Register#		Register Map	Access	Data Type	Data Range	Description
Modbus Decimal	Hex					
40001	0	NA (Memory Test Register)	RW	Int16	0~63	Whatever value written into this register can be read back. No function related to servo drive. Default = 0.
40003	2	Drive Status	R	Int16	0~127	8-bit Servo Drive Status Byte
40004	3	Set ABS Origin	W	Int16	0xFFFF	Set Absolute Origin Zero Command (multi-turn system). Setting Register to 0xFFFF sets current position to multi-turn absolute zero.
40005	4	Main Gain	RW	Int16	1~127	Main Gain parameter. [Pr. 00]
40006	5	Speed Gain	RW	Int16	1~127	Speed Gain parameter. [Pr. 01]
40007	6	Integration Gain	RW	Int16	1~127	Integration Gain parameter. [Pr. 02]
40008	7	Torque Constant	RW	Int16	1~127	Torque Filter Constant parameter. [Pr. 03]
40009	8	High Speed	RW	Int16	1~127	High Speed parameter. [Pr. 07]
40010	9	High Accel	RW	Int16	1~127	High Acceleration parameter. [Pr. 11]
40011	a	On Position Range	RW	Int16	1~127	On Position Range parameter. [Pr. 15]
40012	b	GEAR_NUM	RW	Int16	1~16384	Gear Number parameter. Electronic Gear Ratio. Changes take effect after power cycle or Reset. [Pr. 19]
40013	c	LINE_NUM	RW	Int16	1~4095	Line Number parameter. Encoder Output Gear Ratio. Changes take effect after power cycle or Reset. [Pr. 20]
40014	d	Drive SW Enable/Disable	RW	Int16	0xAAAA 0xDEDE	Servo Drive Software Enable/Disable command. Setting register to 0xAAAA is software Enable command. Setting register to 0xDEDE is software Disable command.
40015	e	Turn_ConstSpeed	W	Int16	-2 ¹⁴ ~2 ¹⁴	Turn Constant Speed command [rpm]. Acceleration/deceleration when changing speed is controlled by Max Acceleration parameter.
40016	f	Square_Wave Motion Amplitude	W	Int16	0~4096	Square wave amplitude command. 4096=180degrees. First send SS_Frequency, then Square_Wave Motion Amplitude.
40018	11	Sin_Wave Motion Amplitude	W	Int16	0~4096	Sine wave amplitude command. 4096=180degrees. First send SS_Frequency, then Sin_Wave Motion Amplitude.
40020	13	SS_Frequency	W	Int16	0~60	Square / Sine wave frequency
40021	14	Motor Speed	R	Int16		Motor Speed [rpm] output
40022	15	Go_Absolute_Pos_Profile Command	W	Int32	-2 ²⁷ ~ 2 ²⁷	Go absolute profile position – High 16 Bytes
40023	16					Go absolute profile position – Low 16 Bytes (Start Move Trigger)
40024	17	Go_Relative_Pos_Profile Command	W	Int32	-2 ²⁷ ~ 2 ²⁷	Go relative profile position – High 16 Bytes
40025	18					Go relative profile position – Low 16 Bytes (Start Move Trigger)
40030	1d	Motor Absolute Position	R	Int32	-2 ²⁷ ~ 2 ²⁷	Motor Absolute Position output – High 16 Bytes
40031	1e					Motor Absolute Position output – Low 16 Bytes
40032	1f	Motor Torque	R	Int16	-981~981	Motor Torque readout. 981 corresponds to max servo drive current output.
40033	20	Torque Limit	RW	Int16	0~127	Global Current/Torque Limit. 127 corresponds to 100% current allowed. 64=current limited to 50%. [Pr. 04]
40034	21	Drive Reset	W	Int16	0xABCD	Servo Drive Reset Command. Setting register to 0xABCD sends Reset command to servo drive to clear faults.
	0xFFFE	Diagnostic Counter	R	Int16	0~255	Diagnostic Counter, returns +1 each read.

4.3 Modbus Register Details

Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
40001	0	NA (Memory Test Register)	RW	Int16	0~63
Details					
<p>This register serves as a memory test for the controller. Whatever value written into this register can be read back. Data is volatile and will be lost after power cycle or Reset. Default value after power up is 0. This data serves no function for the servo drive otherwise.</p>					

Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
40003	2	Drive Status	R	Int16	0~127
Details					
<p>This register contains 8-bit Servo Drive_Status Byte. Can be read to monitor servo drive operation status and motion status.</p> <p>Drive_Status byte = b7 b6 b5 b4 b3 b2 b1 b0</p>					
Bit	Value	Description			
b0	0	Motor On Position. $ Pset - Pmotor \leq OnpositionRange$ Parameter Pr.15 Corresponds to ON/LOW state at JP4.Pin14/15 ONPOS output.			
	1	Motor Off Position. $ Pset - Pmotor > OnPositionRange$ Parameter Pr.15 Corresponds to OFF/HIGH state at JP4.Pin14/15 ONPOS output.			
b1	0	Servo Enabled. Both Hardware and Software Enable are Enabled.			
	1	Servo Disabled / Motor Free			
b4 b3 b2	0	No Alarm			
	1	Motor Lost Phase alarm, $ Pset - Pmotor > 8192(\text{steps}), 180(\text{deg})$			
	2	Over Current Alarm			
	3	Overheat Alarm / Overpower Alarm			
	4	Error for CRC code check, refuse to accept current command			
	5	Over Voltage Alarm			
b5	0	Profile Position S-curve motion command completed. Note this only means the motion command is completed, does not mean motor is In-Position. See Appendix B for details.			
	1	Profile Position S-curve motion command running. See Appendix B for details.			
b6	0	Reserved - Do not use			
b7	0	Reserved - Do not use			

Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
40004	3	Set ABS Origin	W	Int16	0xFFFF

Details

When using multi-turn system, setting this register to 0xFFFF will set current servo motor position to Absolute Zero, then also performs a Reset command to reset servo control at current position.

Register has no functionality when using single-turn system. Setting register to any other value than 0xFFFF will return 03 exception response.

Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
40005	4	Main Gain	RW	Int16	1~127
40006	5	Speed Gain			
40007	6	Integration Gain			
40008	7	Torque Constant			
40009	8	High Speed			
40010	9	High Accel			
40011	a	On Position Range			

Details

These registers correspond to standard DYN5 servo drive parameters that can be read/set from Modbus TCP/IP. All parameters have normal DYN5 servo drive functionality.

These below parameters are saved into servo drive main EEPROM so have a 1 million time read/write lifecycle:

- [Pr.00] = Main Gain Parameter
- [Pr.01] = Speed Gain Parameter
- [Pr.02] = Integration Gain Parameter
- [Pr.03] = Torque Constant Parameter
- [Pr.15] = On Position Range Parameter

These below parameters below are saved into non-volatile EEPROM first 6 times after servo drive power up. After 6 times, any additional parameter saves are saved into volatile RAM. This is to accommodate Profile Position commands that constantly need to change Speed and Acceleration for each motion:

- [Pr.07] = High Speed Parameter
- [Pr.11] = High Accel Parameter

All the above parameters can be changed on-the-fly during operation and becomes effective as soon as new setting is saved into servo drive. Allow $\geq 300\mu s$ after sending parameter for new setting to become effective.

Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
40012	b	GEAR_NUM Parameter	RW	Int16	1~16384
40013	c	LINE_NUM Parameter	RW	Int16	1~4095

Details

These registers correspond to standard DYN5 servo drive parameters that can be read/set from Modbus TCP/IP. All parameters have normal DYN5 servo drive functionality.

GEAR_NUM parameter [Pr.19] is used for electronic scaling when sending Position command to servo drive. See Appendix B for usage details.

LINE_NUM parameter [Pr.20] controls the emulated incremental encoder output resolution from JP4.Pin44/45/46/47/48/49.

LINE_NUM parameter setting	Output Pulse Resolution Calculation
1~2048	Output Pulse = LINE_NUM x 4 Ex. If LINE_NUM is set to 500, servo drive outputs 2,000 pulse per motor revolution.
2048~4095	Output Pulse = (LINE_NUM - 2047) x 4 Ex. If LINE_NUM is set to 4095 (maximum), servo drive outputs 8,192 pulse per motor revolution.

These parameters are saved into servo drive EEPROM so have a 1 million time read/write lifecycle.

Change into these parameters are effective after power cycle or Rest command.

Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
40014	d	Drive SW Enable/Disable	RW	Int16	0xAAAA 0xDEDE

Details

This register is used to send servo drive Software Enable/Disable command. Setting register to 0xAAAA is software Enable command. Setting register to 0xDEDE is software Disable command.

After powered up or Reset command, Software Enable is automatically Enabled.

Setting register to any other value than 0xAAAA or 0xDEDE will return 03 exception response.

Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
40015	e	Turn_Const-Speed	W	Int16	2 ¹⁴ ~ 2 ¹⁴

Details

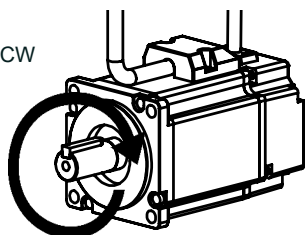
This register is used to send servo drive speed command. When data is sent to this register, servo drive internally switches into Speed Servo Mode to run speed command. Register data is rpm units. Ex. Setting register to 500 runs motor at 500rpm in positive direction.

Setting register to 0 stops motor movement.

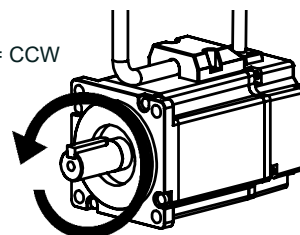
Acceleration/Deceleration when changing speed is controlled by *Max_Accel* parameter [Pr.11] and *GEAR_NUM* parameter [Pr.19]:

Motor Acceleration/Deceleration [rpm/s] = $Max_Accel \times 635.78 \times (4096 \div GEAR_NUM)$

Positive Command = CW



Negative Command = CCW



Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
40021	14	Motor Speed	R	Int16	

Details

This register contains servo motor speed in [rpm] units. Read this register to obtain current servo motor speed. Register data update rate is 300us.

The servo motor speed can be read whenever the encoder is correctly connected to the servo drive. Even when the servo drive is Disabled or Faulted/Alarmed and motor is free/coasting, the motor speed can be read.

Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
40016	f	Square_Wave Motion Amplitude	W	Int16	0~4096
40018	11	Sin_Wave Motion Amplitude	W	Int16	0~4096
40020	13	SS_Frequency	W	Int16	0~60

Details

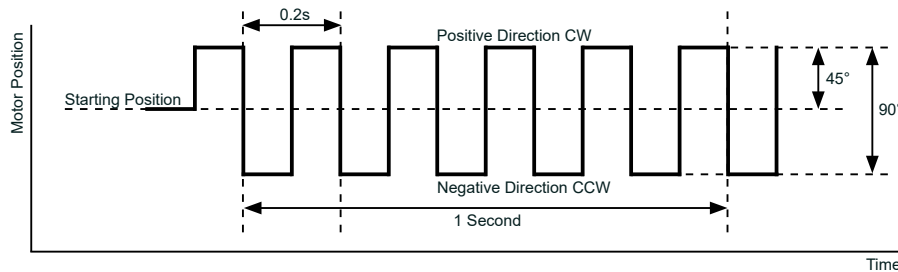
These registers are used to generate and run internal Square Wave and Sine Wave motion commands. First, send SS_Frequency command, then send Square_Wave or Sin_Wave amplitude command.

When Amplitude command is received by servo drive, servo drive internally switches to Square or Sine motion mode and runs motion immediately.

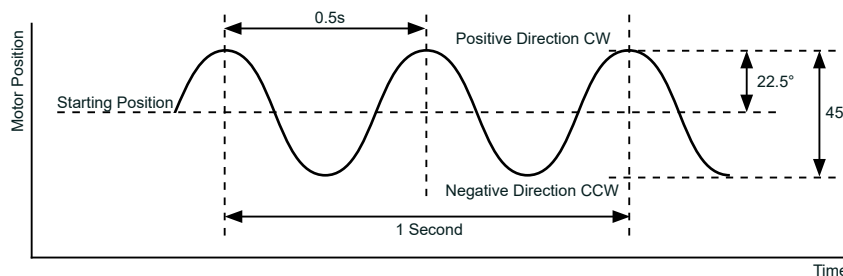
Square wave motion is a max acceleration/deceleration motion profile. Sine wave motion is a smooth, constant acceleration/deceleration motion profile.

Amplitude 0~4096 range corresponds to 0~180degrees motor movement.
Frequency 0~60 range corresponds to 0~60Hz motor movement.

Example 1: Setting Register 0x13 to 5, then setting Register 0x0F to 2048 runs a 5Hz Square wave motion at 90degree amplitude.



Example 2: Setting Register 0x13 to 2, then setting Register 0x11 to 1024 runs a 2Hz Sine wave motion at 45degree amplitude.



* See previous page for Register 40021 / 0x14 (Motor Speed) details.

Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
40022	15	Go_Absolute_Pos_ Profile Command	W	Int32	-2 ²⁷ ~ 2 ²⁷
40023	16		W		

Details

These two registers are used to send Absolute Profile Position Command to servo drive. When command is sent, servo drive internally switches to Position servo mode to run command immediately.

Position command is 32-bits size. Register 0x15 contains high 16-bits data. Register 0x16 contains low 16-bits data. When servo drive receives data into register 0x16, it will combine data with register 0x15 and start running command. In order to maintain data consistency, always set both high register and low register together. Do not set low register without setting high register.

Positive command turns motor in CW direction, negative command turns motor in CCW direction. See Appendix B for motion profile calculation. Allowed position command range = $\pm 134,217,728$.

Example:	<ul style="list-style-type: none"> - Move motor to 4,726,140 absolute position - 4,726,140 = 0x00481D7C. - Send 0x0048 to register address 0x15. Drive stores 0x0048 as Go Absolute Position high bytes. Does not run command. - Send 0x1D7C to register address 0x16. Drive stores 0x1D7C as Go Absolute Position low bytes. Combines value with high bytes, checks to see if command is within allowed range and runs command.
----------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
40024	17	Go_Relative_Pos_ Profile Command	W	Int32	-2 ²⁷ ~ 2 ²⁷
40025	18		W		

Details

These two registers are used to send Relative Profile Position Command to servo drive. When command is sent, servo drive internally switches to Position servo mode to run command immediately.

Position command is 32-bits size. Register 0x17 contains high 16-bits data. Register 0x18 contains low 16-bits data. When servo drive receives data into register 0x18, it will combine data with register 0x17 and start running command. In order to maintain data consistency, always set both high register and low register together. Do not set low register without setting high register.

Positive command turns motor in CW direction, negative command turns motor in CCW direction. See Appendix B for motion profile calculation. Allowed position command range = $\pm 134,217,728$.

Example:	<ul style="list-style-type: none"> - Move motor -15,898 relative position - -15,898 = 0xFFFFC1E6. - Send 0xFFFF to register address 0x17. Drive stores 0xFFFF as Go Relative Position high bytes. Does not run command. - Send 0xC1E6 to register address 0x18. Drive stores 0xC1E6 as Go Relative Position low bytes. Combines value with high bytes, checks to see if command is within allowed range and runs command.
----------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
40030	1d	Motor Absolute Position	R	Int32	-2 ²⁷ ~ 2 ²⁷
40031	1e				

Details

These registers contain the absolute position of the motor. Full motor position is 32-bits long. Register 0x1d contains higher 16-bits, register 0x1e contains lower 16-bits.

If using 16-bit encoder, one motor revolution will equal 65,536 absolute position. If using 20-bit encoder, one motor revolution will equal 1,048,576 absolute position.

If servo drive is in Relative servo mode, the position after power up or Reset is zero.

If servo drive is in Absolute servo mode, the position after power up or Reset is single turn or multi-turn absolute zero.

Positive position means motor is in CW direction relative to zero position.

Motor absolute position range is $\pm 134,217,728$. If position exceeds this, position will roll back to 0.

The servo motor position can be read whenever the encoder is correctly connected to the servo drive. Even when the servo drive is Disabled or Faulted/Alarmed and motor is free/coasting, the motor position can be read.

Always read the high bytes (0x1d) first, then the low bytes (0x1e). Read low bytes after high bytes as soon as possible to maintain best position reference. Position register update rate is 300us.

Timing / Position Synchronization Example using 16-bit encoder:

Time [ms]	Motor Position [pts]	Operation
0	69514	
25	75201	0x1d register read command sent. Servo drive stores current position 75201 = 0x000125C1 and returns high bytes 0x0001
50	85218	0x1e register read command sent. Servo drive returns stored position low bytes 0x25C1
75	98521	
...		
125	129921	0x1d register read command sent. Servo drive stores current position 129921 = 0x0001FB81 and returns high bytes 0x0001
150	148759	0x1e register read command sent. Servo drive returns stored position low bytes 0xFB81
175	160258	

Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
40032	1f	Motor Torque	R	Int16	±981

Details

This register contains the reference value of instantaneous output current from servo drive to motor. 981=peak output current of servo drive. Value is positive when current/torque is applied in CCW direction. Torque update rate is 300us.

Example:

- Servo drive used = DYN5-H01 Frame. Peak output current = 20A
- Servo motor used = 11A-DHT-A6HK1. Torque coefficient = 0.774Nm/A
- Motor Torque read value = 0xFF63 = -157
- $157 / 981 = 0.16 * 20A = 3.2A$
- $3.2A * 0.774Nm/A = 2.48Nm$ applied in CW direction since reading is negative

Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
40033	20	Torque Limit Parameter	RW	Int16	1~127

Details

This register contains servo drive parameter Torque Limit [Pr.04]. Controller can read and write this parameter as necessary.

This parameter is saved into servo drive EEPROM so has a 1 million time read/write lifecycle.

Change into this parameters is effective after and power cycle or Rest command.

The TorqueLimit parameter is used to limit the current output from the servo drive, directly proportional to the peak current output of the servo drive. A setting of 127 means the limit is turned off and servo drive can output peak current. A setting of 64 means the current output is limited to 50% peak current.

DYN5-H01 servo drive example:

Torque Limit Parameter Setting	Peak Output Current
127	20A
64	10A
10	1.57A

The current sent to the motor can be converted to torque using the servo motor Torque Constant (Torque Coefficient) specification. For example, the 880-DXT motor has a Torque Constant specification of 0.56Nm/A. If the servo drive sends 10A to the motor, the motor outputs 5.6Nm torque at the shaft.

Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
40034	21	Drive Reset	W	Int16	0xABCD

Details

This register is used to send servo drive Reset command.

If this register is set to 0xABCD, servo drive issues Reset command. Used to clear servo drive faults or reset motor position. Register resets to 0x0000 after reset.

After Reset command is sent to servo drive, servo drive re-initializes control and communication so Modbus TCP/IP communication will be down for approximately 5 seconds. Wait at least 10 seconds after sending Reset command before sending new Modbus TCP/IP commands to servo drive.

After Reset command wait at least 10 seconds before sending additional Reset commands.

Setting register to any other value than 0xABCD will return 03 Exception response.

Register# (Modbus Decimal)	Modbus Address# [hex]	Register Map	Access	Data Type	Data Range
	0xFFFE	Diagnostic Counter	R	Int16	0~255

Details

This register contains servo drive internal unsigned 8-bit counter used for testing and diagnostics. Register value increments by 1 each time it is read. Rolls back to 0 after 255.

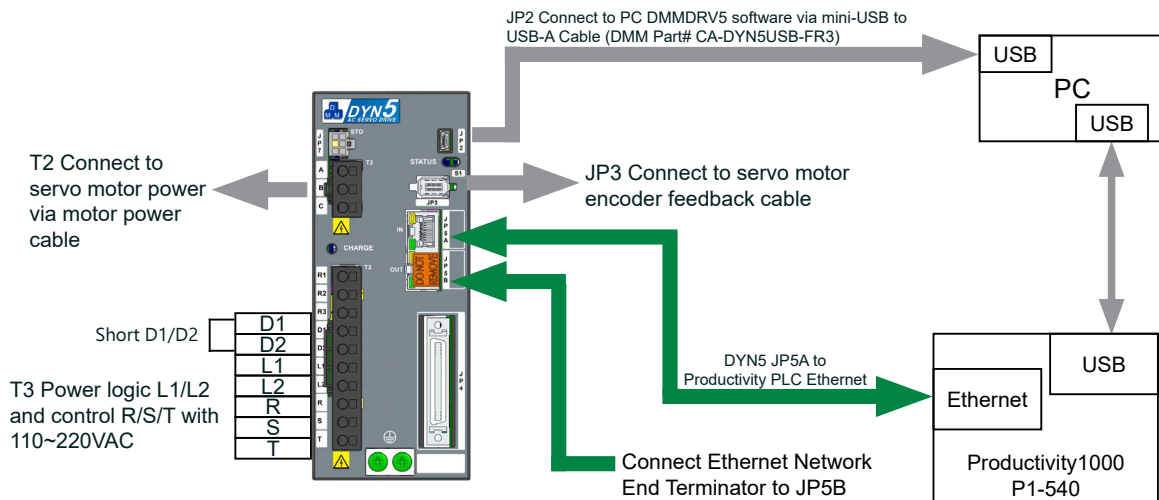
When initially developing or testing Modbus TCP/IP communication, it is recommended to read from this register to check correct communication since read data can be easily identified by incrementing response.

Section 5. PLC Communication Example

This section will demonstrate how to setup Modbus TCP/IP communication with a PLC. And also instructions to send and read speed and position command with DYN5 servo drive.

The PLC used is Automation Direct Productivity1000 PLC Model# P1-540. These instructions are identical for all Productivity PLC models with Modbus TCP Client capability.

◆ Hardware Layout



◆ Network Address

In this section, the below IP address settings are used for each hardware device. All devices are connected directly on a local network. No ethernet switches are used.

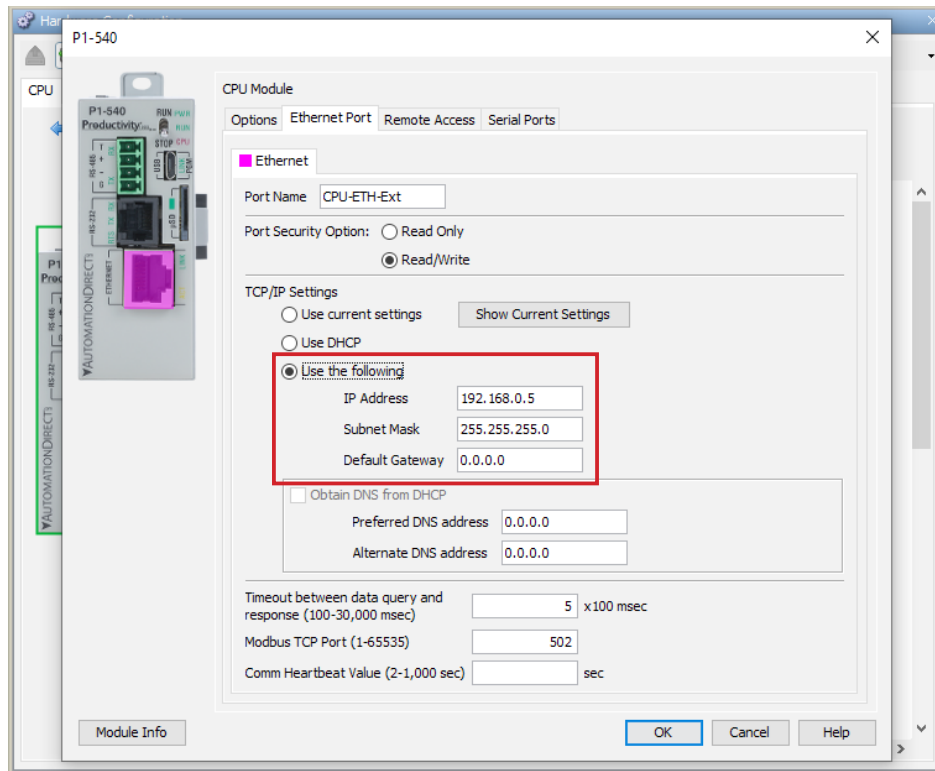
Hardware Device	IP Address	Subnet Mask
PC Computer	192.168.0.10	255.255.255.0
DYN5 Servo Drive	192.168.0.20	n/a
Productivity1000 PLC	192.168.0.5	255.255.255.0

Before starting PLC communication setup in Section 5.1, ensure the servo drive Modbus TCP/IP communication has been tested following Section 3.

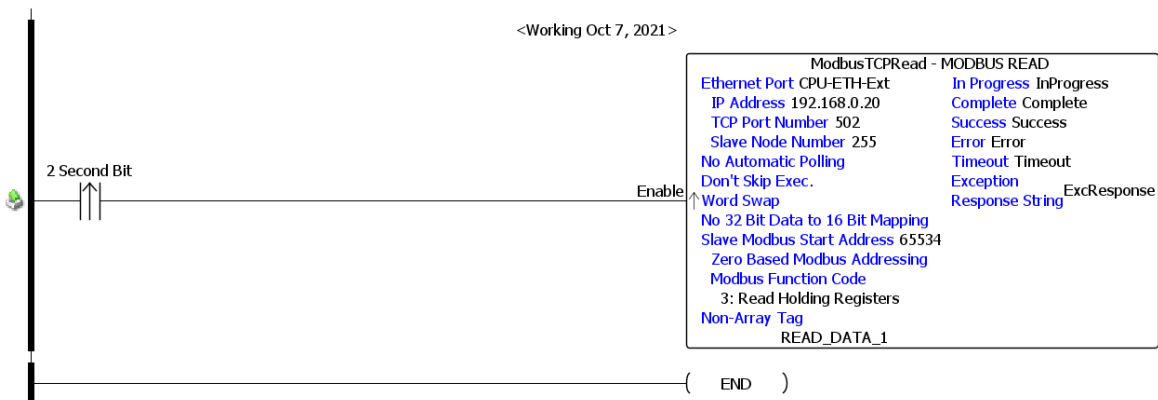
5.1 Initial Communication - Read from register 0xFFFE Diagnostic Counter

(1) Ensure that the DYN5 servo drive is set into Ethernet Modbus TCP/IP communication mode and network addressing of each device is set appropriately. Connect all devices per Hardware Layout diagram above and power up each device.

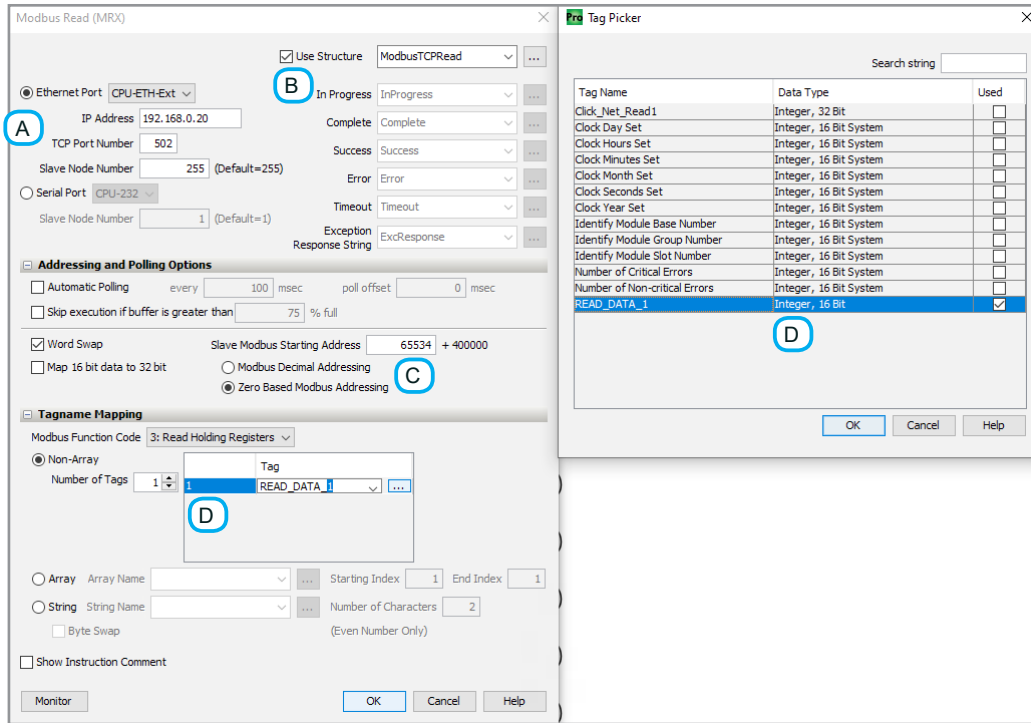
(2) Open the Productivity PLC programming software and establish communication with PC. Unless otherwise noted in these instructions, all settings in the PLC are factory default settings. Set the PLC TCP/IP address settings as below:



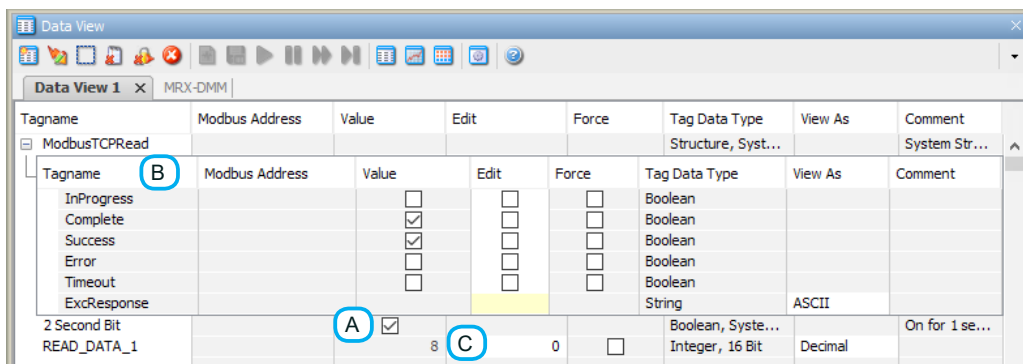
(3) Create a New Project in PLC and write an instruction to read servo drive Diagnostic Counter register 0xFFFE. Shown below is this sample program. In this program, the PLC built-in 2 second bit toggles high/low at 2 second period, triggering the ModbusTCPRead instruction every 2 seconds.



- (4) The details of the ModbusTCPRead instructions is show as below.
- A. Select Ethernet Port and input servo drive IP address. TCP Port number is always 502. Slave Node Number is not relevant.
 - B. Select “Use Structure” to automatically create Tag Names for communication status.
 - C. Select Zero Based Modbus Addressing and input Slave Modbus Starting Address as 65534 corresponding to servo drive register 0xFFFFE.
 - D. Create a new 16-Bit tag name to store the read data from servo drive register 0xFFFFE



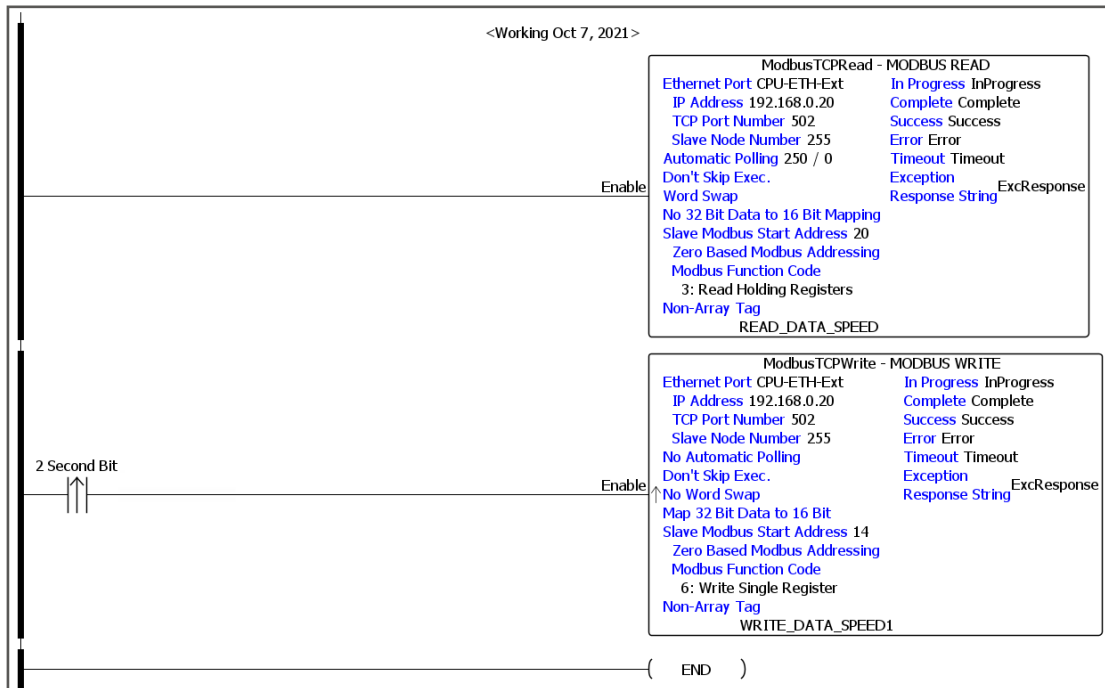
- (5) Save and Write the program into PLC and Run. Open the Data View window and input the Tagname as shown below to monitor the data.
- A. “2 Second Bit” Value will toggle on/off at 2 second interval
 - B. “ModbusTCPRead” Tagnames will indicate communication status. “Complete” and “Success” should be ON indicating successful communication with the servo drive.
 - C. READ_DATA_1 will increment by 1 each time “2 Second Bit” toggles on. This indicates correct response from servo drive register 0xFFFFE and correct communication between PLC and servo drive.



5.2 Speed Command and Speed Feedback Example

In this example, we will send a Speed command to the servo drive and passively monitor the motor speed using Automatic Polling.

The overall program logic is shown below. The Modbus Read function is polled automatically using Automatic Polling Opting in PLC. The Modbus Write function is run every 2 seconds triggered by the "2 Second Bit".



The two Modbus TCP/IP Read and Write Instruction details are shown below.

Modbus Read:

1. Set drive IP address and Port settings
2. Select Tag name for instruction status
3. Select Automatic Polling and set interval at 250ms
4. Select Zero Based Modbus Addressing. Read from register 20 = 0x14 Motor Speed
5. Function code select 03 Read Holding Register
6. Create Tag to store read speed data. Data type should be 16 Bit Signed as speed could be positive CW or negative CCW.

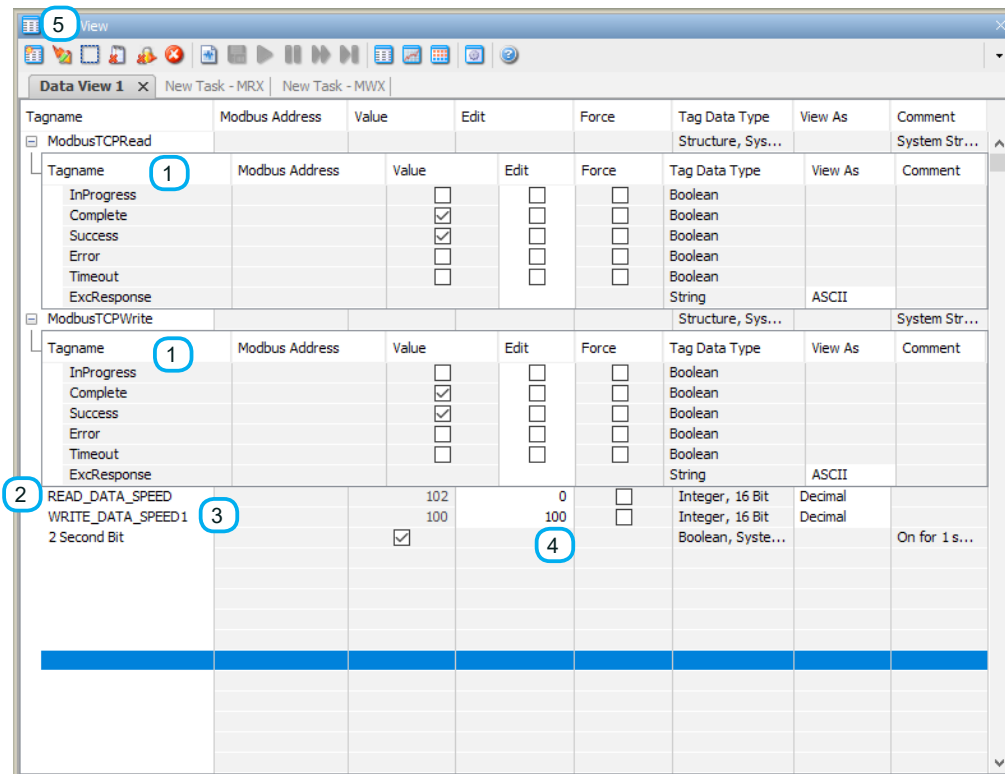
Modbus Write:

1. Set drive IP address and Port settings
2. Select Tag name for instruction status
3. Select Zero Based Modbus Addressing. Write to register 14 = 0x0e Turn_Const-Speed command register
4. Function code select 06 Write Single Register
5. Create Tag to store read speed data. Data type should be 16 Bit Signed as speed could be positive CW or negative CCW.

Save and Write the program into PLC and Run. Open the Data View window and input the Tagname as shown below to monitor the data.

The Modbus Read instruction is running Automatic Polling so the instruction is called every 250ms. Every 250ms, the PLC is reading from register 0x14 Motor Speed and storing the data into READ_DATA_SPEED tag.

1. “ModbusTCPRead” and “ModbusTCPWrite “ Tagnames indicates the two instruction status. Both tags turns ON Complete and Success bits to indicate successful communication.
2. READ_DATA_SPEED tag is read data read from register 0x14 which contains the motor speed. This data is the actual motor speed in rpm units.
3. WRITE_DATA_SPEED1 tag is the data used to send motor speed command into register 0x0e.
4. Edit WRITE_DATA_SPEED1 tag to desired motor speed command
5. Click Send Edit (while WRITE_DATA_SPEED1 row is selected) to send edit to PLC
6. Following program instruction logic, Modbus Write is executed next time 2 Second Bit toggles ON. When Modbus Write is executed, new speed command is sent to servo drive.
7. READ_DATA_SPEED 2 updates reflects new motor speed.



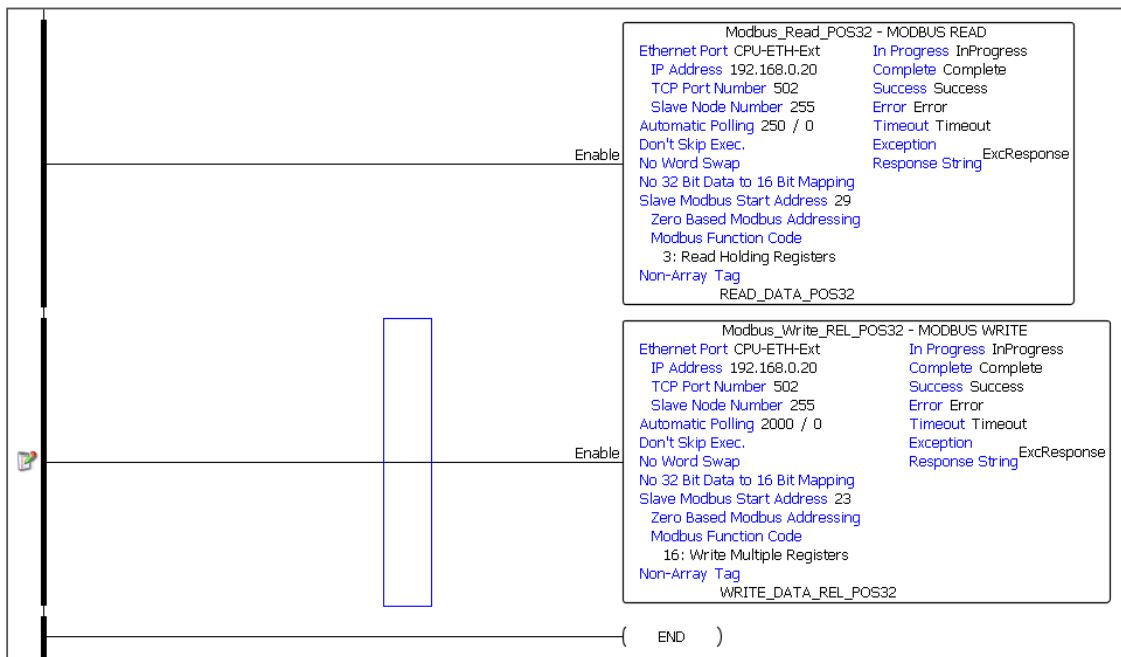
5.3 Profile Relative Position Command Example

In this example, we will send a Profile Relative Position command to the servo drive and passively monitor the motor position using Automatic Polling.

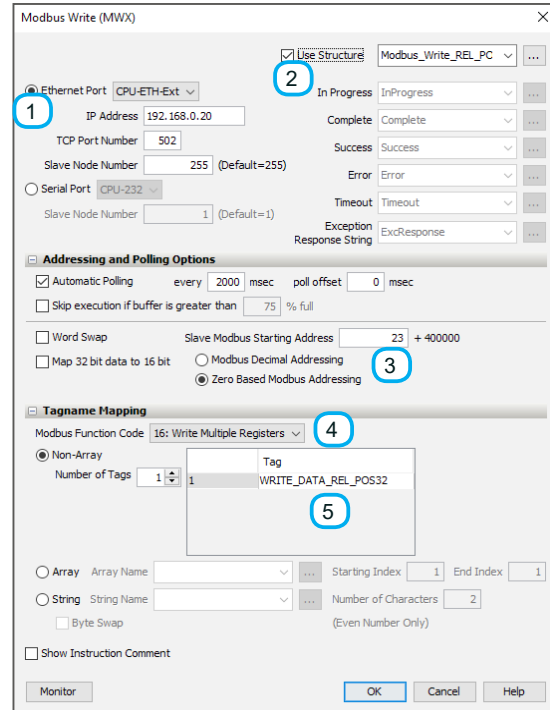
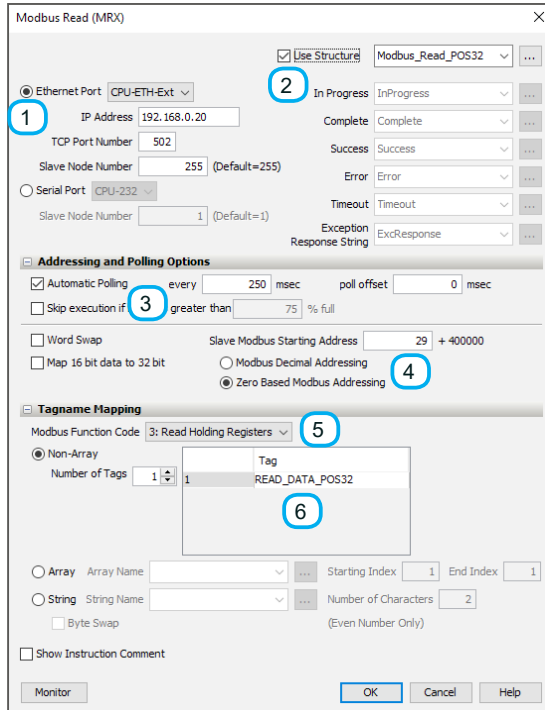
For this example, we will use a motor with 16-bit encoder (65,536ppr). With GEAR_NUM parameter [Pr.19] set to 16384 for 1:1 command distance ratio, meaning a position command of 65536 will move the motor 1 revolution. User can also change the High_Speed [Pr.07] and High_Acceleration [Pr.11] parameters to control the motion profile speed and acceleration. See Appendix B for motion profile details.

GEAR_NUM [Pr.19], High_Speed [Pr.07] and High_Acceleration [Pr.11] parameters do not need to be set for every move command. The servo drive will use the last saved parameter in the memory to generate the motion profile. These parameters only need to be changed when necessary.

The overall program logic is shown below. The Modbus_Read_POS32 instruction is polled automatically using Automatic Polling Opting in PLC at 250ms interval. The Modbus_Write_REL_POS32 instruction is polled automatically using Automatic Polling Opting in PLC at 2s interval.



The two Modbus TCP/IP Read and Write Instruction details are shown below.



Modbus Read:

1. Set drive IP address and Port settings
2. Select Tag name for instruction status
3. Select Automatic Polling and set interval at 250ms
4. Select Zero Based Modbus Addressing. Read from register 29 = 0x1d Motor Absolute Position
5. Select Function Code 03 Read Holding Register
6. Create Tag to store read speed data. Data type should be Int 32-bit since motor absolute position is 32-bit size.

Modbus Write:

1. Set drive IP address and Port settings
2. Select Tag name for instruction status
3. Write to register 23 = 0x17 Turn_Const-Speed command register
4. Select Function Code 16 Write Multiple Registers
5. Create Tag to store read speed data. Data type should be Int 32-bit. This will set the instruction so it automatically sends the 32-bit position command as two 16-bit data into registers 0x17 and 0x18.

Both data tags READ_DATA_POS32 and WRITE_DATA_REL_POS32 should be Int 32 Bit Type.

Name	Type	System ID	I/...	R...	Nu...	R...	Inl...	Wi...	M...	M...	F...	In...	Inl...	C...	Re...	Def...	In ...
READ_DATA_POS32	Integer, 32 Bit	S32-000002						0						0		De...	
WRITE_DATA_REL_POS32	Integer, 32 Bit	S32-000004						0						0		De...	

Save and Write the program into PLC and Run. Open the Data View window and input the Tagname as shown below to monitor the two data READ_DATA_POS32 and WRITE_DATA_REL_POS32.

Both Modbus Read and Write instruction is called by Automatic Polling.

1. The Read is called every 250ms, so motor position is read and updated into READ_DATA_POS32 every 250ms.
2. Edit a position value into WRITE_DATA_REL_POS32. In this example, 5000 is used as relative position command.
3. While WRITE_DATA_REL_POS32 row is selected, press click Send Edit to send edit new value to PLC.
4. After sending the edit, the Write command with data 5000 is called every 2 seconds. Moving the motor 5000points in positive direction every 2 seconds. The 16-bit encoder is used with GEAR_NUM parameter set to 16384 so a 5000 command will move the motor 5000/65536revolutions in CW direction each time. Negative data command will move motor in CCW direction.
5. READ_DATA_POS32 1 updates continuously at 250ms to reflect current motor position.

Tagname	Modbus Address	Value	Edit	Force	Tag Data Type	View As	Comment
WRITE_DATA_REL_POS32		5000	5000	<input checked="" type="checkbox"/>	Integer, 32 Bit	Decimal	
READ_DATA_POS32		50000	0	<input type="checkbox"/>	Integer, 32 Bit	Decimal	

Section 6. Servo Drive Communication Response Time

When designing time-sensitive applications, consider below timing information regarding the data in the modbus registers.

◆ General Modbus Register Update Rate

- General update time delay between DYN5 servo drive main servo CPU and modbus module is 300 microseconds (us).
- Write commands into modbus register become effective in the servo loop after $\leq 300\text{us}$. This is true for all position, speed or torque commands. Also true for parameters that can be changed on the fly.
- Data in Modbus register (to be Read) may be delayed by $\leq 300\text{us}$. Meaning data in the Modbus register lags actual servo drive data no more than 300us.

Appendix A - DMMDRV5 Communication Setup

This section will outline quick communication setup and jog of servo motor from DMMDRV5 program. Refer to DMM manual# DSFEN_A15 for full feature specification of DMMDRV5 program.

■ DMMDRV5 System Requirements

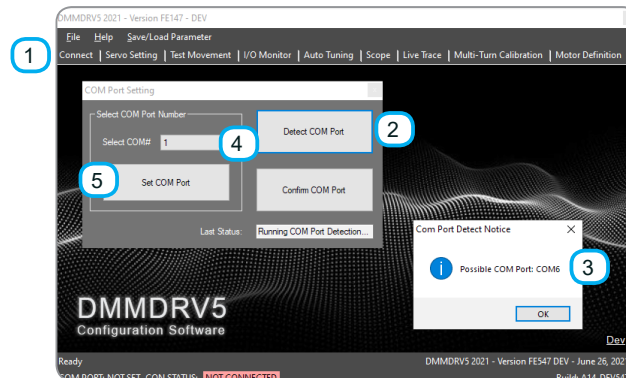
Operating System: Windows XP SP3 or higher *Recommended: Windows 7 or Higher

Processor: Pentium 1 GHz or higher RAM: 512 MB or more

Minimum disk space: 200MB Display: Minimum 1280×720

(1) Continuing from Section 3.4.2, the servo drive should be powered up and both STATUS and CHARGE LED should be lit Green. If STATUS LED is not solid Green, the DYN5 servo drive is faulted and operator can continue below instructions to check fault code.

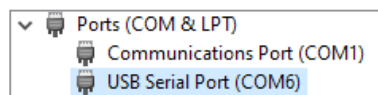
(2) Download and install DMMDRV5 program, available from DMM website. Connect servo drive to PC using USB cable (Part# CA-USBMA-FR3).



(3) Run DMMDRV5 program. Click Connect (1) to open COM Port Setting window to establish connection with servo drive. Click Detect COM Port (2) to automatically detect COM port number. Program will prompt with COM port number of servo drive (3). Select corresponding COM port number (4) and click Set COM port (5). Program should indicate successful connection with servo drive.

Notes:

A. If program is unable to detect COM port, check COM port manually in PC Device Manager. The DYN5 servo drive will appear as “USB serial port” under Ports (COM&LPT) section (COM6 as example):



B. DMMDRV5 program supports COM1~COM8. If COM port number is above 8, manually change COM port number by right clicking USB Serial Port->Properties->Port Settings Tab->Advanced. COM port number can be manually changed on top left.

C. When configuring multiple servo drives, the PC may assign different COM port number to each individual servo drive. In this case, manually change COM port number for each servo drive to same COM port number. For OEM applications requiring high volume configuration, contact DMM representative and special software will be provided to automatically configure COM port to all equal.

Appendix B - Profile Position Command Trajectory Calculator

When sending Profile Absolute or Relative position command to the servo drive, refer to the below specification to calculate the motion profile.

The Max Acceleration, Max Speed, and GEAR_NUM parameters are used for generating the motion profile. The DYN5 servo drive also applies a smoothing filter to the acceleration profile to generate a S-Curve trajectory. This Profile motion and S-curve trajectory reference is applicable to both Go_Absolute_Pos and Go_Relative_Pos commands.

Dynamic Target Position Update (DTPU) is also applied to all Profile position commands. See Appendix C for DTPU specification.

GEAR_NUM = GEAR_NUM parameter [Pr. 19]

MaxSpd = High Speed parameter. [Pr. 07]

MaxAcl = High Acceleration parameter. [Pr. 11]

$$\text{Profile Motor Speed [rpm]} = \frac{(\text{MaxSpd}+3) \cdot (\text{MaxSpd}+3)}{16} * 12.21 * \text{Gear Ratio}$$

$$\text{Gear Ratio} = \frac{4,096}{\text{GEAR_NUM}}$$

$$\text{Profile Motor Acceleration [rpm/s]} = \text{MaxAcl} * 635.78 * \text{Gear Ratio}$$

$$\text{Motor Movement Distance} = \text{Command Position} * \text{Gear Ratio} * 4$$

Example using 16-bit encoder (65,536pts/rev):

Set parameter	Motion Profile
Gear_Num = 4096	Gear Ratio = 1
MaxSpd = 48	Maximum Motor Speed = 1985 rpm
MaxAcl = 30	Maximum Motor Acceleration = 19073 rpm/s
Command Position = 140,000 pts	Motor Movement = 560,000 pts = 8.545 revolutions CW

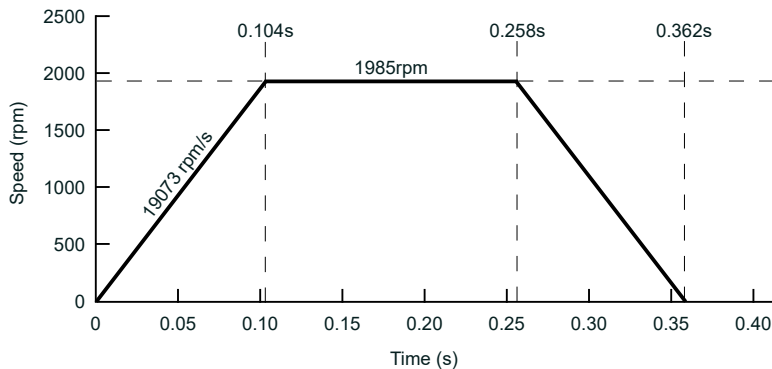
Motion Profile:

Acceleration Time = 0.104 s

Distance During Acceleration = 1.72 rev

Constant Speed Travel Time = 0.154 s

Total Profile Motion Time = 0.362 s



Appendix C - DTPU Dynamic Target Position Update Specification



The DYN servo drive's built in S-Curve generator is able to update the target position instantaneously regardless of whether the current command position has completed or not. As soon as a new command position is received, the servo drive immediately updates the servomotor target to the newest position. This function is applicable to both relative (incremental) and absolute positioning for all linear, or arc path profiles.

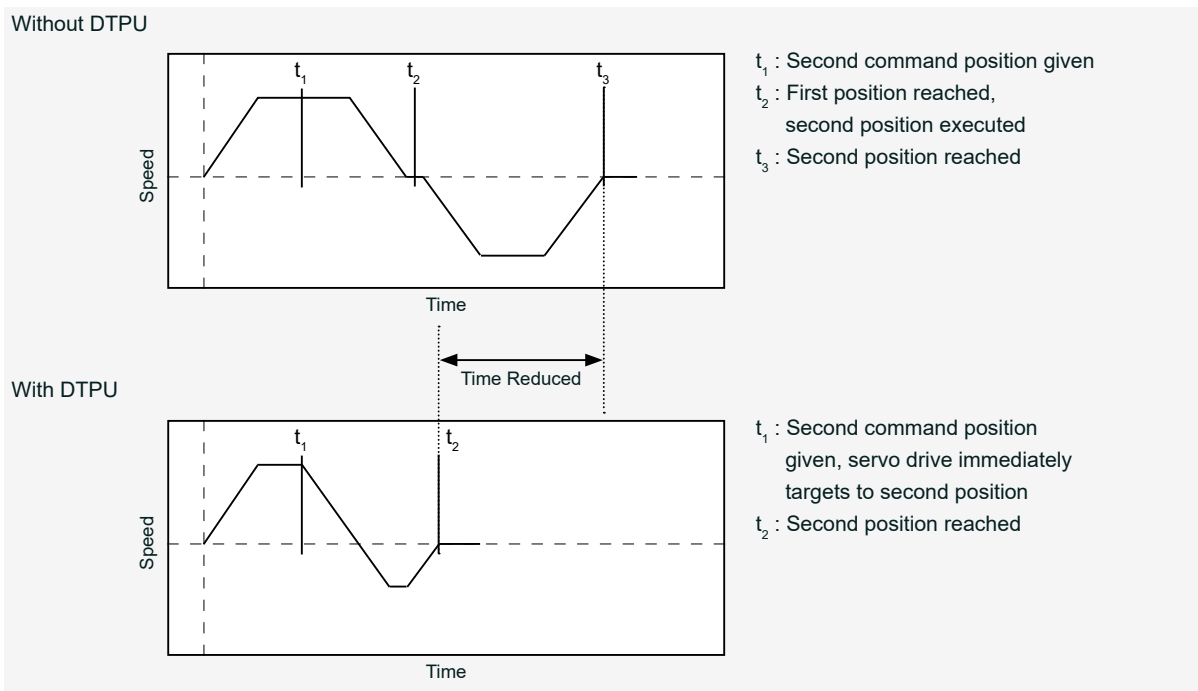
Without Dynamic Target Position Update DTPU technology, the servo drive must wait until the first, or current position command is completed before executing the next one. This limits the rate at which the motor position can be updated and can also have detrimental effects on safety for the machine and the operator. With DTPU technology, the servo drive is always under active command from the controller, allowing much faster cycle time and higher universal efficiency.

The servo drive also applies a curved acceleration command to the S-Curve to maintain smoothest servo motor motion. At each S-Curve "transition" point, the normally rigid path is curved into smooth speed transitions.

◆ Positioning Efficiency

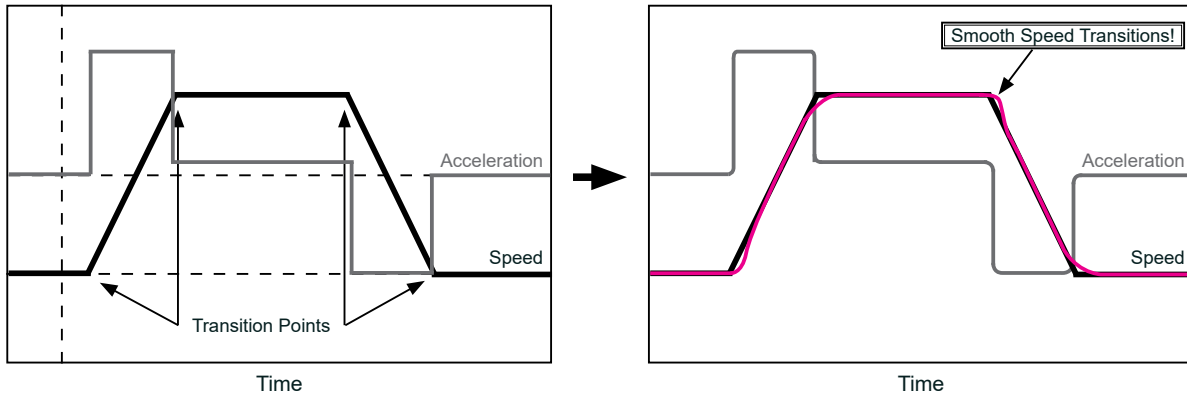
When the axis is command to a new position, the servo drive immediately updates the target position and generates new S-Curve profile to reach new target position. Without DTPU technology, the axis must first finish its current command before executing the next one, causing a delay in the overall positioning time.

This also allows more flexibility in programming and path planning as the controller no longer needs to wait until a particular movement is finished before calculating the succeeding one. Robotic movements can be controlled and commanded in real-time, significantly simplifying kinematic motion planning requirements on the controller. Machine-level trajectory planning can almost be eliminated.



◆ S-Curve Acceleration/Deceleration

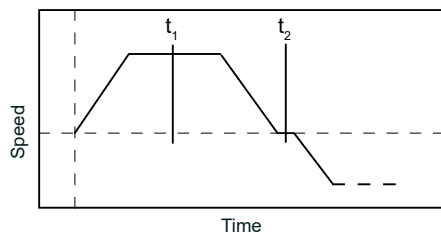
The DTPU algorithm also applies a curved acceleration to maintain smooth motion. At each S-Curve transition point, the acceleration/deceleration is curved at the edges so speed is smoothly changed. This decreases motor vibration. The smoothing is applied relative to total command movement so overall distance and position accuracy is not affected.



◆ Safety

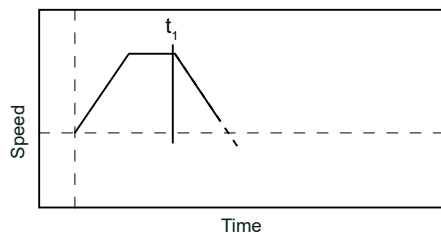
Dynamic Target Position Update DTPU allows the axis to be commanded as soon as a safety hazard or warning is detected. This means protection measures can be executed immediately. Without DTPU, the axis must finish the current positioning before executing protection measures.

Without DTPU



t_1 : Safety warning/hazard detected, axis commanded to retract
 t_2 : Current positioning reached, axis commanded to retract

With DTPU



t_1 : Safety warning/hazard detected, axis commanded to retract
 t_2 : Axis immediately retracts to safe position

Warranty and Liability

■ Warranty

Products from DMM Technology Corp. are supported by the following warranty.

- 1-year from the date of product received by customer or 14 months from the month of original shipment.

Within the warranty period, DMM Technology Corp. will replace or repair any defective product free of charge given that DMM Technology Corp. is responsible for the cause of the defect. This warranty does not cover cases involving the following conditions:

- The product is used in an unsuitable or hazardous environment not outlined in this manual, resulting in damages to the product.
- The product is improperly handled resulting in physical damage to the product. Including falling, heavy impact, vibration or shock.
- Damages resulting from transportation or shipping after the original factory delivery.
- Unauthorized alterations or modifications that have been made to the product.
- Alterations have been made to the Name Plate of the product
- Damages resulting from usage of the product not specified by this manual.
- Damages to the product resulting from natural disasters.
- The product has cosmetic alterations.
- The product does not conform to the original factory manufactured standards due to unauthorized modifications.

■ Liability

Use, operation, handling and storage of the DYN5 AC Servo Drive is the sole responsibility of the customer. Any direct or indirect commercial loss, commercial profit, physical damage or mechanical damage caused by the DYN5 AC Servo Drive is not responsible by DMM Technology Corp. The features and functionality of the product should be used with full discretion by the operator.

Manual Disclaimer

■ Manual Disclaimer

DMM Technology Corp. constantly strive to improve its product performance and reliability. As such, the contents and information in this manual may be changed without notice to reflect corrections, improvements or changes to the product. Refere to the DMM website to download latest version of this manual.

Manual Revisions

Published	Revision	Internal Reference	Section	Revised Content
October 2021	SL425-10A	--	--	First Edition

DYN5 Series

AC Servo Drive

AC SERVO DRIVE

Modbus TCP/IP Specification

MANUAL CODE: DYN5-MBTCP-SL425-10A

REVISION: 10A

RELEASE DATE: October 2021

Electronic Version

Copyright © 2021 DMM Technology Corp.

Published In Canada A4P

DMM TECHNOLOGY CORP.
120 - 21320 Gordon Way Richmond, British Columbia V6W1J8 Canada

PHONE: +1 (604)-370-4168
FAX: +1 (604) 285-1989
WEB: <http://www.dmm-tech.com>
SALES: sales@dmm-tech.com
INFO: info@dmm-tech.com

