



Dynamic Motor Motion
Technology Corporation

DYN232M
DYN AC SERVO SYSTEM - RS232 MOTION



adaptive TUNING II



DYN4 Series

AC SERVO DRIVE INSTRUCTION MANUAL

MODEL NUMBER:

DYN4 - □□□ A TYPE A - PULSE / ANALOG / RS232

DYN4 - □□□ B TYPE B - MODBUS RTU RS485

DYN4 - □□□ C TYPE C - CAN

MANUAL CODE: DYN4MS-ZM7-A10A

REVISION: A1.0A 02

RELEASE DATE: DECEMBER, 2018

■ Safety Notice ■

The user or operator should read through this manual completely before installation, testing, operation, or inspection of the equipment. The DYN4 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 following warning convention is used for certain specifications.



■ Notations Used ■

All specification and units of measurement used in the manual are in METRIC:

Mass: Kilogram [kg]

Length: Millimeter [mm]

Time: Seconds [s]

Temperature: Celsius [°C]

■ Standards Compliance ■

| | |
|----|--|
| CE | Machinery Directive 2006/42/EC Low Voltage Directive 2014/35/EU Electromagnetic Compatibility 2014/30/EU |
| | EN12100:2010 EN 60034-1:2010; EN 60204-1:2006/AC: 2010 EN 61000-6-1: 2007; EN 61000-6-2:2005/AC: 2005 |

Manual Contents

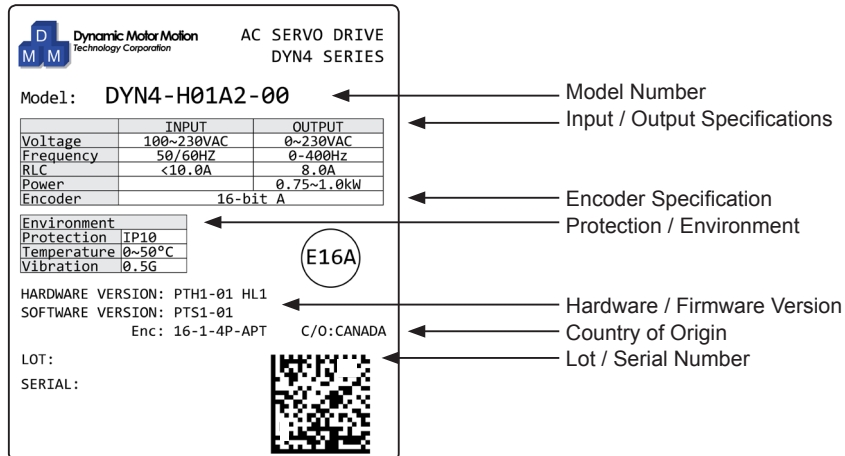
| | |
|--|-----------|
| ■ Safety Notice ■ | 2 |
| ■ Notations Used ■ | 2 |
| ■ Standards Compliance ■ | 2 |
| Manual Contents | 3 |
| 1 General Specification | 5 |
| 1.1 Servo Drive Name Plate | 5 |
| 1.2 Model Designation | 5 |
| 1.3 Servo Drive Overall Specification | 6 |
| 1.4 Circuit Block Diagram | 7 |
| 1.5 Encoder Specification | 7 |
| 1.6 Servo Motor Pair | 8 |
| 2 Connection and Wiring | 9 |
| 2.1 DYN4 Servo Drive Body Layout | 9 |
| 2.2 Connector Terminal and Signal Specification | 10 |
| 2.3 JP4 Main I/O Terminal Details | 11 |
| 2.3.1 JP4 Specification | 11 |
| 2.3.2 JP4 Interface Circuit Examples | 15 |
| 2.3.3 JP4 Consolidated Interface Circuit | 19 |
| 2.4 JP5 Encoder Output Terminal Details | 20 |
| 2.5 Main Circuit Wiring | 21 |
| 2.6 Holding Brake Circuit Wiring | 26 |
| 3 Start - Up | 27 |
| 3.1 Preparation and Mounting | 27 |
| 3.2 Multiple Drive Spacing | 27 |
| 3.3 Grounding | 27 |
| 3.4 Power Up Timing Chart | 28 |
| 3.5 Software Communication | 31 |
| 4 Operation | 33 |
| 4.1 Position Servo Mode | 33 |
| 4.2 Speed Servo Mode | 37 |
| 4.3 Torque Servo Mode | 39 |
| 4.4 RS232 Command Input Mode | 40 |
| 4.5 Encoder Output | 40 |
| 4.6 Servo Drive Monitor Outputs | 41 |
| 5 Parameters and Tuning | 43 |
| 5.1 Parameters Outline | 43 |
| 5.2 Servo Drive Gain Tuning | 45 |
| 6 Maintenance | 47 |
| 6.1 Alarm Specifications | 47 |
| 6.2 Servo Drive Maintenance | 48 |
| 7 RS232 Communication Protocol | 49 |
| 7.1 Interface and Format | 50 |
| 7.2 Packet Definition | 52 |
| 7.3 Drive Configuration and Status Register | 56 |
| 7.4 Common Function Details | 57 |
| 7.5 Dynamic Target Position Update (DTPU) | 59 |
| 7.6 Packet Structure Examples | 61 |
| 7.7 Application Examples | 62 |
| 7.8 RS485 Network | 66 |
| 7.9A Appendix : C++ Code for Serial Communication Protocol | 67 |
| 8 Modbus RTU (RS485) Communication | 74 |
| 9 CAN Communication | 75 |
| Appendix | 76 |
| Appendix A - Servo Drive Dimensions | 76 |
| Appendix B - Application Note# AP15-48 DYN4 Servo Drive - AC Power Supply Guidelines | 77 |
| Appendix C - Analog Input Fine Adjustments | 78 |
| Warranty and Liability | 79 |
| Product and Manual Disclaimer | 80 |
| ■ Document Revisions ■ | 81 |

This page is intentionally blank

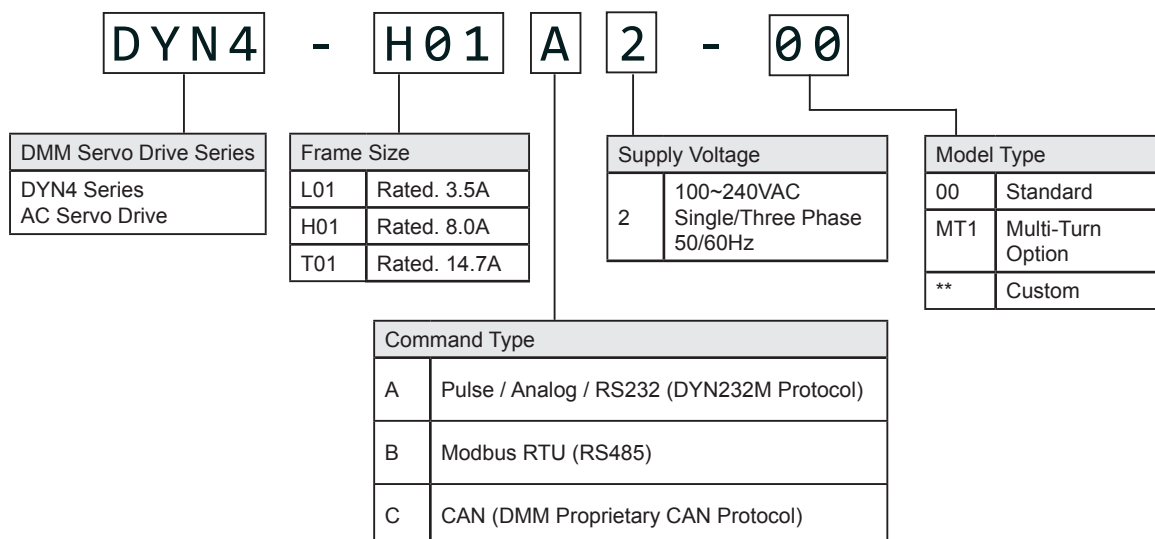
1 General Specification

1.1 Servo Drive Name Plate

*Note the name plate is region specific and may vary between each region model.



1.2 Model Designation

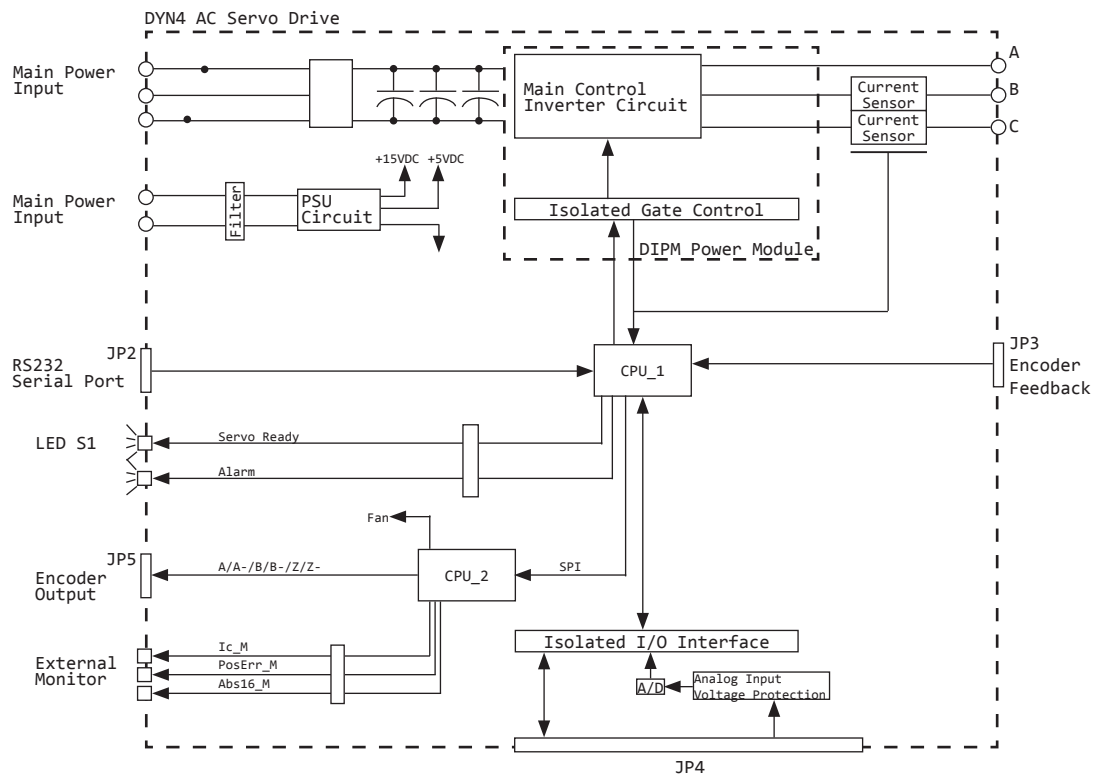


1.3 Servo Drive Overall Specification

| Model: DYN4 - □□□ - A2 | | L01 | H01 | T01 |
|----------------------------|---------------------------|--|------|-------|
| Main Circuit | Rated Voltage | Single-phase or Three-phase 110~240VAC ± 10% 50/60Hz 300VAC Peak | | |
| | Rated Current | 3.5A | 8.0A | 14.7A |
| Control Logic Circuit | Rated Voltage | Single-phase 110~240VAC ± 10% 50/60Hz | | |
| | Rated Current | 0.2A | | |
| Servo Motor Control Method | | SVPWM | | |
| Switching Circuit | | DMM DIPM Power Module | | |
| Dynamic Brake | | Integrated non-adjustable | | |
| Encoder Feedback | | 16-bit [65,536ppr] Absolute Serial, Single or Multi-Turn | | |
| Encoder Output | | A/B/Z Quadrature Differential Line Driver | | |
| Protection Functions | | Over-Current, Over-Voltage, Under-Voltage, Temperature, Over-Power, Position Lost Follow, Encoder Error, CRC Error | | |
| Position Servo | Command Reference Pulse*1 | Pulse+Sign, A/B Phase Quadrature 90° Phase Differential, CW+CCW | | |
| | Max. Input Frequency | 500kHz | | |
| | Input Voltage | 5VDC ± 5% (Higher voltage available as option) Over drive photocoupler diode | | |
| | Electronic Gear Ratio | Set by parameter 500 to 65,536 ppr | | |
| | On Position Output Range | 0 ~ 508 pulse | | |
| Speed Servo | Speed Control Range | 0:5000 | | |
| | Input Reference Voltage | -10VDC ~ +10VDC 3,000rpm at ± 5VDC | | |
| | Resolution | 1rpm | | |
| | Max Input Voltage | ± 12VDC | | |
| Torque Servo | Input Reference Voltage | -10VDC ~ +10VDC | | |
| | Resolution | 5mA motor coil current | | |
| | Max Input Voltage | ± 12VDC | | |
| Environment | Protection | IP10 | | |
| | Cooling | Chassis Heat Sink | | |
| | Operation Temperature | 0~55°C | | |
| | Storage Temperature | -20 ~ 65°C | | |
| | Max. Operation Humidity | 95RH% (no dew) | | |
| | Max. Storage Humidity | 95RH% (no dew) | | |
| Standards Compliance | | CE, ROHS | | |

Note: 1. CW+CCW Pulse format optional

1.4 Circuit Block Diagram



1.5 Encoder Specification

■ Encoder Options

The servo drive should be selected according to the encoder option used. Note the same encoder option must also be selected for the servo motor as well.

| Encoder Model# | Type | Resolution | Data Type | Interface Type | Sensor | Supply Voltage | DYN4 Pair | Status |
|----------------|----------------------------------|---|---------------|------------------------------|----------|----------------|----------------|--------|
| ABS-16-00 | Absolute Single-Turn | 16-bits [65,536ppr] | 4-Wire Serial | Differential Driver/Receiver | Magnetic | +5VDC | 00 Standard | A |
| ABS-16-MB1 | Absolute Multi-Turn With Battery | 16-bits Single Turn 16-bits Multi-Turn | 6-Wire Serial | Differential Driver/Receiver | Magnetic | +5VDC | MT1 Multi-Turn | A |
| ABS-16-ML1 | Absolute Multi-Turn Battery-Less | 16-bits Single Turn 16-bits Multi-Turn | 4-Wire Serial | Differential Driver/Receiver | Magnetic | +5VDC | MT1 Multi-Turn | A |
| ABS-16-LP1 | Absolute Single-Turn Low Profile | 16-bits [65,536ppr] | 4-Wire Serial | Differential Driver/Receiver | Magnetic | +5VDC | 00 Standard | A |
| ABS-14-00 | Absolute | 14-bits [16,384ppr] | 6-Wire Serial | Differential Driver/Receiver | Magnetic | +5VDC | N/A | L |

A Status = In Production
L Status = Not In Production

■ Applicable Standard Servo Motor Model

DST Series Standard Motors

| Model Number | Output | Voltage | Frame | Rated Torque | Peak Torque | Rated Speed | Peak Speed | Rotor Inertia | DYN4 Servo Drive |
|---------------|-------------|---------|-------|--------------|-------------|-------------------|-------------------|--------------------|------------------|
| | | | | Nm | Nm | min ⁻¹ | min ⁻¹ | kg-cm ² | |
| 405-DST-A6HK1 | 50W | 220V | 40mm | 0.16 | 0.48 | 3000 | 5000 | 0.036 | DYN4-L01 |
| 405-DST-A6HKB | 50W Brake | 220V | 40mm | 0.16 | 0.48 | 3000 | 5000 | 0.044 | DYN4-L01 |
| 410-DST-A6HK1 | 100W | 220V | 40mm | 0.32 | 0.96 | 3000 | 5000 | 0.063 | DYN4-L01 |
| 410-DST-A6HKB | 100W Brake | 220V | 60mm | 0.32 | 0.96 | 3000 | 5000 | 0.076 | DYN4-L01 |
| 620-DST-A6HK1 | 200W | 220V | 60mm | 0.64 | 1.91 | 3000 | 5000 | 0.232 | DYN4-L01 |
| 620-DST-A6HKB | 200W Brake | 220V | 60mm | 0.64 | 1.91 | 3000 | 5000 | 0.278 | DYN4-L01 |
| 640-DST-A6HK1 | 400W | 220V | 60mm | 1.27 | 3.82 | 3000 | 5000 | 0.426 | DYN4-L01 |
| 640-DST-A6HKB | 400W Brake | 220V | 60mm | 1.27 | 3.82 | 3000 | 5000 | 0.511 | DYN4-L01 |
| 880-DST-A6HK1 | 750W | 220V | 80mm | 2.39 | 7.16 | 3000 | 5000 | 1.4 | DYN4-H01 |
| 880-DST-A6HKB | 750W Brake | 220V | 80mm | 2.39 | 7.16 | 3000 | 5000 | 1.63 | DYN4-H01 |
| 11A-DST-A6HK1 | 1.0kW | 220V | 130mm | 4.4 | 12 | 1500 | 3000 | 8.5 | DYN4-H01 |
| 11A-DST-A6HKB | 1.0kW Brake | 220V | 130mm | 4.4 | 12 | 1500 | 3000 | 10.2 | DYN4-H01 |
| 115-DST-A6HK1 | 1.3kW | 220V | 130mm | 8.27 | 23.3 | 1500 | 3000 | 18.9 | DYN4-T01 |
| 115-DST-A6HKB | 1.3kW Brake | 220V | 130mm | 8.27 | 23.3 | 1500 | 3000 | 21.74 | DYN4-T01 |
| 120-DST-A6HK1 | 1.8kW | 220V | 130mm | 11.5 | 28.7 | 1500 | 3000 | 23.8 | DYN4-T01 |
| 120-DST-A6HKB | 1.8kW Brake | 220V | 130mm | 11.5 | 28.7 | 1500 | 3000 | 27.37 | DYN4-T01 |

DHT Series Standard Motors

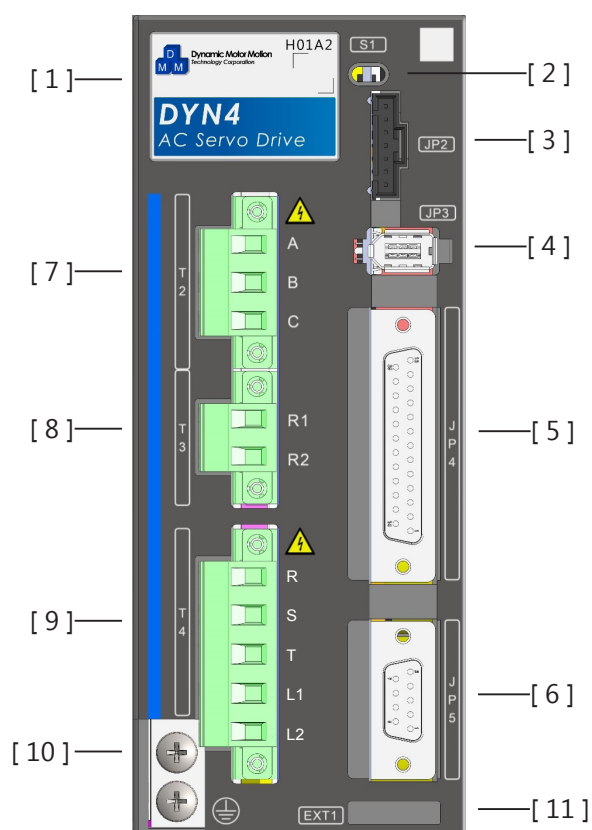
| Model Number | Output | Voltage | Frame | Rated Torque | Peak Torque | Rated Speed | Peak Speed | Rotor Inertia | DYN4 Servo Drive |
|---------------|-------------|---------|--------|--------------|-------------|-------------------|-------------------|--------------------|------------------|
| | | | | Nm | Nm | min ⁻¹ | min ⁻¹ | kg-cm ² | |
| 86L-DHT-A6TD1 | 250W | 60V | NEMA34 | 0.8 | 2.4 | 3000 | 5000 | 0.91 | N/A |
| 57N-DHT-A6TD1 | 400W | 60V | NEMA23 | 1.27 | 3.82 | 3000 | 5000 | 0.426 | DYN4-L01 *1 |
| 57N-DHT-A6TDB | 400W Brake | 60V | NEMA23 | 1.27 | 3.82 | 3000 | 5000 | 0.479 | DYN4-L01 *1 |
| 86N-DHT-A6MD1 | 750W | 150V | NEMA34 | 2.4 | 7.2 | 3000 | 5000 | 2.45 | DYN4-H01 |
| 86N-DHT-A6MDB | 750W Brake | 150V | NEMA34 | 2.4 | 7.2 | 3000 | 5000 | 2.94 | DYN4-H01 |
| 86M-DHT-A6MK1 | 750W | 150V | 86mm | 2.4 | 7.2 | 3000 | 5000 | 2.45 | DYN4-H01 |
| 86M-DHT-A6MKB | 750W Brake | 150V | 86mm | 2.4 | 7.2 | 3000 | 5000 | 2.94 | DYN4-H01 |
| A15-DST-A6HK1 | 1.3kW | 220V | NEMA42 | 5 | 15 | 1500 | 3000 | 6.3 | DYN4-T01 |
| A15-DST-A6HKB | 1.3kW Brake | 220V | NEMA42 | 5 | 15 | 1500 | 3000 | 7.56 | DYN4-T01 |

*1 These motors are 60V class motors and are only compatible with 110/120VAC input on the servo drive.

2 Connection and Wiring

2.1 DYN4 Servo Drive Body Layout

■ DYN4 - L01, H01, T01 Frame



| No. | Name | Description |
|-----|------|---|
| 1 | --- | Face plate and frame code |
| 2 | S1 | Status LED / charge LED |
| 3 | JP2 | PC interface connection RS232, Modbus, CAN connection |
| 4 | JP3 | Encoder feedback connector |
| 5 | JP4 | Main signal I / O |
| 6 | JP5 | Encoder output connector |
| 7 | T2 | Servo motor power output |
| 8 | T3 | Regenerative resistor connector |
| 9 | T4 | Main circuit and control logic circuit input connector |
| 10 | PE | Protective earth / chassis ground |
| 11 | EXT1 | Extended interface connector |

2.2 Connector Terminal and Signal Specification

Connector Details

Signal Layout

- JP2 PC interface connection
Connector Type: 2.54mm Pitch Rectangular
Drive Header: (Molex) 70553-0041
Plug Connector: (Molex) 50-57-9407

| | |
|-------|------------------------------------|
| Pin.1 | Gnd |
| Pin.2 | RS485+, CAN_H |
| Pin.3 | RS485-, CAN_L |
| Pin.4 | NC |
| Pin.5 | RS232 signal input, RxD, TTL/CMOS |
| Pin.6 | RS232 signal output, TxD, TTL/CMOS |
| Pin.7 | +5(V), <10(mA) |

- JP3 Encoder feedback connector
Connector Type: IEEE1394 USB
Drive Header: (3M) 3E106
Plug Connector: (3M) 36210-0100FD

| | |
|---------|-------|
| Pin.1 | +5VDC |
| Pin.2 | Gnd |
| Pin.3~4 | NC |
| Pin.5 | S+ |
| Pin.6 | S- |

- JP4 Main Signal I/O
*Refer to Section 2.3

*Refer to Section 2.3

- JP5 Encoder output connector
*Refer to Section 2.4

*Refer to Section 2.4

- T2 Servo motor power output
Connector Type: Terminal Block
Drive Header: (Phoenix) GMSTB 2.5/3-GF-7.62
Plug Connector: (Phoenix) GMSTB 2.5/3-STF-7.62

| | |
|-------|---------------|
| Pin.1 | Motor A Phase |
| Pin.2 | Motor B Phase |
| Pin.3 | Motor C Phase |

- T3 Regenerative resistor connector
Connector Type: Terminal Block
Drive Header: (Phoenix) GMSTB 2.5/2-GF-7.62
Plug Connector: (Phoenix) GMSTB 2.5/2-STF-7.62

| | |
|-------|-----------------|
| Pin.1 | R1 - Resistor 1 |
| Pin.2 | R2 - Resistor 2 |

- T4 Main circuit input connector
Connector Type: Terminal Block
Drive Header: (Phoenix) GMSTB 2.5/5-GF-7.62
Plug Connector: (Phoenix) GMSTB 2.5/5-STF-7.62

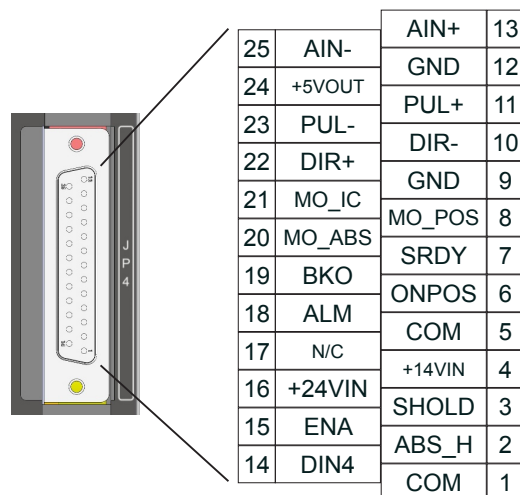
| | |
|-------|------------------|
| Pin.1 | Main circuit R |
| Pin.2 | Main circuit S |
| Pin.3 | Main circuit T |
| Pin.4 | Logic circuit L1 |
| Pin.5 | Logic circuit L2 |

2.3 JP4 Main I/O Terminal Details

2.3.1 JP4 Specification

- JP4 Main signal I/O
Connector Type: D-Sub 25 pin female
Recommended Wire Gauge: 0.2mm² (AWG24)

- Pin Layout (Viewed from drive header side)



- Pin Details

See section 2.3.2 for interface circuit type examples

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|------------------------|-----|-------------------|
| 1 | GND | Internal Signal Ground | NA | NA |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|---|-------|-------------------|
| 2 | ABS_H | <ul style="list-style-type: none">- Absolute encoder auto home- Servo motor position returns to absolute encoder zero position. Servo starts from absolute zero.- Active Low, servo drive runs to absolute zero when input LOW. | Input | A |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|---|-------|-------------------|
| 3 | SHOLD | <ul style="list-style-type: none">- Servo hold input (Positive Direction)- Servo drive ignores command input- Active High | Input | A |

See section 2.3.2 for interface circuit type examples

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|---|-------|-------------------|
| 4 | +14VIN | - External +14VDC input - Supply power for digital input circuit A | Input | A |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|---|--------|-------------------|
| 5 | COM | - Output photocoupler open collector common - For digital output circuit B | Output | B |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|--|--------|-------------------|
| 6 | ONPOS | - Servo on position output - Photo transistor ON when on position, output Low - Servo On Position if motor position error within value set by <i>OnPosRange</i> parameter. - Refer to section 5.1 for parameter setting details | Output | B |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|--|--------|-------------------|
| 7 | SRDY | - Servo Ready Output - Photo transistor ON, output Low when servo ready for command | Output | B |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|---|--------|-------------------|
| 8 | MO_POS | - Position error monitor output - Refer to section 4.6 for details | Output | E |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|--------------------------|-----|-------------------|
| 9 | GND | - Internal signal ground | NA | NA |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|--------------------------------|-------|-------------------|
| 10 | DIR- | DIR-, B-, CCW- pulse reference | Input | C |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|-------------------------------|-------|-------------------|
| 11 | PUL+ | PUL+, A+, CW+ pulse reference | Input | C |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|--------------------------|-----|-------------------|
| 12 | GND | - Internal signal ground | NA | NA |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|-----------------------------------|-------|-------------------|
| 13 | AIN+ | - Analog reference input POSITIVE | Input | D |

See section 2.3.2 for interface circuit type examples

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|---|-------|-------------------|
| 14 | DIN4 | <ul style="list-style-type: none"> - Optional digital input 4 - By default, input is set to servo drive Alarm Reset | Input | A |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|--|-------|-------------------|
| 15 | ENA | <ul style="list-style-type: none"> - Servo enable/disable input - Active Low (servo disabled when low) - Motor coasts when servo disabled (shaft free) - Disable clears all pulse/analog commands - Disable clears all position calculation and error | Input | A |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|---|-------|-------------------|
| 16 | +24VIN | <ul style="list-style-type: none"> - External +24VDC input - Supply power for digital input circuit A | Input | A |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|---|--------|-------------------|
| 17 | N/C | <ul style="list-style-type: none"> - By default, this pin is not used, do not connect - Optional +14VDC output, max Current Draw: 100mA | Output | NA |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|--|--------|-------------------|
| 18 | ALM | <ul style="list-style-type: none"> - Servo alarm output - Photo transistor ON when servo alarmed, output Low - When servo drive alarmed, output toggled Low/High at 1Hz period interval | Output | B |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|--|--------|-------------------|
| 19 | BKO | <ul style="list-style-type: none"> - Holding brake output - Refer to section 2.6 for wiring details - Refer to section 3.3.2 for timing details | Output | Section 2.6 |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|--|--------|-------------------|
| 20 | MO_ABS | <ul style="list-style-type: none"> - Absolute encoder position monitor output - Refer to section 4.6 for details | Output | E |

See section 2.3.2 for interface circuit type examples

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|--|--------|-------------------|
| 21 | MO_IC | - Servo drive output current monitor - Refer to section 4.6 for details | Output | E |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|--------------------------------|-------|-------------------|
| 22 | DIR+ | DIR+, B+, CCW+ pulse reference | Input | C |

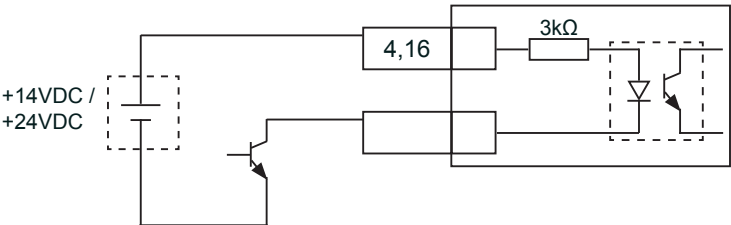
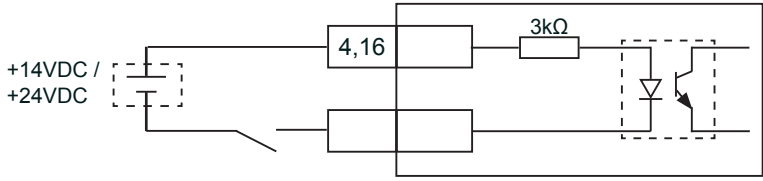
| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|-------------------------------|-------|-------------------|
| 23 | PUL- | PUL-, A-, CW- pulse reference | Input | C |

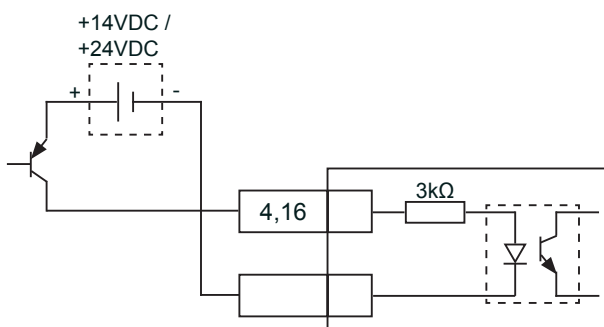
| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|---|--------|-------------------|
| 24 | +5VOUT | - Servo drive internal +5VDC output - Max current draw: 50mA | Output | NA |

| Pin No. | Symbol | Details | I/O | Interface Circuit |
|---------|--------|-----------------------------------|-----|-------------------|
| 25 | AIN- | - Analog reference input NEGATIVE | NA | D |

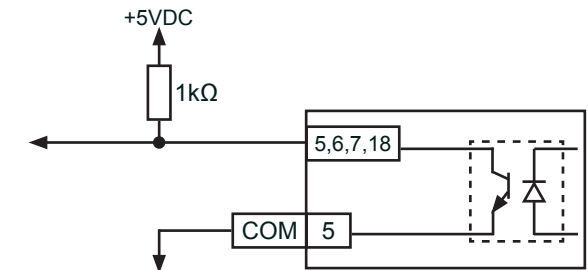
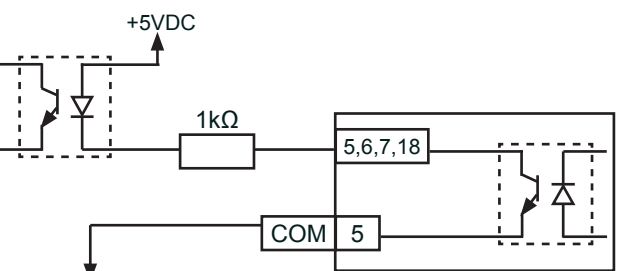
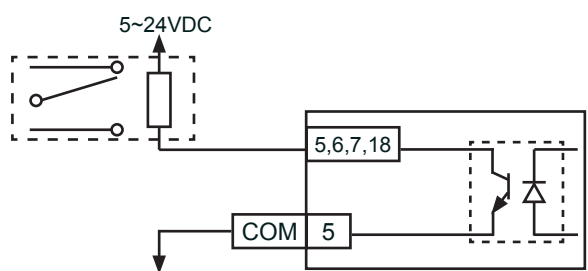
2.3.2 JP4 Interface Circuit Examples

General Input Circuit

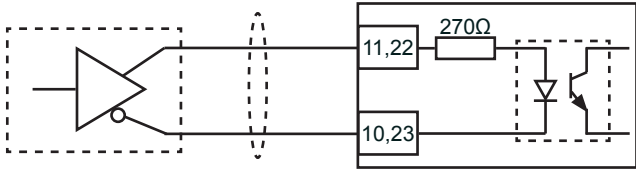
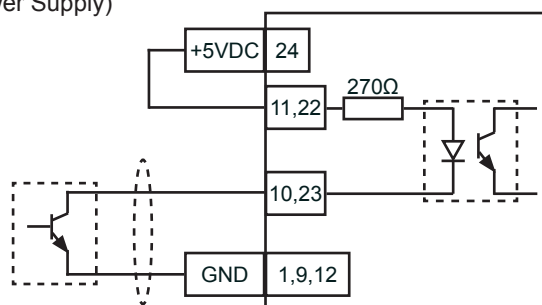
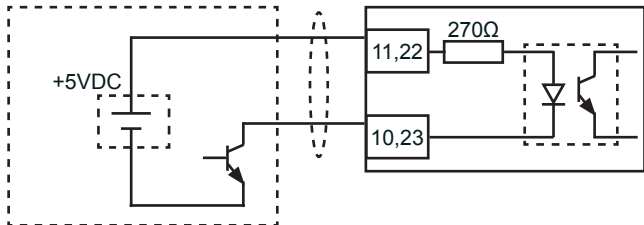
| Interface Type | Applicable Signals | Specification |
|---|--|---|
| A | [JP4-2] ABS_H [JP4-3] SHOLD [JP4-4] +14VIN [JP4-14] DIN4 [JP4-15] ENA [JP4-16] +24VIN | - Use external +14VDC or +24VDC power supply - Can also use internal +14VDC power supply JP4-17 - Max 30VDC input, 10mA - Do not use same +24VDC power supply as holding brake |
| Circuit Example (Sink Circuit) | | Notes |
| Open Collector  | | - NPN Transistor |
| Relay/Switch  | | |

| Circuit Example (Source Circuit) | Notes | | | | | | | | | | | |
|---|------------------|---------------|----------------|---------------|--------|--------|-----|----|-----|----|-----|------|
| <p>Open Collector</p>  | - PNP Transistor | | | | | | | | | | | |
| <p>Example logic using +24VDC power supply:</p> <table><tr><th>Power Supply</th><th>PNP Transistor</th><th>Voltage Level</th><th>Signal</th></tr><tr><td rowspan="2">+24VDC</td><td>OFF</td><td>0V</td><td>Low</td></tr><tr><td>ON</td><td>24V</td><td>High</td></tr></table> | | Power Supply | PNP Transistor | Voltage Level | Signal | +24VDC | OFF | 0V | Low | ON | 24V | High |
| Power Supply | PNP Transistor | Voltage Level | Signal | | | | | | | | | |
| +24VDC | OFF | 0V | Low | | | | | | | | | |
| | ON | 24V | High | | | | | | | | | |

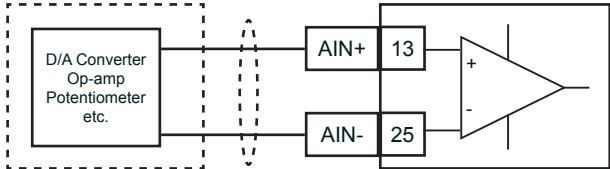
General Output Circuit

| Interface Type | Applicable Signals | Specification |
|--|--|--|
| B | [JP4-5] COM [JP4-6] ONPOS [JP4-7] SRDY [JP4-18] ALM | - Max 30VDC, 50mA - Recommended +24VDC or +5VDC |
| Circuit Example | | Notes |
| <p>Collector</p>  | | |
| <p>Photocoupler</p>  | | |
| <p>Relay</p>  | | |

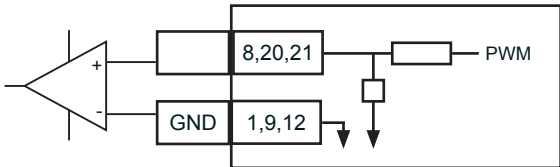
Position Reference Pulse Input Circuit

| Interface Type | Applicable Signals | Specification |
|---|--|---|
| C | [JP4-10] DIR- [JP4-11] PUL+ [JP4-22] DIR+ [JP4-23] PUL- | - Voltage: +5VDC \pm %10 (Contact DMM if higher level such as 12/24VDC is required) - Max pulse frequency: 500kHz - Minimum pulse width: 800us |
| Circuit Example | | Notes |
| Line Driver  | | - Twisted pair cable with shield grounded on receiver side. |
| Open Collector (DYN4 Internal Power Supply)  | | - Twisted pair cable with shield grounded on receiver side. |
| Open Collector (External Power Supply)  | | - Power supply provided by host controller or external source. - Twisted pair cable with shield grounded on receiver side. |

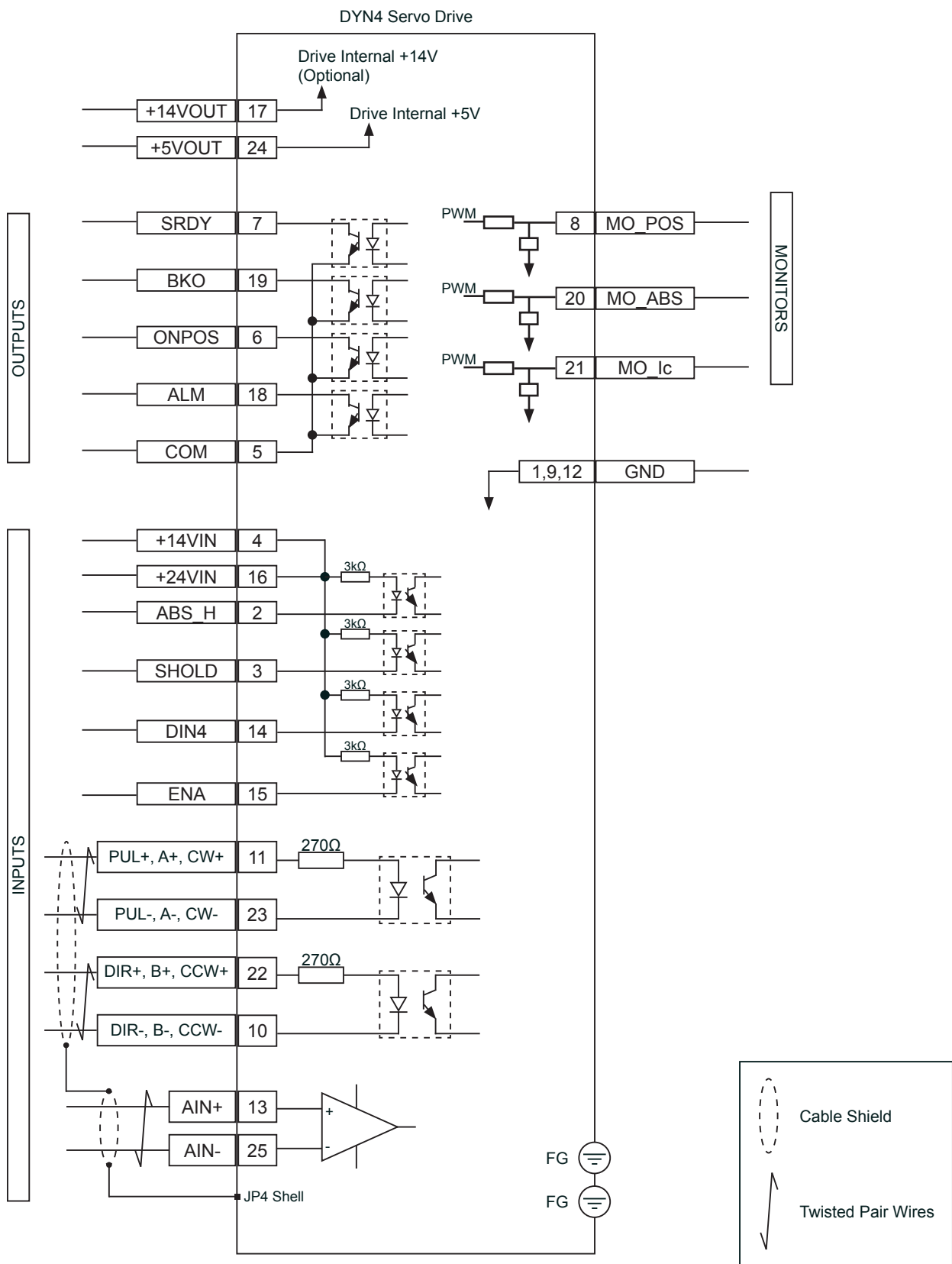
■ Analog Input Circuit

| Interface Type | Applicable Signals | Specification |
|---|--------------------------------|--|
| D | [JP4-13] AIN+ [JP4-25] AIN- | - Max $\pm 12\text{VDC}$ - Max 0.1mA |
| Circuit Example | | Notes |
|  | | |

■ Analog Output Circuit

| Interface Type | Applicable Signals | Specification |
|---|---|---|
| E | [JP4-8] MO_POS [JP4-20] MO_ABS [JP4-21] MO_IC | - $0\sim 3.3\text{VDC}$ output - Max 0.33mA |
| Circuit Example | | Notes |
|  | | |

2.3.3 JP4 Consolidated Interface Circuit



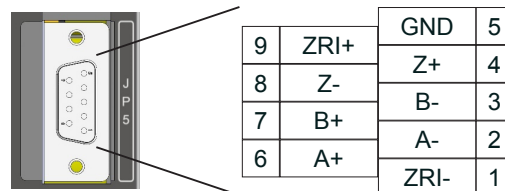
2.4 JP5 Encoder Output Terminal Details

- JP5 Encoder output
Connector Type: D-Sub 9-pin
Recommended Wire Gauge: 0.2mm² (AWG24)

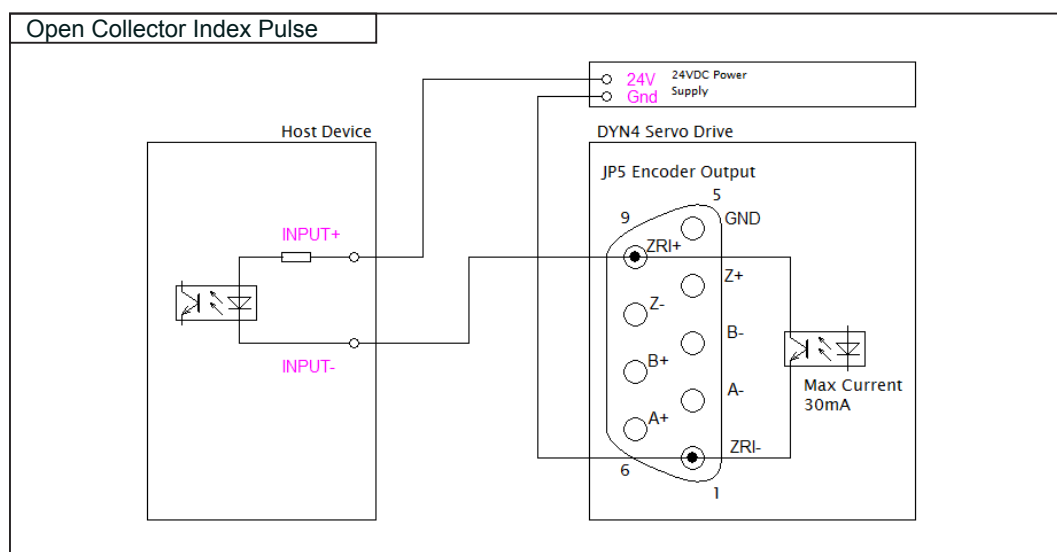
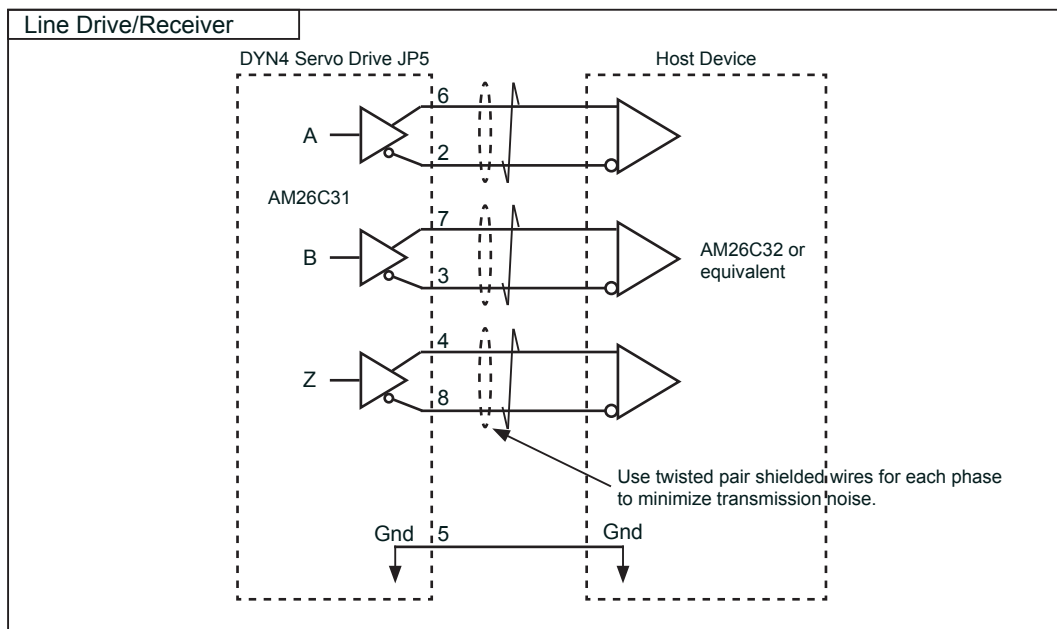
- Specification
*See Section 4.5 for encoder output specification.

- Pin Layout (Viewed from drive header side)

Ground of JP5 connector is connected to D-Sub 9 shell. When using encoder output, make sure ground between host device and DYN4 servo drive is connected together.



- Connection Circuit



2.5 Main Circuit Wiring

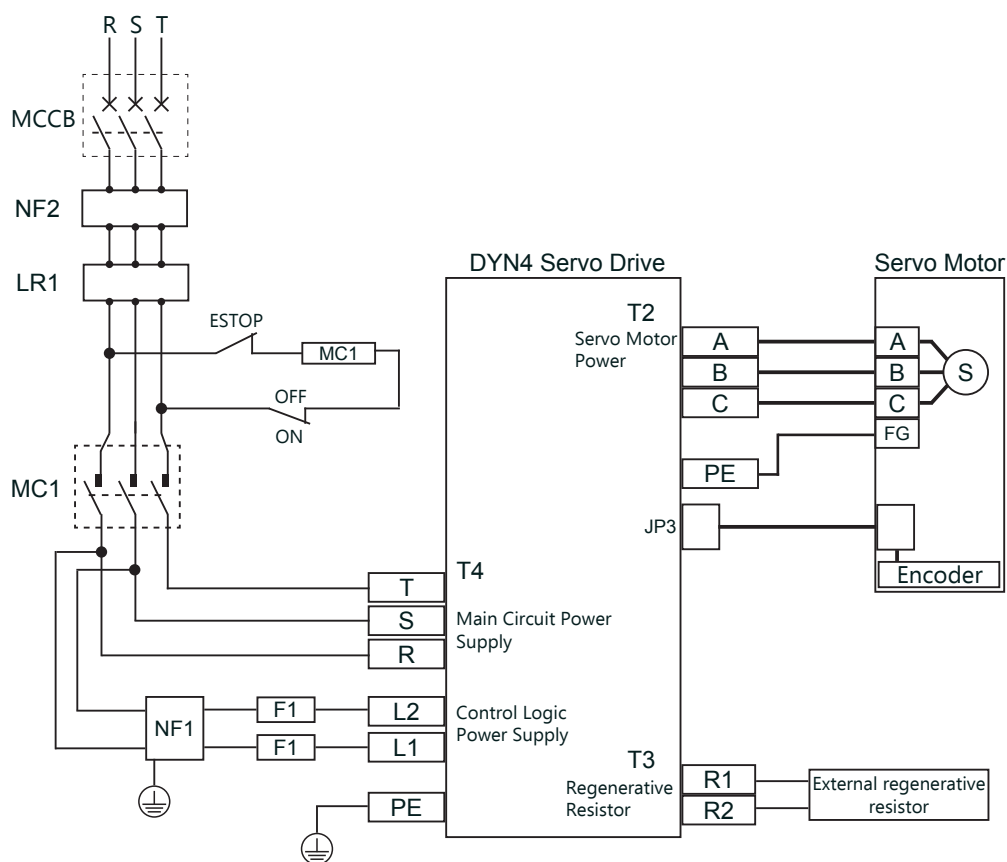
Main Circuit Wiring

Supply Voltage: Single / Three Phase 100VAC ~ 240VAC.

⚠ WARNING

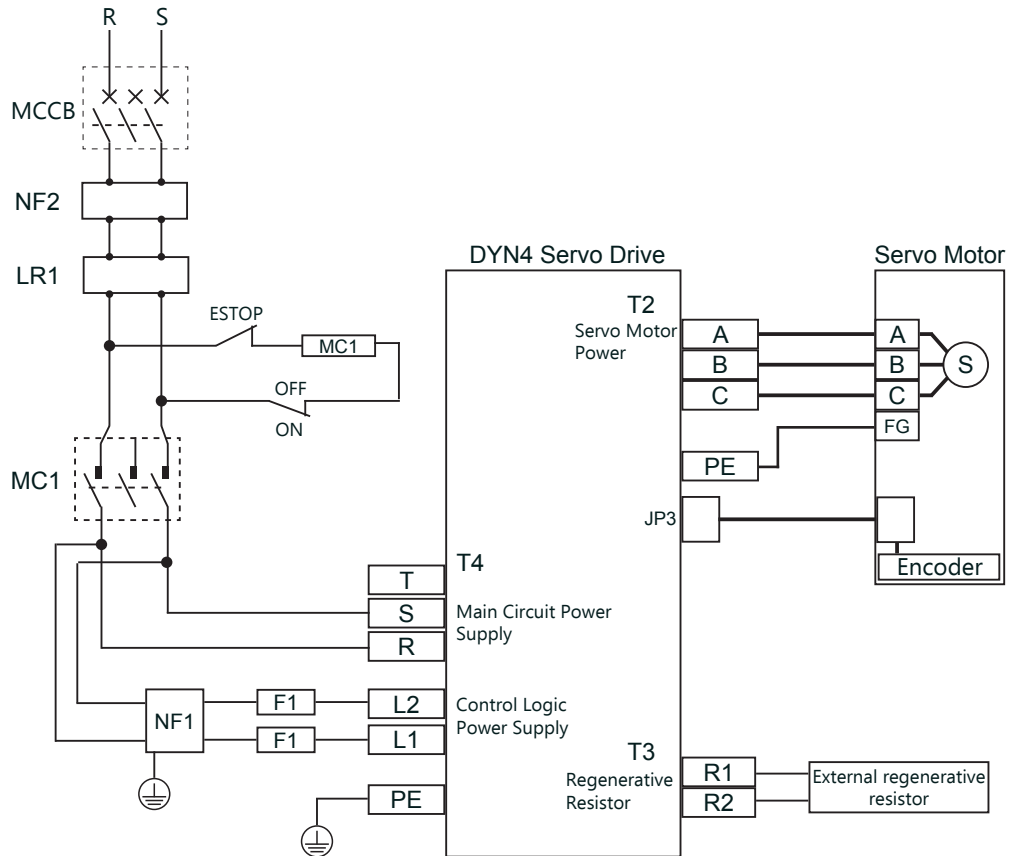
- For Single Phase input, connect the power supply to the R and S terminals of the Main Circuit Power Supply T4.
- Voltage across any two R,S,T,L1,L2 terminals must not exceed 240VAC.
- Refer to Appendix B - DYN4 Servo Drive - AC Power Supply Guidelines (Application Note# AP15-48)

Three Phase Input:



| Component | Part | Recommended Current Rating | | |
|-----------|-----------------------------|----------------------------|--------------------|--------------------|
| | | DYN4-L01 | DYN4-H01 | DYN4-T01 |
| MCCB | Molded case circuit breaker | 10A | 20A | 30A |
| NF2 | Main Noise Filter | 5A FN355-6-05 | 10A FN355-10-05 | 15A FN355-20-03 |
| LR1 | AC Line Reactor (optional) | 5A | 10A | 15A |
| MC1 | Magnetic contactor | 5A | 10A | 15A |
| F1 | Logic Fuse | 1A | | |
| NF1 | Logic Noise Filter | 1A Schaffner FN2030-1-06 | | |

Single Phase Input:

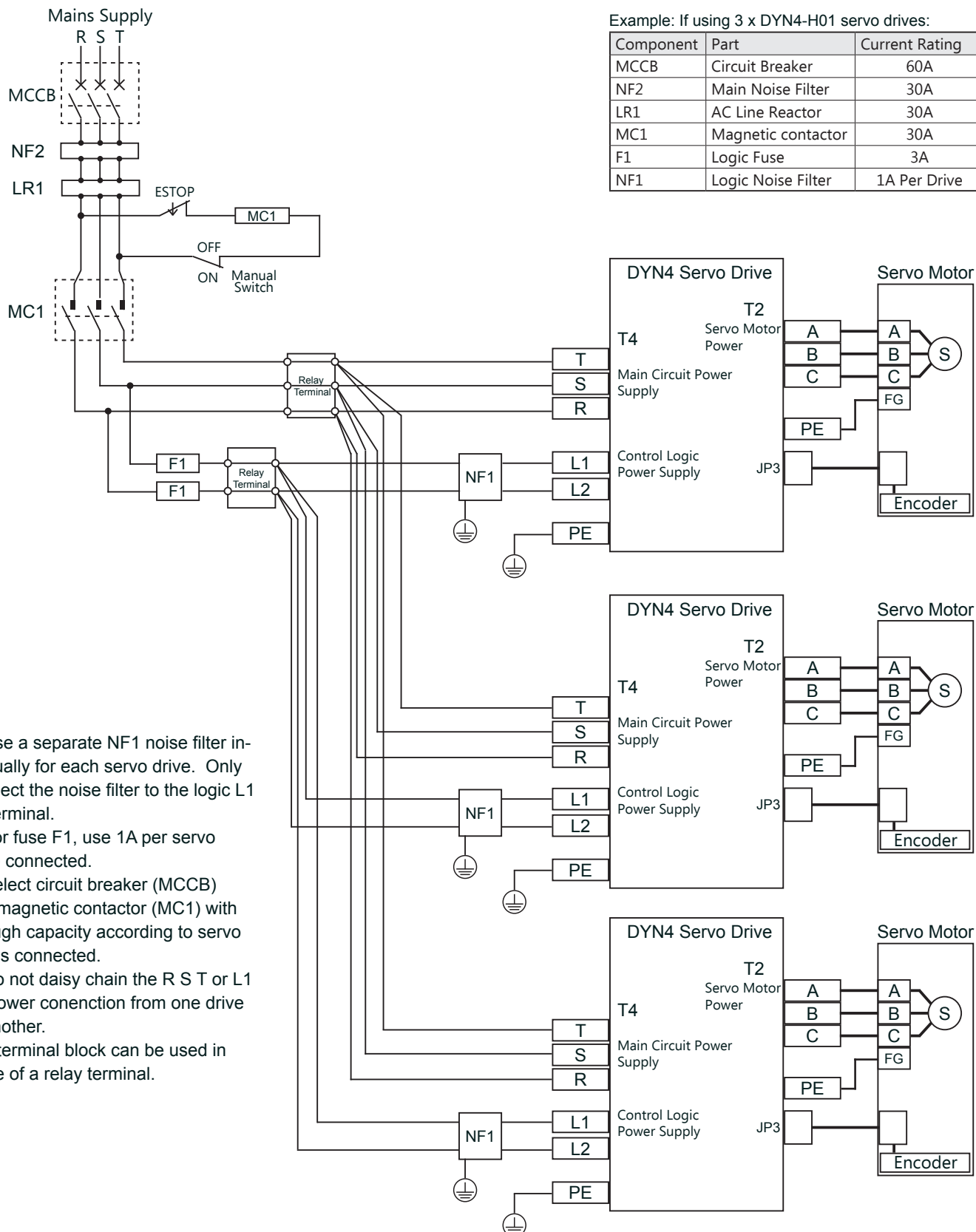


| Component | Part | Recommended Current Rating | | |
|-----------|-----------------------------|----------------------------|---------------------|---------------------|
| | | DYN4-L01 | DYN4-H01 | DYN4-T01 |
| MCCB | Molded case circuit breaker | 10A | 20A | 30A |
| NF2 | Main Noise Filter | 5A FN2030-6-06 | 10A FN2030-10-06 | 15A FN2030-16-06 |
| LR1 | AC Line Reactor (optional) | 5A | 10A | 15A |
| MC1 | Magnetic contactor | 5A | 10A | 15A |
| F1 | Logic Fuse | 1A | | |
| NF1 | Logic Noise Filter | 1A Schaffner FN2030-1-06 | | |

Multiple Drives

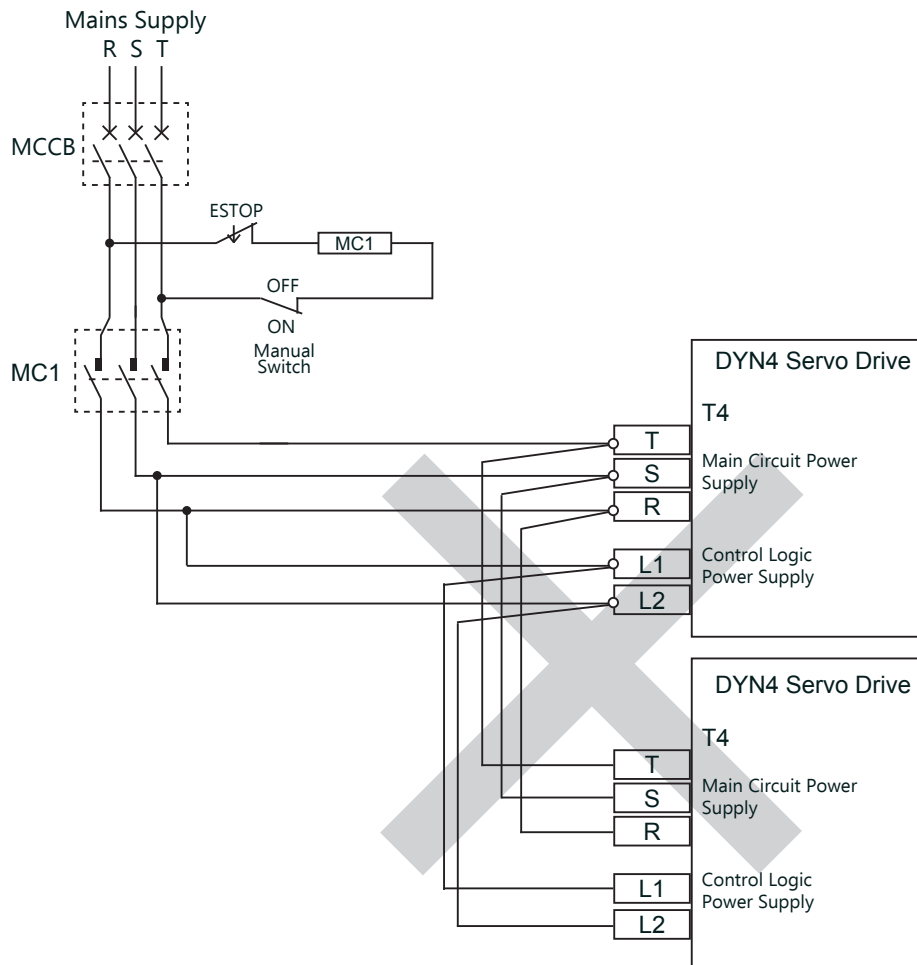
Multiple drives can share the same mains power supply. Select MCCB, MC1 and F1 size according to number of drives. Example. if 2 servo drives are connected, select double the rated MCCB and fuse. Use a single point star connection from a single relay or strip terminal to distribute power to each servo drive.

Connect noise filter NF1 individually for each drive.



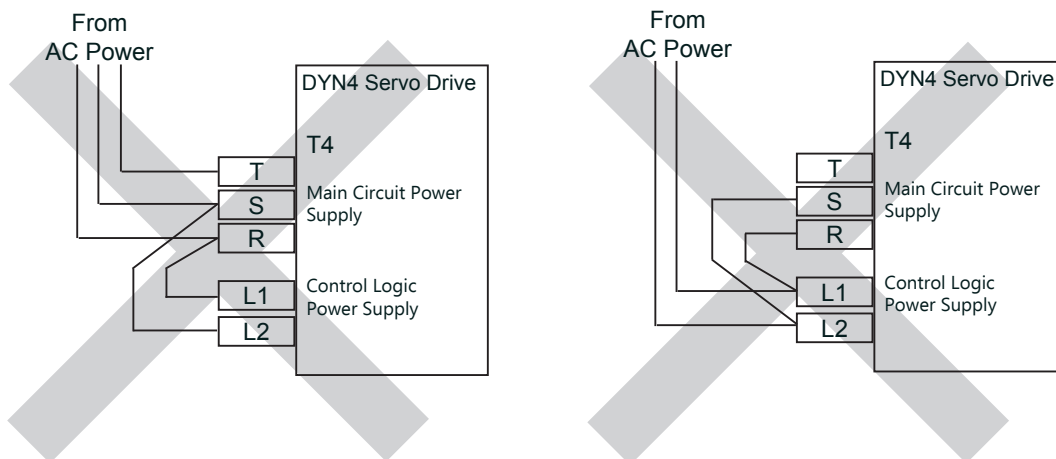
! WARNING

- Do not daisy chain the main power input from one drive to another. The different power levels of each servo drive can cause permanent damage at power ON or during operation.



! WARNING

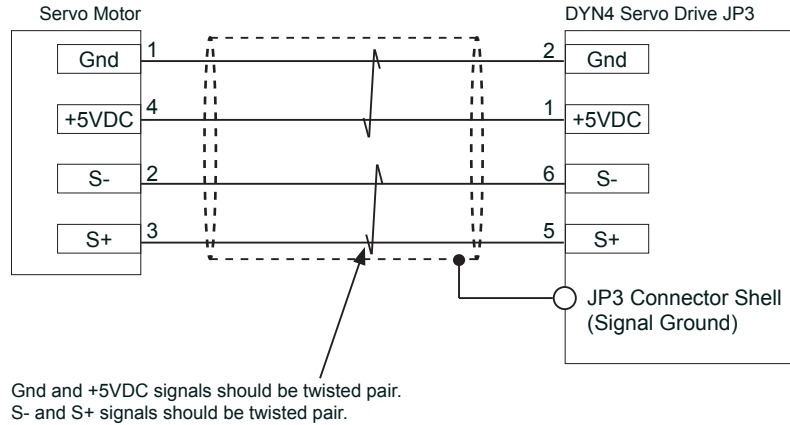
- Do not jumper the control R,S,T and logic L1, L1 terminals. Connect each to the power supply source directly.



⚠ WARNING

- Do not connect encoder cable shielding to earth ground. Connect the encoder cable shielding to the signal ground on the servo drive end.

Encoder Cable



Regenerative Resistor

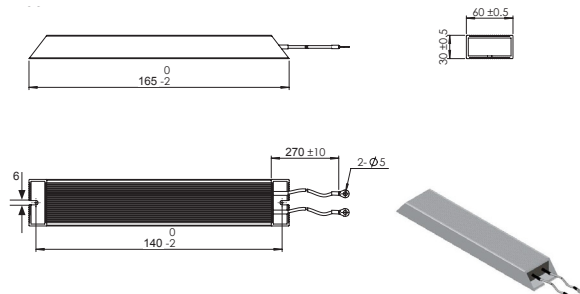
The DIPM power module unit built into the DYN4 servo drive is designed with a very efficient bus voltage attenuation control circuit. It is much more efficient than conventional servo drives when handling servo motor regenerative energy. Generally, external regenerative resistors are not needed. If the servo drive often throws “Over Voltage” alarm, contact DMM representative for regenerative energy sizing procedures.

Resistors available from DMM:

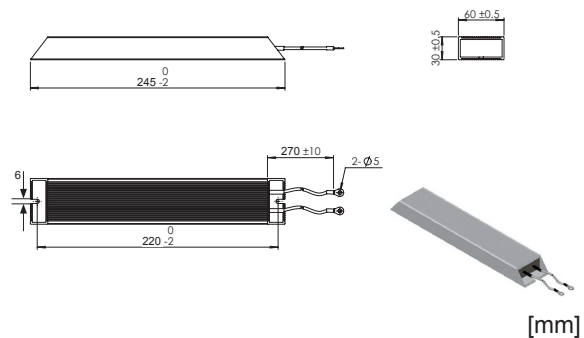
| Motor Size | Resistor Model # | Resistance | Rated Power | Specification |
|-------------|------------------|------------|-------------|--|
| 50W~750W | SRGA-200W100R | 100Ω ±5% | 200W | Insulation Resistance: DC1000V, 100MΩ Withstand Voltage: AC1800V, 1min, Leakage Current≤5mA |
| 1.0kW~1.8kW | SRGA-500W100R | 100Ω ±5% | 500W | |

Dimensions:

• SRGA-200W100R

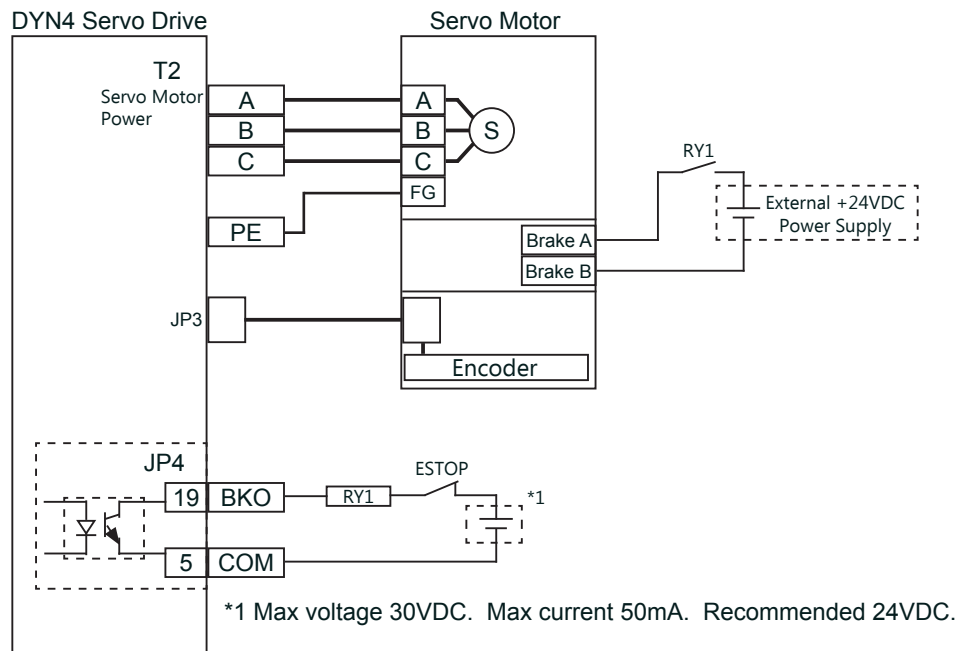


• SRGA-500W100R



An external relay circuit should be used to control servo motor holding brake. The relay logic should be triggered by the BKO signal [JP4-19]. An e-stop switch should also be able to engage the holding brake in emergency situations. Refer to Section 3.3.2 for the holding brake control timing. BKO logic determined by servo Enable logic. BKO phototransistor ON whenever servo drive is enabled.

The DYN4 servo drive is enable by default upon power ON. If the control logic circuit (L1, L2) is powered ON before the main circuit (R, S, T) the brake motor might fall due to BKO phototransistor ON and motor not energized. In this case, connect an additional relay to control BKO circuit logic.



- Do not use the holding brake to decelerate or stop the servo motor under normal operation.
- Check the servo motor brake connector polarity before operating the brake.
- The holding brake draws higher current than standard I/O signals, use independent DC power supplies for the holding brake and the servo drive I/O control interface power.
- Holding brake inertia will affect servo motor performance. Servo motors with holding brake option will have lower load inertia ratio capacity and angular acceleration.
- Holding brake is servo motor frame size-specific. Contact DMM representative for full specifications.

3 Start - Up

3.1 Preparation and Mounting

■ Unpacking

Remove the servo drive from the box and lining and visually inspect there is no damage to product. Check that the plug connectors for terminals T2, T3 and T4 are included. The servo drive packaging does not include a physical copy of the instruction manual. Take note of the hardware and firmware version labeled and ensure it is the same version as listed on this instruction manual cover.

■ Mounting

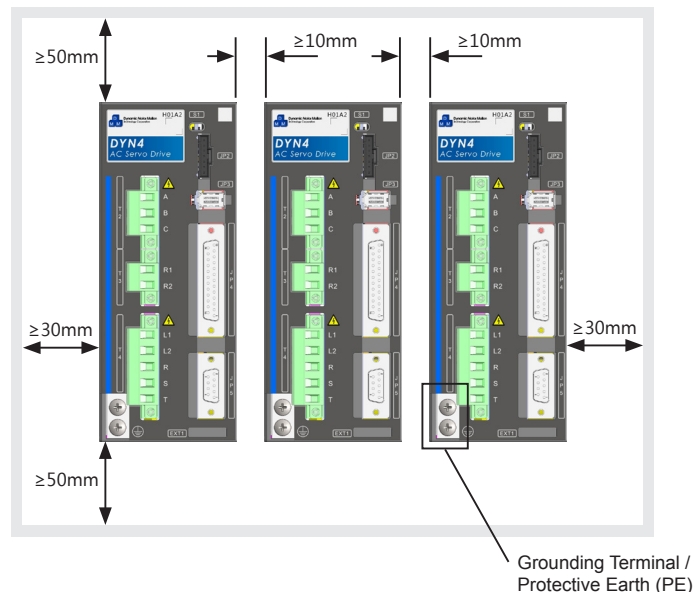
The servo drive can be mounted vertically or horizontally. Ensure that the circulation air flow is parallel to the heat sink blades.

Use 2 x M4 screws to mount the servo drive onto a grounded and electrically conductive surface. See Appendix A for mounting dimensions.

■ Cables and Wire Management

Keep all cables wires as short as possible and avoid loops. Keep high voltage lines including main power input and servo motor power output lines far away from low voltage lines including I / O cables, PC interface cable and encoder feedback cable. Securely route all cables to avoid damage during operation. Ensure none of the cables are pulling on the connectors or terminals. Maintain highest contact surface area.

3.2 Multiple Drive Spacing



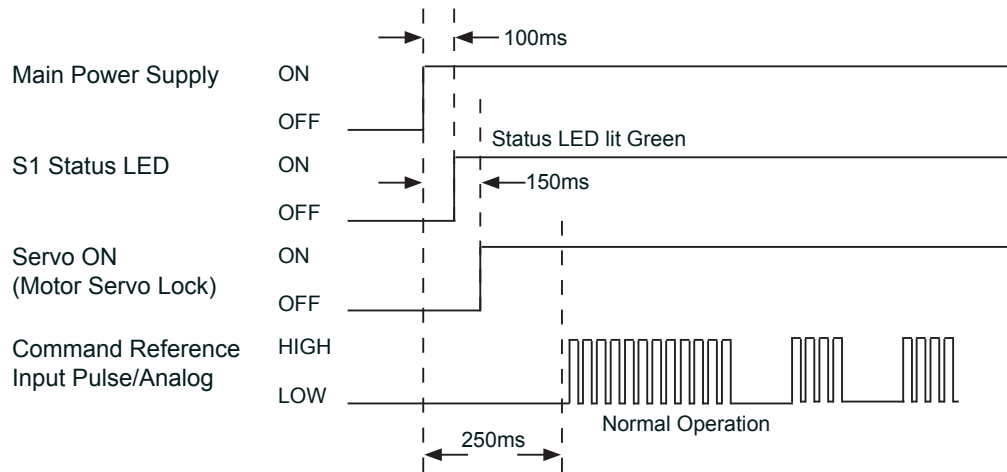
3.3 Grounding

Use a star connection to connect each servo drive grounding terminals to earth at one single point. Do not use daisy chain connection between each servo drive as it may cause noise and interference. Connect power source earth to one terminal and motor frame ground to other terminal.

3.4 Power Up Timing Chart

3.4.1 Power ON Timing

After servo drive power ON, wait at least 150ms before sending pulse, analog or serial command to servo drive.



◆ Supply Power Cycle

Do not cycle the main power supply quickly as internal circuit may be permanently damaged. Power to servo drive should be turned on once during each operation cycle.

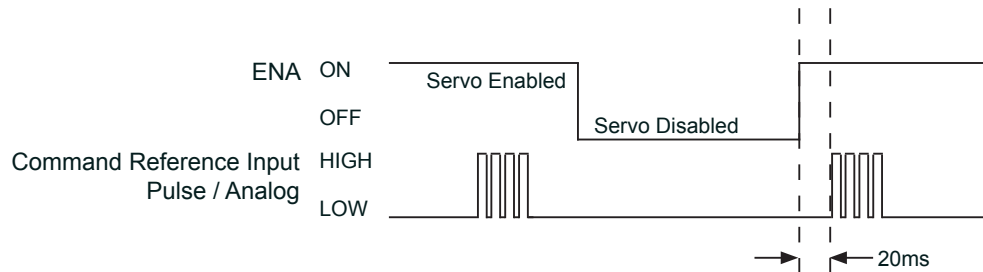
◆ Power Off Residual Voltage

After drive power off, the user should wait at least 60 seconds before servicing or touching the servo drive frame. Residual voltage may remain in the circuit after immediate power off. 60 seconds is needed for full discharge.

The residual voltage may cause the servo motor to rotate for a short period even after immediate power off. Consider this effect for emergency situations and take safety precaution to prevent damage to personnel, equipment or machine.

■ Servo Disable / Enable Timing

When using the ENA signal to disable the servo drive to coast servo motor, do not cycle this input rapidly ON/OFF. If the signal is cycled too fast, the servo drive will not have enough time to initialize the control program during Enable and can cause unwanted or dangerous results. Ensure that in the control program, the below timing is satisfied. Once Disabled, do not Enable the servo drive during motor coast or any time motor shaft is rotating, make sure motor rotation has completely stopped before Enable again.



By factory default, the Enable logic is Active High. Servo drive is Enabled when powered up, without any input.

Enable logic can be toggled in the DMMDRV program to Active Low if desired:

DYN4 AC Servo Drive

COM Port: 3

Main Gain: 18
Speed Gain: 4
Integration Gain: 24
Torque Filter Constant: 127
Max Acceleration: 8
Max Speed: 20
Driver ID [0:127]: 0
OnPosRange [1:127]: 127
Gear Number [500:16,384]: 4096
Line Number [500:4,095]: 2048
Gear_Ratio = 4096/Gear_Num 1~2048

☐ Start from ABS Zero
☐ RS485/232 Network
☒ Servo Enable Active Low

Command Input Mode:
☒ RS232
☐ PULSE / DIR
☐ A / B Phase
☐ CW / CCW
☐ Analog
☐ CAN
☐ Modbus

Servo Mode:
☒ Position Servo
☐ Speed Servo
☐ Torque Servo

Modbus Setting:
Baud Rate: 38400
Format: 0: 8-bit data, 1

CAN Setting:
Baud Rate [kHz]: 10

Last Readout Parameters:

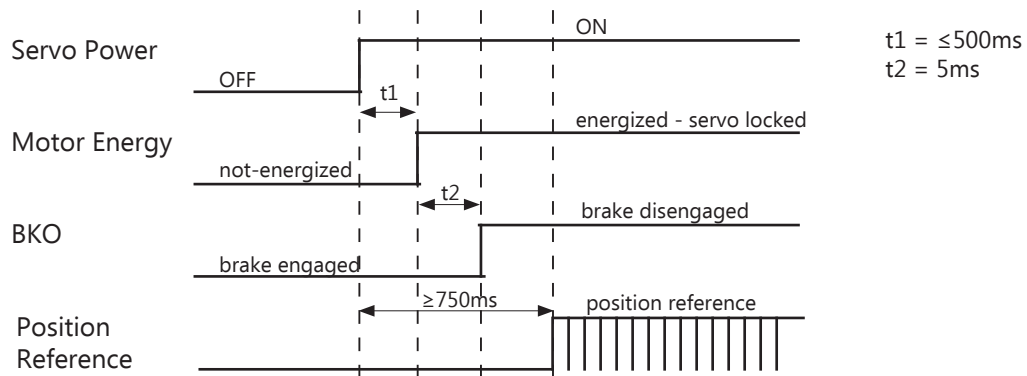
| Parameter | Drive Setting |
|------------------------|---------------|
| Main Gain | 0 |
| Speed Gain | 0 |
| Integration Gain | 0 |
| Torque Filter Constant | 0 |
| Max Speed | 0 |
| Max Acceleration | 0 |
| On Position Range | 0 |
| GEAR_NUM | 0 |
| LINE_NUM | 0 |
| Driver Status | 0 |
| Driver ID | 0 |

Test Motions:
☐ Trimming
☐ Sinusoidal
☐ Step
☐ Constant Speed
Position Offset: 1
Frequency [Hz]: 1
Frequency [Hz]: 1
Speed (RPM): 0
Trims from Abs Zero
Amplitude [Pulse]: 1
Amplitude [Pulse]: 1
☐ Reverse Turn

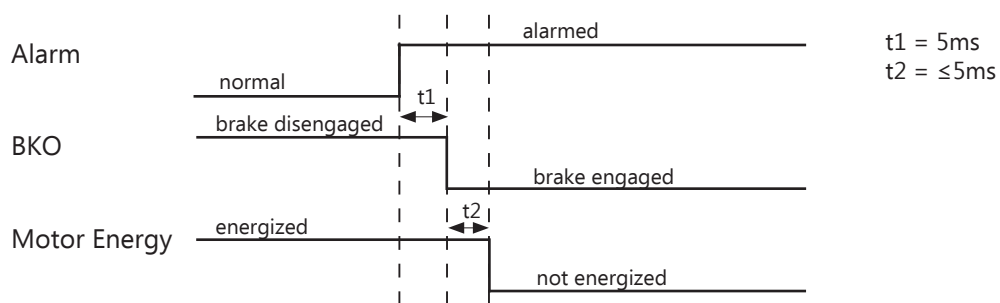
Buttons: Read, Save All, Refresh Cons

3.4.2 Holding Brake Timing

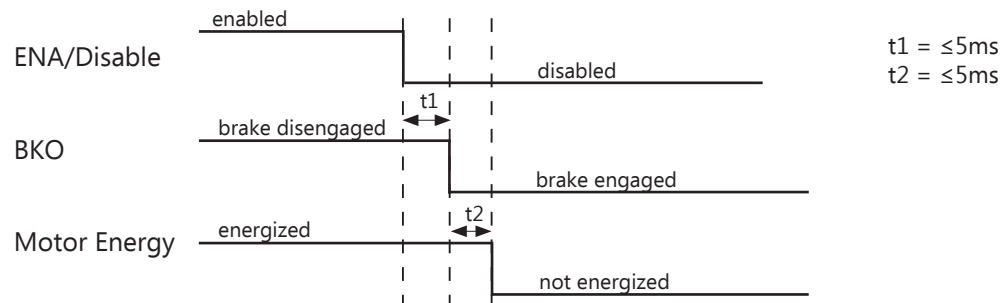
Timing 1 - Servo Main Power ON



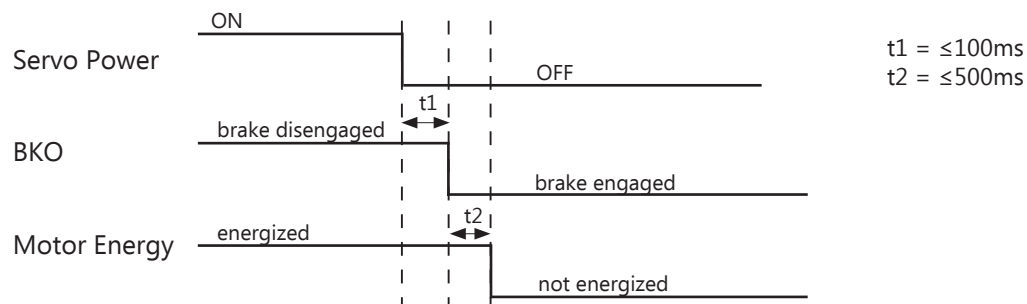
Timing 2 - Alarm



Timing 3 - Servo OFF / Disable



Timing 4 - Servo Main Power OFF



3.5.1 DMMDRV Software Communication



◆ PC Requirements

Operating System: Windows XP SP3 or higher

*Recommended: Windows 7 (32-bit / 64-bit)

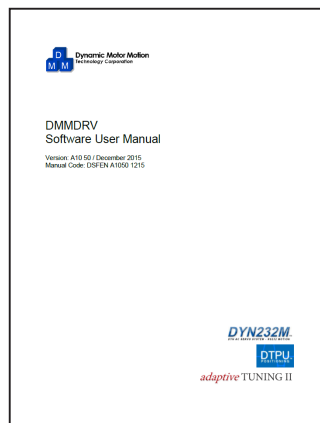
Processor: Pentium 1 GHz or higher

RAM: 512 MB or more

Framework: .NET Framework 4 or higher

Minimum disk space: 60MB

*See User Manual DSFEN for complete set up instructions:





◆ PC Running Requirements

Win98/XP/2000/Vista/7
250Mhz CPU
64MB RAM
250MB Hard Disk Space

The servo drive should be powered up with the servo motor encoder feedback and motor power cables connected. The servo motor shaft will be servo-locked when powered ON. Connect the RS232 tuning cable from port JP2 to host PC.

◆ DMMDRV Start Up

- 1) Open the DMMDRV4.exe executable
- 2) Select "COMSET" --> "COM PORT"
- 3) Change the port number to the servo drive connected RS232 port, then select "OK"
- 4) Select "SERVO SETTING"
- 5) Select DYN4 Servo Drive
- 6) Press Read on the Setting driver parameters and mode dialogue box. After approximately 1~2 seconds, the on-screen parameters will change according to the current internal parameter settings of the connected servo drive. Ensure that the Driver Status indicates ServoOnPos to indicate that the drive has closed the position loop with the motor and is fully operational.



WARNING

- During Test movement procedures, the servo motor can rotate very quickly in either direction. Ensure that the servo motor is free to rotate and no objects are attached to or is near the motor shaft. Secure the motor by its flange.

◆ Test Movements

- 1) Select "RS232" under the command input mode option
- 2) Select "Position Servo" under Servo mode, then click "Save All" to save this setting.
- 3) Under the Test Motions menu, the user can select one of 4 test motions to JOG, STEP, SINE or TRIM the servo motor. Only one test motion profile can be run at a time, use the radio buttons below each section to select the movement profile.

◆ Operation

- 1) Select the command input mode
- 2) Select the Servo mode
- 3) Click Save All to save the parameters to the servo drive
- 4) Refer to the next section (Section 4) for individual operation mode specifications

4 Operation

4.1 Position Servo Mode

■ Pulse Specifications

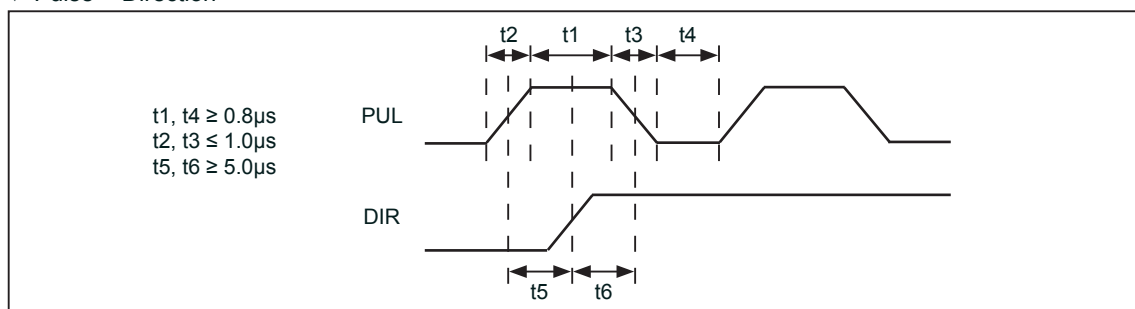
Voltage: +5VDC \pm %10 (Contact DMM if higher level such as 12/24VDC is required)

Max pulse frequency: 500kHz

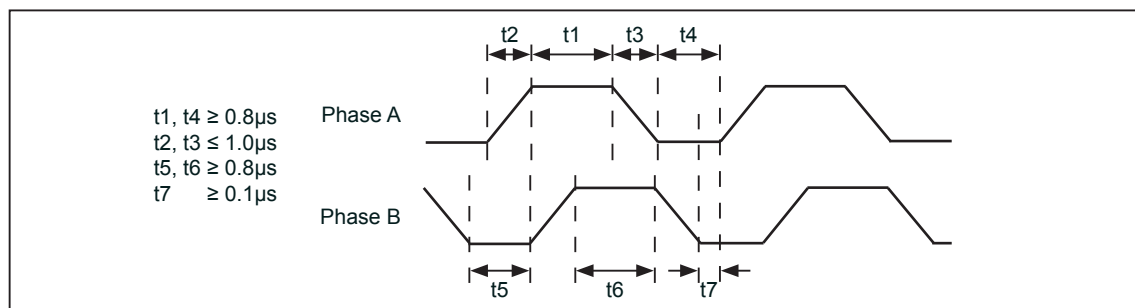
Minimum pulse width: 0.8 μ s

This control mode is activated by putting the servo drive into Position Servo Mode and Pulse/Dir command input mode

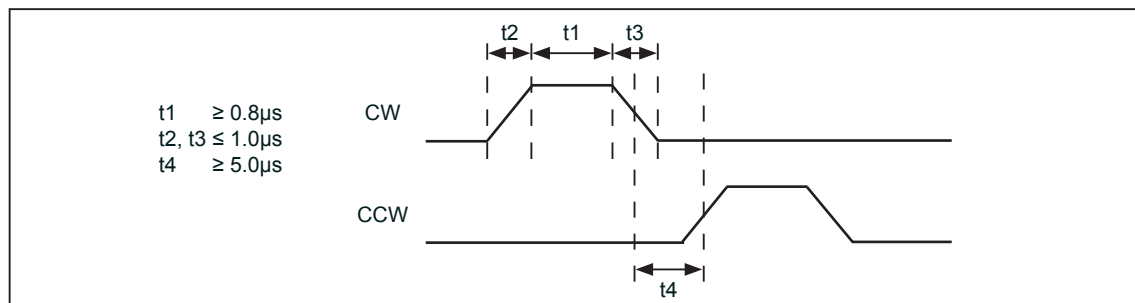
◆ Pulse + Direction



◆ A/B phase quadrature with 90° phase differential



◆ CW + CCW




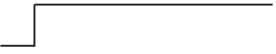




Reference Pulse Format





The DYN4 servo drive accepts FORWARD reference as CLOCKWISE motor shaft rotation as viewed from motor shaft side.

The GEAR_NUM parameter in the servo drive controls the input pulse resolution. GEAR_NUMx4=number of pulses needed from the host controller to move the motor 1 revolution. ex. If GEAR_NUM is set to 8,000, then 32,000pulses needed to move motor 1 revolution.





◆ Pulse + Direction

| Forward Reference | Reverse Reference |
|---|---|
| <p>PUL+ JP4-11 </p> <p>DIR+ JP4-22 </p> | <p>PUL+ JP4-11 </p> <p>DIR+ JP4-22 </p> |

◆ A/B phase quadrature with 90° phase differential

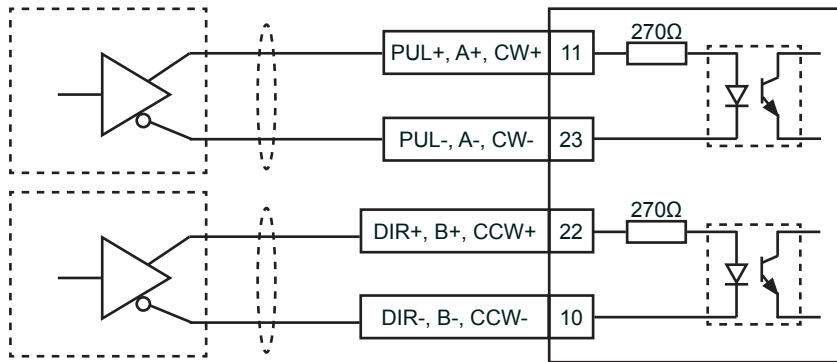
| Forward Reference | Reverse Reference |
|---|---|
| <p>A+ JP4-11 </p> <p>B+ JP4-22 </p> <p>A Leads B</p> | <p>A+ JP4-11 </p> <p>B+ JP4-22 </p> <p>B Leads A</p> |

◆ CW + CCW

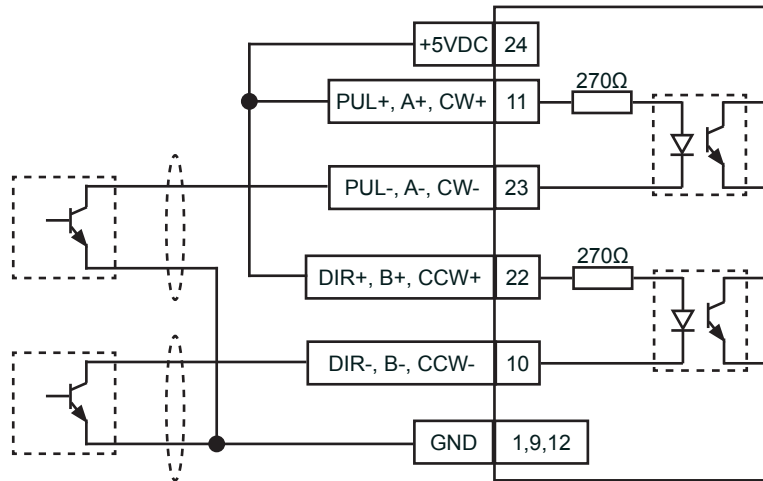
| Forward Reference | Reverse Reference |
|--|--|
| <p>CW+ JP4-11 </p> <p>CCW+ JP4-22 </p> | <p>CW+ JP4-11 </p> <p>CCW+ JP4-22 </p> |

Connection Example

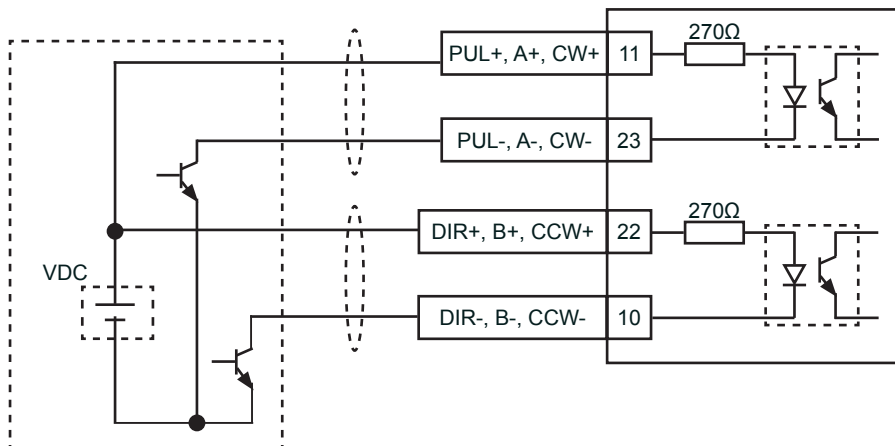
◆ Line Drive Output



◆ Open Collector Output - Internal Power Supply



◆ Open Collector Output - External Power Supply



Electronic Gearing (*GEAR_NUM* Parameter)

Gear number (*Gear_Num*) is set from 500 to 16,384, default value is 4,096. Gear number provides an electrical gear ratio: $4096 / \text{Gear_Num}$, from 0.25 ~ 8.192. For example, if Gear number = 4,096, then 16,384 input counts from pulse will turn motor exactly one revolution. If Gear number = 500, 2,000 pulses will turn motor one revolution. Gear number parameter is only applicable to position servo mode.

Analog Input in Position Servo Mode

In position servo mode, the controller can use 0 ~ 10VDC analog input to turn the motor. 0 ~ 10VDC analog input commands motor from 0 ~ $90 \times 4,096 / \text{Gear number}$ (degrees). Ex. if *Gear_Num*=8,192, a 5.5V input will move the motor 24.75degrees.

Servo In Position Output Specifications (*ONPOS*)

On position range is a value used for determining whether the motor has reached the commanded position or not. This on position range is selectable according to customer's requirement. Suppose the *Pset* is the commanded position, and *Pmotor* is the real motor position, if

$$|\text{Pset} - \text{Pmotor}| \leq \text{OnRange}$$

it is said motor is ON the commanded position, otherwise not. That *OnRange* is set from 1~127. The real position on range is: $\text{OnRange} \times 360(\text{deg}) / 16,384$. The *ONPOS* output (JP3-9) phototransistor ON if motor in position and OFF if motor off position.

On position range defined as:

$$\text{OnPosRange} \times 4 / 65,536 \times 360(\text{deg}) = \text{"On Position"} (\text{deg})$$

Ex. If *OnPosRange* is set to 24, then:

$$24 \times 4 / 65536 \times 360 = 0.53 \text{deg}$$

JP4. Pin 6 Phototransistor becomes ON when motor within 0.53degree of command position.

Servo Position Error Accumulation

The servo drives internal position error decides the status of the On Position signal and the Lost Phase servo drive alarm.

The On Position signal is triggered (LOW) when the servo position error is within the *OnPosRange* set in the *DMMDRV* program. The Lost Phase alarm is triggered when the servo motor is physically 90° or more out of position for ~2 seconds.

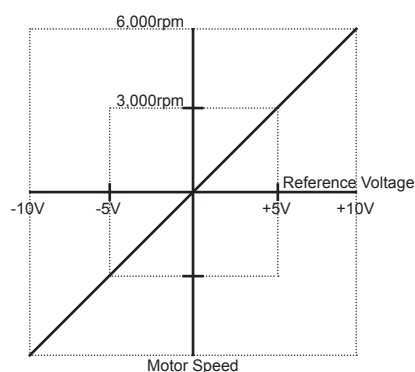
The servo position error is cleared when the drive is disabled using the *ENA* input and does not accumulate when the drive is disabled.

In speed servo mode, the DYN4 servo drive takes command from an external $\pm 10\text{VDC}$ analog reference voltage from the host controller to drive a linear proportional motor speed.

In speed servo mode, the torque output depends on the load on the servo motor and determined by the motor feedback position. Design the system so it can withstand the peak torque of the motor in use.

Control Reference

The DYN4 servo drive accepts FORWARD reference as CLOCKWISE motor shaft rotation as viewed from motor shaft side. Positive reference voltage rotates the servo motor in the FORWARD (CLOCKWISE) direction and negative reference voltage rotates the servo motor in the REVERSE (COUNTER CLOCKWISE) direction.



| Reference Voltage | Motor Speed | Reference Direction | Motor Direction |
|-------------------|-------------|---------------------|-----------------|
| +10V | 6,000rpm | FWD | CW |
| +5V | 3,000rpm | FWD | CW |
| -3V | 1,800rpm | REV | CCW |

Analog Input Specification

See APPENDIX C for analog input scaling and deadzone settings.

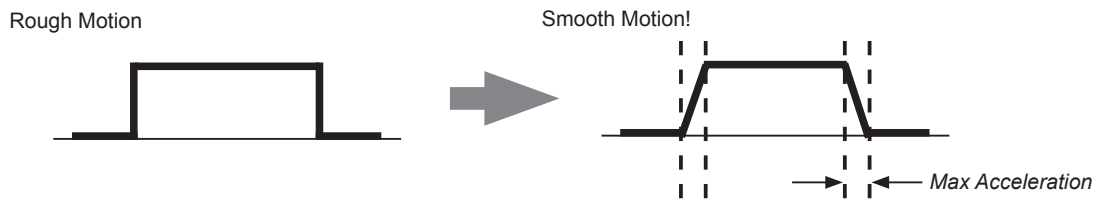
Acceleration / Deceleration Soft Start

In Speed Servo Mode, the *Max Acceleration* parameter in the servo drive can be used to soft start/stop the servo motor. If the analog speed command is sent as a step reference, it is often desirable to ramp up/down this command to smooth motor motion. Without soft start, the servo motor can accelerate/decelerate instantaneously. Soft start creates a smooth s-curve motion with consistent acceleration/deceleration.

The relation to physical acceleration / deceleration time is measured as the rise time from 10% of the target speed to 90% of the target speed.

Rise from 10% to 90% time = $59.98 / (\text{Max Acceleration})^2$ seconds

Physical acceleration time = $1.2 * 59.98 / (\text{Max Acceleration})^2$ seconds



Torque Filter Constant

TrqCons is a first order low-pass filter used to smooth torque delivery to improve stability and accuracy of servo motor speed. The bigger value means wider frequency range of that filter. That filter can be expressed as:

$a / (S + a)$, here $a = 26 * \text{TrqCons}$; if $\text{TrqCons} = 100$, then $a = 2600$.

The filter is used to make the torque sent to the servo torque loop more smooth especially for heavier loads when bigger SpeedGain settings are used. If a very quick response servo with small load is desirable, TrqCons can be increased up to 127 to ensure stability and fast dynamic following.

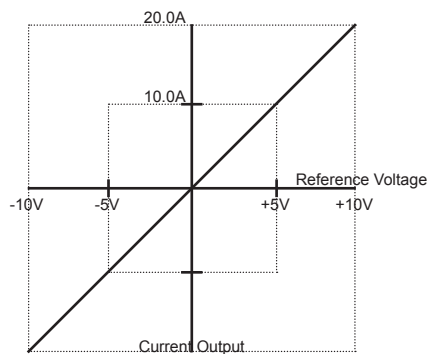
The Torque Filter Constant parameter should only be used in speed and torque servo mode. Leave this parameter at "127" in position servo mode.

4.3 Torque Servo Mode

In torque servo mode, the DYN4 servo drive takes command from an external $\pm 10\text{VDC}$ analog reference voltage from the host controller to drive a linear proportional output current.

■ Control Reference - [H01] Capacity Model: DYN4 - H01

The DYN4 servo drive accepts FORWARD reference as CLOCKWISE motor shaft rotation as viewed from motor shaft side. Positive reference voltage rotates the servo motor in the FORWARD (CLOCKWISE) direction and negative reference voltage rotates the servo motor in the REVERSE (COUNTER CLOCKWISE) direction.



| Reference Voltage | Output Current | Reference Direction | Motor Direction |
|-------------------|----------------|---------------------|-----------------|
| +10V | 20.0A | FWD | CW |
| +5V | 10.0A | FWD | CW |
| -3V | 6.0A | REV | CCW |

■ Analog Input Specification

See APPENDIX C for analog input scaling and deadzone settings.

4.4 RS232 Command Input Mode

The RS232 port is always active after power on for DYN-series servo drive. This active RS232 port could be used for reading and setting Drive parameters and status, and also could be used for sending point to point position command.

If the position command input mode is selected as Pulse mode or Analog mode, the RS232 port is still active as mentioned above but it only can be used for reading and setting Drive parameters. The RS232 port could be easily accessed by using the DMMDRV program after the connection between PC and the Drives RS232 port. This is the easiest way to tune up the servo and make test movements. The RS232 port could be accessed by other micro-controller, or DSP if sending and reading data by using DYN Drives RS232 protocol.

The PC or RS232 controller is working as Master and the servo drive is always working as Slave. Several servo drives could be RS485 networked for integrated multi-axis control.

See Section 7 for DYN4 servo drive RS232 protocol definitions.

4.5 Encoder Output

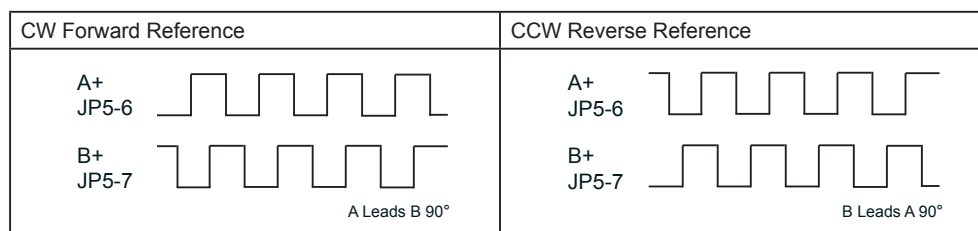
Refer to Section 2.4 for encoder output connection diagram and circuit. This real time emulated encoder output is scalable using the LINE_NUM parameter set in the DMMDRV program. The pulse output per revolution is set according to:

$$\text{Pulse output per revolution} = \text{LINE_NUM} * 4$$

For example, if LINE_NUM = 2,000 then 8,000 pulses will be output per motor revolution. The Z pulse is output once per motor revolution. LINE_NUM can be set from 500 to 2,048.

Encoder Pulse Specifications

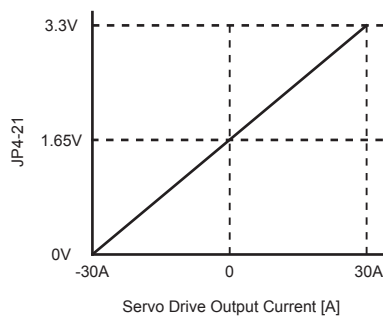
- ◆ Pulse Logic: +5VDC
- ◆ Z Pulse: Width $\geq 0.7\mu\text{s}$. No phase relation to A/B pulse.
- ◆ Forward Reference = CW as viewed from motor shaft side



Current Monitor (JP4-21)

The current monitor outputs an analog signal relative to the real-time current output from the servo drive to the servo motor. The current can be monitored relative to both positive (CCW from motor shaft) and negative (CW from motor shaft) directions.

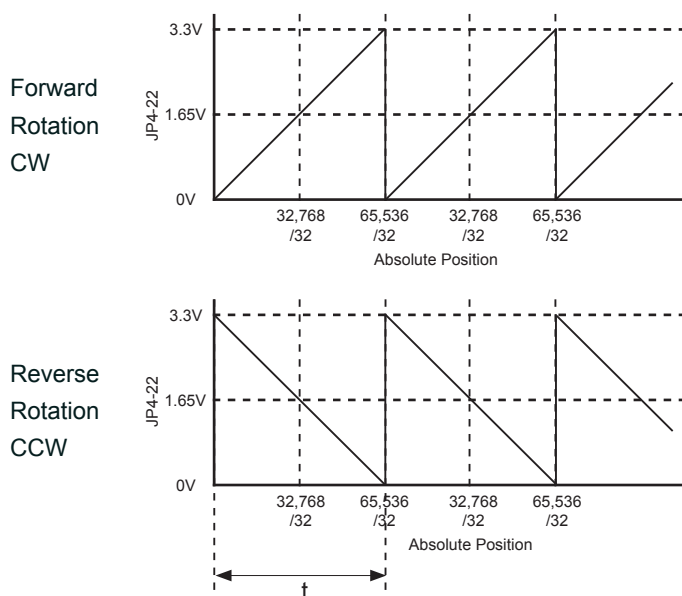
The analog output has a domain of [0V : 3.3V] relative to [-30A : 30A] current output.



| Servo Drive Current Output [Ic] | JP4-21 Analog Output |
|-----------------------------------|----------------------|
| 30A | 3.3V |
| 0A | 1.65V |
| -30A | 0V |

Absolute Position Monitor (JP4-22)

The absolute position monitor outputs an analog signal relative to the real-time single turn absolute position of the servo motor. The signal can be used to track servo motor position or can be used to calculate servo motor speed according to the signal period. Can also be used to detect servo motor zero speed. The analog output has a domain of [0V : 3.3V]. Forward direction rotates the servo motor CW as viewed from the shaft. Each servo motor rotation will output 32 cycles.



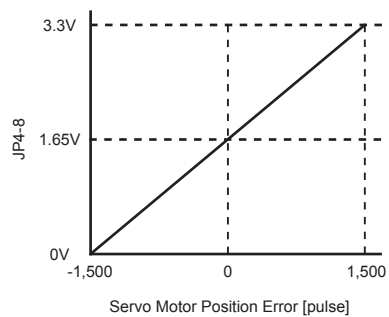
| Servo Motor Absolute Position | JP4-22 Analog Output |
|-------------------------------|----------------------|
| 65,536/32 | 3.3V |
| 4,096/32 | 0.21V |
| 0 | 0V |

$$\text{Time per motor rotation [s]} = 32 * t$$

$$\text{Servo motor rotation speed [rpm]} = 1,920 / t$$

Position Error Monitor (JP4-8)

The servo motor position follow error can be monitored from JP4-8. The position error can be monitored relative to both positive (CCW from motor shaft) and negative (CW from motor shaft) directions. The analog output has a domain of [0V : 3.3V].



| Servo Motor Position Error | JP4-8 Analog Output |
|----------------------------|---------------------|
| 0 | 1.65V |
| 1,500pulse (8.24°) | 3.3V |
| -1,500pulse (-8.24°) | 0V |

5 Parameters and Tuning

5.1 Parameters Outline

The following parameters are adjustable by connection through RS232 or USB interface from the servo drive to the PC. No matter the command mode, the JP2 RS232 port is always active for parameter setting and drive configuration.

The Drive Configuration and Servo Status are stored in the EEPROM of servo drive when the save button is pressed or parameter setting is issued through the serial communication.

The guaranteed write cycle for the EEPROM is 1 million times. Do not use serial communication to constantly change the drive parameters as this will decrease servo drive life span. Major parameter change and setting should only be done during initial testing and tuning. Actual drive operation should not require constant parameter changes unless changing servo control modes on the fly through RS232.

| Parameter Name | Setting Range | Details | Applicable Servo Mode |
|------------------|---------------|--|--------------------------------------|
| Main Gain | [1 : 127] | The main gain for the servo loop, usually to be increased as the motor load increases. The bigger value of MainGain means wider frequency range of servo loop relatively. | Position Speed Torque RS232 |
| Speed Gain | [1 : 127] | The speed gain for the servo loop, usually to be increased as the motor load increases. The bigger value of Speed Gain means narrower frequency range of servo loop relatively. Physically, heavier loads or higher inertia loads should have lower dynamic ability, so the servo loop frequency range should be more narrow by using bigger value of Speed Gain. If the Speed Gain is too high, there will be noise and vibration in servo motor because the torque command will be too coarse, not smooth. For higher Speed Gain settings, a smaller Torque Constant (see TrqCons) setting can be used to attenuate noise and vibration. | Position Speed Torque RS232 |
| Integration Gain | [1 : 127] | There is an integrator in the servo loop to ensure the error between position command and real position be zero during the steady state. Also that integrator will let servo have more ability to attenuate the outside disturbance torque. The bigger value of Integration Gain, the more ability of the servo to attenuate the outside disturbance torque. Integration Gain should be decreased for heavier loads or higher inertia loads. | Position Speed Torque RS232 |
| Torque Constant | [1 : 127] | TrqCons is a first order filter constant, the bigger value means wider frequency range of that filter. That filter can be expressed as : $a / (S + a)$, here $a = 26 * \text{TrqCons}$, if $\text{TrqCons} = 100$, then $a = 2600$. That filter is used to make the torque sent to torque loop more smooth, especially for heavier loads when bigger SpeedGain is used. If a very quick response servo with small load is desirable, a bigger value or even the max value 127 should be used to ensure the dynamic stability. | Speed Torque RS232 |

| Parameter Name | Setting Range | Details | Applicable Servo Mode |
|-------------------|------------------|--|--|
| Max Acceleration | [1 : 127] | Determines the S-curve acceleration when using RS232 command mode. Also controls the response time of the first order low pass filter in speed and torque servo control (soft start). *Max Acceleration is automatically reset to 12 after power OFF | RS232 Speed Torque |
| Max Speed | [1 : 127] | Determine the S-curve max speed when using RS232 mode to make point to point motion linear/circular. *Max Speed is automatically reset to 24 after power OFF | RS232 |
| Driver ID | [1 : 126] | Every drive has a unique ID number, which can be assigned. Parameter only settable when <i>RS485/232 Net</i> box not set (checked), and there is only one Drive connected through the RS232 port. The default ID number for every Drive is 0. This ID number can be used for the network connection of RS485 or for drive unit identification purposes. When <i>RS485/232 Net</i> box is checked and there are more than one Drive connected to the physical RS485/232 network, only the setting for the Drive with the indicated ID number in the DMMDRV ServoSetting module can be read out or saved. | Position Speed Torque RS232/485 |
| On Position Range | [1 : 127] | On position range is a value used for determining whether the motor has reached the command position. On position range is selectable according to user's requirement. Suppose the Pset is the commanded position, and Pmotor is the real motor position, if $ Pset - Pmotor \leq OnPosRange$ it is said motor is ON the commanded position, otherwise not. On position range defined as: $OnPosRange \times 4/65,536 \times 360(\text{deg}) = \text{"On Position"} (\text{deg})$ Ex. If <i>OnPosRange</i> is set to 24, then: $24 \times 4/65536 \times 360 = 0.53\text{deg}$ JP4. Pin 6 Phototransistor becomes ON when motor within 0.53degree of command position. | Position Speed Torque RS232 |
| Gear Num | [500 : 16,384] | The amount of motor travel with reference to the number of input pulses is set using the parameter Gear_Num. The number of reference pulse needed for one complete motor revolution is calculated as, $One\ motor\ revolution = 4 \times GEAR_NUM$ For example, if Gear_Num is set to 4,096, then 16,384 pulses are needed from the host controller for the motor to rotate one complete revolution. | Position RS232 |
| Line Num | [500 : 4,095] | The number of pulse outputs per motor revolution from JP5 encoder output. Number of pulse output calculated as: Pulse output per rev = LINE_NUM * 4 If LINE_NUM = 2,000 then 8,000 pulses will be output per servo motor revolution. Pulse output active for all servo modes. | Position Speed Torque RS232 |

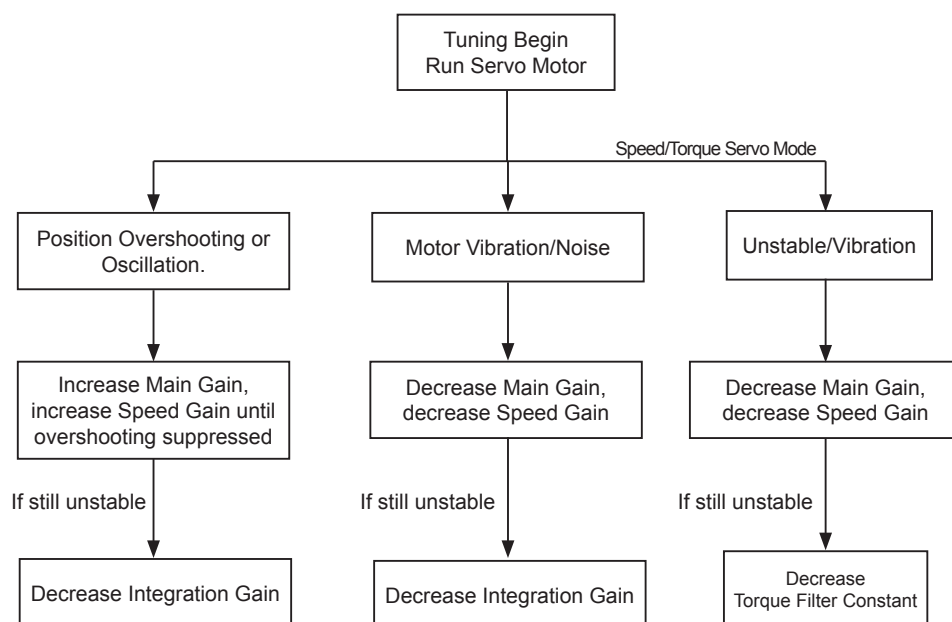
The DYN4 servo drive features simple 3 parameter Gain tuning to achieve optimized smooth performance. The user should adjust the servo gain parameters Main Gain, Speed Gain and Integration Gain until they achieve target response qualities. These parameters are all adjustable using the DMMDRV program, or by external controller via RS232 serial communication.

The overall method of Gain tuning follows as load mass or load inertia increase, the Main Gain and Speed Gain parameters should be increased. If these are set too high, the servo may be over-tuned and start vibrating or become noisy. The parameters should be increased/decreased until the motor smoothly follows command without vibration, noise or oscillations. The Integration Gain increase servo response stiffness. Integration Gain should be higher for lower load inertia quick, rapid movements (high acceleration) and lower for higher load inertia slow and sweeping movements (low acceleration). Once the load becomes smooth and stable, the user can then fine tune the parameters to make the motor “harder” (faster response, more rigid motion) or “softer” (slower response, smoother motion).

Adaptive Tuning algorithm optimize response time and torque vibration. As long as the 3 gain parameters are close to ideal settings relative to load inertia, the servo will achieve best response.

The servo motor should be coupled to the final machine before tuning. Make sure during tuning, the motor is running the load and speed of the final application. The user should use a trial and error method to achieve the desired servo response.

♦ Gain Tuning Procedure Flow



Sample Load Type Tunings

◆ Ball Screw

Ball screw systems are mechanically very rigid. If high resolution pitch (e.g. 5mm or 10mm) the default setting could even be used. The servo drive can be easily tuned using Main Gain, Speed Gain, and Integration Gain. Increase Main Gain, Speed Gain and Integration Gain relative to load mass until target response achieved. Decrease Integration Gain if load inertia is big and oscillates when moving or stopping.

◆ Direct Mechanical Drive (Rigid systems, Robots)

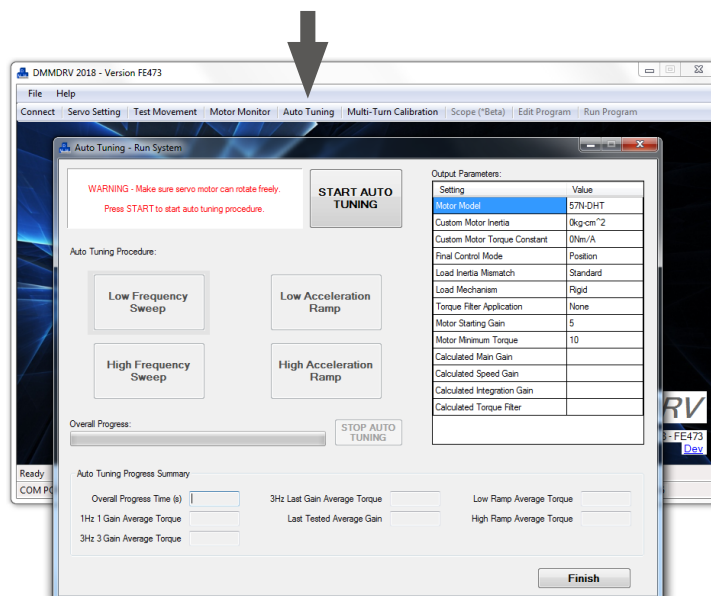
Depending on load mass and inertia, increase Main Gain, Speed Gain and Integration Gain until target response achieved. Decrease Integration Gain if load inertia is high and system oscillates. In speed/torque servo mode, if relative load inertia is very high, high Speed Gain can increase motor torque noise, then decrease the Torque Filter Constant to attenuate torque loop noise.

◆ Belt Drive / Pulley

Belt drive or pulley systems are low mechanical rigidity with slower relative response. Main Gain and Speed Gain should be increased with higher load mass and relative load inertia. Integration Gain should be decreased to give the position loop more time to react to the low rigidity belt.

Auto Tuning

Auto tuning is most often used to automatically set the servo Gain parameters. The DMMDRV program includes the Auto Tuning module. Follow the instructions in the module to allow the program and servo drive to automatically tune the Main Gain, Speed Gain, Integration Gain and Torque Filter Constant Parameters.



6 Maintenance

6.1 Alarm Specifications

The DYN4 servo drive is protected by 5 fault alarms. The S1 status indicator LED will flash to indicate when an alarm is triggered. The specific alarm status can be read using the DMMDRV program.

◆ Internal Driver Status Readout

- (1) Connect the PC to the servo drive JP2 port using RS232 cable
- (2) Press Read on the Setting driver parameters and mode main screen.
- (3) The Driver Status box will display the current status of the Servo Drive.

| Alarm | Cause | Recommended Correction |
|------------------|---|---|
| Over Voltage | The internal DC bus voltage has exceeded the allowed maximum levels. The input DC voltage is too high. | <ul style="list-style-type: none">- Check and confirm the connections to the servo motor.- Check that the servo motor is driving a mass appropriate to its size.- Check for any mechanical irregularities that might be preventing the motors to move freely.- Add an external regenerative resistor |
| Over Temperature | The servo drives protective thermal resistor has detected an unusually high temperature inside the drive. The control power transistor temperature is too high. | <ul style="list-style-type: none">- Check that the drives ventilation openings and heat sink are not being blocked.- Consult the servo drives ambient temperature specifications and check if the operation conditions are met. |
| Lost Phase | The encoder has detected an irresolvable position error in the motor relative to the command signal. | <ul style="list-style-type: none">- Check that the encoder feedback cable is securely plugged from the servo motor to the JP3 port of the servo drive.- Check for any mechanical irregularities that might be preventing the motors to move freely. |
| Over Power | The servo drive has experienced an output power exceeding the rated value relative to the average value. | <ul style="list-style-type: none">- Check and confirm the connections to the servo motor.- Check that the servo motor is driving a mass appropriate to its size.- Check for any mechanical irregularities that might be preventing the motors to move freely. |
| Over Current | Servo drive internal current increased above rated and protection levels. | <ul style="list-style-type: none">- Check that the encoder feedback cable is securely plugged from the servo motor to the JP3 port of the servo drive.- Check for any mechanical irregularities that might be preventing the motors to move freely. |

◆ Alarm Motor Stop

The power to the servo motor will be stopped when an alarm is triggered. Internal servo control turns off and servo motor shaft becomes free. Power still remains in the logic circuit for drive diagnostic and drive status reading. All commands including pulse, analog and RS232 will be ignored and internal position error will not accumulate.

◆ Alarm Reset

Once servo drive triggers an alarm, the user should use the DMMDRV program to read out the alarm condition then inspect the machine, load or operation for cause to the alarm. The problem should be fixed before re-setting the servo drive and running again. The servo drive main power should be cycled to fully re-set and clear the servo alarm status.



WARNING

- If the servo motor is coupled to a vertical axis that can drop due to gravity when alarm triggered and the shaft becomes free, take measure to prevent injury or damage when the drive alarm is triggered. A motor with holding brake option may be necessary to stop vertical axis, or any axis acted on by an external force, from dropping or crashing.

6.2 Servo Drive Maintenance

Do not perform maintenance on the servo drive unless instructed to do so by DMM. The servo drive cover or chassis should never be removed as high voltage components can cause electric short, shock or other damage upon contact. Disassembly, repairs or any other physical modification to the servo drive is not permitted unless approved by DMM.

◆ Regular Inspection

Inspect the servo drive regularly for:

- Dirt, dust or oil on the servo drive - make sure the servo drive cooling duct and heat sink are free from debris
- Environment - ambient temperature, humidity and vibration according to servo drive specification
- Loose screws
- Physical damage to servo drive or internal components

The RS232 port is always active after power on for DYN series drive. This active RS232 port could be used for reading and setting Drive parameters and status and also can be used for sending point to point position command if the RS232 mode is selected for position command input.

This DYN232M integrated motion command includes point-to-point S-curve, linear, arc and circular interpolation for up to 3-axis of coordinated motion. These profiles can be easily executed using dedicated function codes. The DYN servo drive features the most advanced built in S-Curve Generator in the industry to realize point to point S-Curve motion. Response is extremely fast and motion filters are built in to optimize stability and provide smoothest motor response. Featuring Dynamic Target Position Update (DTPU) technology, target position can be instantaneously changed (without current command completion) and robot movements can be realized with much faster cycle time and higher universal efficiency.

If the position command is selected as other modes, such as PULSE/DIR, CW/CCW, SPI or Analog mode, the RS232 port is still active as mentioned above but only can be used for reading and setting drive parameters and reading and setting drive status registers (Section 7.3).

The RS232 port can be accessed by a variety of host controllers including PC, microcontroller, FPGA, Arduino or motion controller. The host device is working as a master and the servo drive is always working as a slave. Several drives can be linked for a serial network in RS485.

RS232 Functions Include:

- ◆ Reading and changing servo drive parameters
- ◆ Reading and monitoring servo drive status including alarm, busy, in position, enable etc.
- ◆ Reading and monitoring servo drive configuration including servo mode, incremental/absolute mode, command mode, enable etc.
- ◆ Absolute encoder homing
- ◆ Absolute encoder position monitor: 16-bit absolute, 32-bit multi-turn
- ◆ Initiate generic profiles ConstSpeed, Square Wave, Sine Wave
- ◆ DYN232M motion control commands including:
 - ◆ S-Curve point to point
 - ◆ 3-axis coordinated linear motion
 - ◆ 3-axis coordinated circular motion (arc, circle, oval)
 - ◆ Incremental (relative) or absolute modes

Example Host Controllers:

- Microcontroller/Embedded Controller
- PC (windows serial port via C/C++/C#, VB, Java etc.)
- PLC/HMI with serial output
- Arduino

The sample code in Section 7.7A *Appendix : C++ Code for Serial Communication Protocol* should be used extensively to efficiently and accurately generate the RS232 data packet. Each subroutine function automatically generates data packet structure for sending command and reading from DYN servo drive.

Never use serial communication to set the Drive configuration or parameters at a fast rate. This will cause servo drive EEPROM busy in writing parameters all the time and also shorten its lifetime. The guaranteed parameter writing cycle for EEPROM is 1 million times. Once a group of parameters and configuration are set, use it until next necessary change.

7.1 Interface and Format

Connector Specifications

Connection: JP2

Connector Type: 2.54mm Pitch Rectangular

Drive Header: Molex 70553-0041

Plug Connector: Molex 50-57-9407

Recommended Wire Gauge: 0.3mm² (AWG22)

In order to connect JP2 with a PC's RS232 port, an intermediate level shift buffer is necessary [buffer component: ADM232]. The CA-MRS232-6 and CA-MTUSB-60 tuning cables has the level shift buffer built-in. RxD and TxD RS232 signal from connector JP2 is TTL/CMOS level.

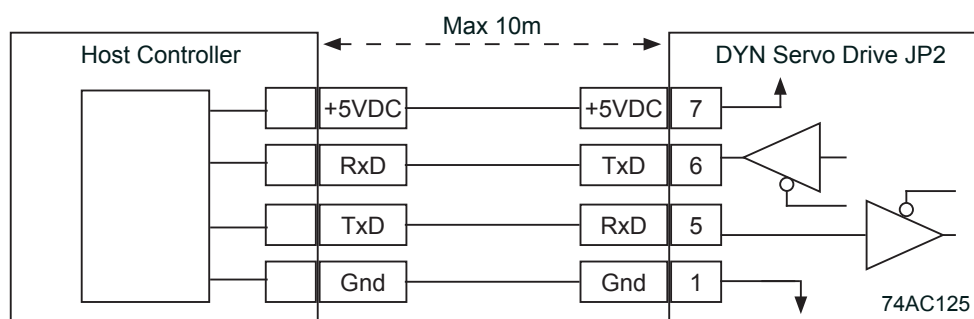
Do not connect servo drive directly to PC RS232 port without buffer component.

| | |
|----------|---|
| Pin. 1 | Gnd |
| Pin. 2~4 | Reserved |
| Pin. 5 | RS232 RxD signal input to Drive, CMOS/TTL level signal |
| Pin. 6 | RS232 TxD signal output from Drive, CMOS/TTL level signal |
| Pin. 7 | +5VDC output from Drive |



WARNING

Pin. 2 ~ 4 are reserved for factory use and are internally connected. Connecting these pins to external devices may result in permanent damage to servo drive.



Communication Format

| | |
|---------------------|---|
| Baud Rate | 38400 (Standard Mode) 312.5k (Fast Mode) ^{*1} |
| Start/Stop Bit | 1 |
| Odd/Even Verify Bit | No |
| Data | 8-bit |
| | Full Duplex |
| | Asynchronous (UART) |
| Voltage | +5V (TTL/CMOS) |

^{*1} Fast Mode is custom option - contact DMM for details. All specifications in this manual are referenced in standard mode.

■ Transmission

The DYN servo drive is always under command from host controller. When a function is called, the servo drive will move the servo motor, return a data packet with the requested information, or set a parameter value. Once a complete data packet has been received, the servo drive will not return any confirmation or acknowledgement code. The command motion will be immediately run, requested data will be returned, or new parameter is saved.

The subroutine in Section 7.5A Appendix should be implemented to automatically generate a full data packet. Otherwise, the host controller must ensure each data packet is complete and accurate before transmission.

■ Reception

The DYN servo drive follows same data packet format and structure when returning data. Each data packet is sent one byte at a time consisting of 8 data bits and two start stop bits for a total of 10 bits. Each byte will be sent sequentially until complete packet is sent.

The host controller must process received data in shift register as soon as each byte is transmitted to avoid overflow and garbage data. Alternatively, the receiver shift register buffer must have enough address to store complete packet. The DYN servo drive will send each byte immediately after another, so at 38400 baud, each byte will take approximately 260us to transmit - host controller should read or sample at this rate or faster when receiving data.

7.2 Packet Definition

7.2.1 Structure

Byte : consists of 8 bits, represented by b7b6b5b4b3b2b1b0 or b[7:0]. b7 is MSB and b0 is LSB, so called little endian. Each packet consists of several bytes, expressed as:

Packet = Bn Bn-1 Bn-2 B1 B0

Packet length = n+1, Total n+1 bytes

Bn is start byte, B0 is end byte, similar to the byte structure, Bn is MSB and B0 is LSB as little endian rule.

The integer n varies as the variation of packet length. Functionally, a packet could be expressed as:

Packet = ID + packetLength + functioncode + data + checksum

| | |
|-----------------------------|-----------------------|
| ID | One byte (Start byte) |
| packetLength + functioncode | One byte |
| data | One to four bytes |
| checksum | One byte |

Minimum packet length is 4 bytes, packet length 4 (n=3), 1 data byte.

Maximum packet length is 7 bytes, packet length 7 (n=6), 4 data byte.

Minimum packet length is 4. There is at least one data byte, for some function code that does not require data, this data byte is meaningless, or called dummy byte which can be set to any value [0~127] and does not affect the overall function of that packet.

7.2.2 Features for the byte inside a packet

The start byte takes form of 0xxxxxxx, or MSB is 0, x for 0 or 1. Any other byte except the start byte takes the form of 1xxxxxxx, where x could be 0 or 1. Most significant bit in a byte can be used for determining if it is a packet's start byte or not.

7.2.3 Start byte Bn

The MSB bit of start byte is always zero, the other seven bits are used for the Drive ID number which is set from 0 ~ 63. The ID number can also be assigned through the DMMDRV software.

ID number 127 is reserved for every drive for broadcasting purposes. In other words, 127 is general ID number. ID numbers 64 ~ 126 are internally reserved.

The communicating servo drive must be set to the same ID number to accept and execute data. The drive ID can only be set if the *RS485/232 Net* check box is not checked (in the DMMDRV software).

7.2.4 Bn-1 byte

The Bn-1 byte is used for representing the packet function and packet length.

Bn-1 = 1 b6 b5 b4 b3 b2 b1 b0

The bit b6 and b5 are for the length of packet, expressed as:

| b6 | b5 | Total packet length(=n+1) |
|----|----|---------------------------|
| 0 | 0 | 4 |
| 0 | 1 | 5 |
| 1 | 0 | 6 |
| 1 | 1 | 7 |

The bit b4~b0 are used for the packet function, expressed as:

| Function (Sent by host) | b[4:0] | Data (Bytes) | Remarks |
|---------------------------|--------|--------------|--|
| Set_Origin | 0x00 | 1(dummy) | Set current position as zero ; See section 7.4.2 |
| Go_Absolute_Pos | 0x01 | 1~4 | See section 7.4.1 |
| Make_LinearLine | 0x02 | 1~4 | |
| Go_Relative_Pos | 0x03 | 1~4 | See section 7.4.1 |
| Make_CircularArc | 0x04 | 1~4 | |
| Assign_Drive_ID | 0x05 | 1 | Assign ID to Drive; See Section 7.6 |
| Read_Drive_ID | 0x06 | 1(dummy) | |
| Set_Drive_Config | 0x07 | 1 | One byte Configuration. See Section 7.3 |
| Read_Drive_Config | 0x08 | 1(dummy) | Read Drive configuration. See Section 7.3 |
| RegisterRead_Drive_Status | 0x09 | 1(dummy) | Ask for Drive status. See Section 7.3 |
| Turn_ConstSpeed | 0x0a | 1~3 | See section 7.4.2 |
| Square_Wave | 0x0b | 1~3 | See section 7.4.2 |
| Sin_Wave | 0x0c | 1~3 | See section 7.4.2 |
| SS_Frequency | 0x0d | 1~3 | See section 7.4.2 |
| General_Read | 0x0e | 1~4 | Read Drive position set |
| ForMotorDefine | 0x0f | 1 | Internal Function - Not customer accessible |
| Set_MainGain | 0x10 | 1 | |
| Set_SpeedGain | 0x11 | 1 | |
| Set_IntGain | 0x12 | 1 | |
| Set_TrqCons | 0x13 | 1 | |
| Set_HighSpeed | 0x14 | 1 | Set MaxSpd,1~127 ; See section 7.4, 7.5 |
| Set_HighAccel | 0x15 | 1 | Set MaxAcl,1~127 ; See section 7.4, 7.5 |
| Set_Pos_OnRange | 0x16 | 1 | If Pset-Pmotor <= OnRange, then motor on Pos OnRange ;1~127 |
| Set_GearNumber | 0x17 | 2 | Gear_Number [500~16,384] ; ; See section 7.4, 7.5 |
| Read_MainGain | 0x18 | 1(dummy) | See section 7.5 Example 11 |
| Read_SpeedGain | 0x19 | 1(dummy) | See section 7.5 Example 11 |
| Read_IntGain | 0x1a | 1(dummy) | See section 7.5 Example 11 |
| Read_TrqCons | 0x1b | 1(dummy) | See section 7.5 Example 11 |
| Read_HighSpeed | 0x1c | 1(dummy) | See section 7.5 Example 11 ; See section 7.4 |
| Read_HighAccel | 0x1d | 1(dummy) | See section 7.5 Example 11 ; See section 7.4 |
| Read_Pos_OnRange | 0x1e | 1(dummy) | See section 7.5 Example 11 |
| Read_GearNumber | 0x1f | 1(dummy) | See section 7.5 Example 11 ; See section 7.4 |

| Functions (Sent by DYN drive) | b[4:0] | Data (Bytes) | Remarks |
|-------------------------------|-------------|--------------|--|
| Not used | 0x00 ~ 0x0a | | *Do not read or write to these addresses |
| Is_MainGain | 0x10 | 1 | Returns [1:127] unsigned data |
| Is_SpeedGain | 0x11 | 1 | Returns [1:127] unsigned data |
| Is_IntGain | 0x12 | 1 | Returns [1:127] unsigned data |
| Is_TrqCons | 0x13 | 1 | Returns [1:127] unsigned data |
| Is_HighSpeed | 0x14 | 1 | Returns [1:127] unsigned data |
| Is_HighAccel | 0x15 | 1 | Returns [1:127] unsigned data |
| Is_Drive_ID | 0x16 | 1 | Returns [1:127] unsigned data |
| Is_PosOn_Range | 0x17 | 1 | Returns [1:127] unsigned data |
| Is_GearNumber | 0x18 | 2 | |
| Is_Status | 0x19 | 1 | |
| Is_Config | 0x1a | 1 | |
| Is_AbsPos32 | 0x1b | 1~4 | |
| Is_TrqCurrent | 0x1e | 1~4 | |

Functions 0x10 ~ 0x1e are sent from the DYN drive in response to a function to request data. For example, when Read_MainGain 0x18 is sent to the DYN2 drive, Is_MainGain 0x10 is returned as the function with the Main Gain value as the data. See section 7.5 Example 11.

7.2.5 Bn-2 ~ B1 bytes

Bn-2 ~ B1 (n>2) are used for representing the data in the packet. 7bits of a byte is used for containing the data. The first bit MSB is always 1.

| n | Data Range | Remark |
|---|----------------------------|-------------------------|
| 3 | -64 ~ 63 | Only B1 is used |
| 4 | -8,192 ~ 8,191 | Only B2, B1 are used |
| 5 | -1,048,576 ~ 1,048,575 | B3, B2, B1 are used |
| 6 | -134,217,728 ~ 134,217,727 | B4, B3, B2, B1 are used |

Minimum packet length is 4. There is at least one data byte, for some function code that does not require data, this data byte is meaningless, or called dummy byte which can be set to any value [0~127] and does not affect the overall function of that packet.

7.2.6 B0 Byte

B0 byte is used for check sum, which is calculated from Bn~B1 as:

$$S = B_n + B_{n-1} + B_{n-2} + \dots + B_1$$

$$B_0 = 0x80 + \text{Mod}(S, 128), B_0 = 0x80 + S - 128 * [S/128]$$

$$B_0 = 128 \sim 255$$

After receiving a packet, then calculate $\text{Temp} = \text{Mod}(S, 128)$, if $\text{Temp} = B_0$, there is no error, otherwise there is error during the packet transmission.

Example manual calculation:

Given: Command to rotate ID=8 motor at 50rpm constant speed

Packet Length = 4

$$n = 3$$

$$B_3 = 0x08$$

$$B_2 = 0x8a$$

$$B_1 = 0xb2$$

$$S = B_3 + B_2 + B_1 = 0x144 = 324$$

$$B_0 = 0x80 + \text{Mod}(S, 128)$$

$$= 0x80 + \text{Mod}(324, 128)$$

$$= 0x80 + 0x44$$

$$B_0 = c4$$

7.3 Drive Configuration and Status Register

Drive configuration such as command input mode (RS232, CW/CCW etc.), alarm status, busy status are described by the two register Config and Status which are stored inside Drives EEPROM and can be read or set through RS232 communication.

■ Drive Status

Driver status is a byte data, lower 7 bit valid for indicating the Drive status, is it in the state of servo, alarm, on position, or free.

Status = x b6 b5 b4 b3 b2 b1 b0

- b0 = 0 : On position, i.e. $|P_{set} - P_{motor}| \leq OnRange$
- b0 = 1 : motor busy, or $|P_{set} - P_{motor}| > OnRange$
- b1 = 0 : motor servo
- b1 = 1 : motor free
- b4 b3 b2 = 0 : No alarm
 - 1 : motor lost phase alarm, $|P_{set} - P_{motor}| > 8192(steps), 180(deg)$
 - 2 : motor over current alarm
 - 3 : motor overheat alarm, or motor over power
 - 4 : there is error for CRC code check, refuse to accept current command
 - 5~ 7 : TBD
- b5 = 0 : means built in S-curve, linear, circular motion completed; waiting for next motion
- b5 = 1 : means built in S-curve, linear, circular motion is busy on current motion
- b6 : pin2 status of JP3, used for Host PC to detect CNC zero position or others

■ Drive Configuration

Drive configuration for communication mode, servo mode etc is expressed by Config.

Config = x b6 b5 b4 b3 b2 b1 b0

- b1 b0 = 0 : RS232 mode
 - 1 : CW,CCW mode
 - 2 : Pulse/Dir or (SPI mode Optional)
 - 3 : Analog mode
- b2 = 0 : works as relative mode(default) like normal optical encoder
- b2 = 1 : works as absolute position system, motor will back to absolute zero or POS2(Stored in sensor) automatically after power on reset.
- b4 b3 = 0 : Position servo as default
 - 1 : Speed servo
 - 2 : Torque servo
 - 3 : TBD
- b5 = 0 : let Drive servo
- b5 = 1 : let Drive free, motor could be turned freely
- b6 : TBD

The default Config = x0000000, RS232 communication mode, absolute position sensor works as relative mode, position servo, servo enabled. If the bit 5 of Config register is set to be 1, Drive will let motor shaft free (servo disabled).

7.4 Common Function Details

7.4.1 Point to Point Movement (S-Curve)

Max Acceleration, Max Speed, and Gear Number are important data parameters for generating the S-Curve. The DYN servo drive also applies a smoothing filter to the acceleration profile to generate best S-Curve performance. The S-Curve profile is calculated as the following,

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

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

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

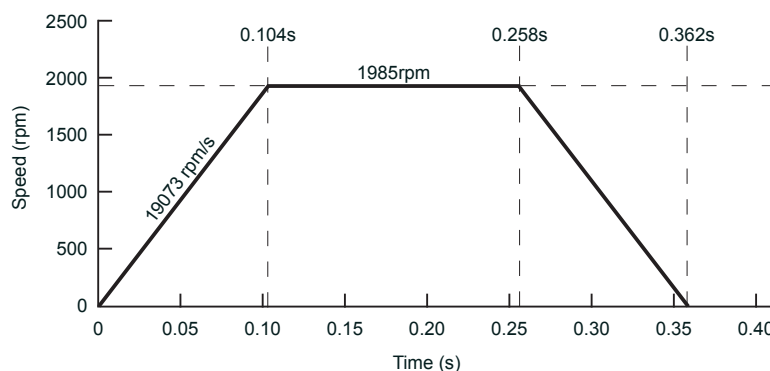
$$\text{Motor Movement Position} = \text{Command Position} * \text{Gear Ratio} * 4$$

Example:

| Set parameter | Output |
|----------------------------|---|
| 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 | Motor Movement Position = 560,000 positions |

S-Curve:

Acceleration Time = 0.104 s
 Distance During Acceleration = 1.72 rev
 Constant Speed Travel Time = 0.154 s
 Total S-Curve Time = 0.362 s



7.4.2 Constant Speed, Square Wave, Sin Wave

■ Turn Constant Speed

| Function (Sent by host) | b[4:0] | Data (Bytes) |
|----------------------------------|--------|--------------|
| Turn_ConstSpeed | 0x0a | 1~3 |

The servo motor rotates at constant speed according to the rpm speed set by the Data Bytes. The direction of rotation is CW (as viewed from shaft side) for positive speed and CCW for negative speed.

Example:

| Set command | Motion |
|---|--|
| Function = 0x0a (Turn_ConstSpeed) Data = 0x578 (1,400) (use 2 data bytes B2, B1) B2 = 1000 0101 *MSB must be 1 B1 = 1111 1000 *MSB must be 1 | Servomotor rotates in CW direction (as viewed from shaft side) at 1,400rpm |
| Function = 0x0a (Turn_ConstSpeed) Data = 0xff88 (-120) (use 2 data bytes B2, B1) 0xff88 = 0x1111 1111 1000 1000 B2 = 1111 1111 B1 = 1000 1000 | Servomotor rotates in CCW direction (as viewed from shaft side) at 120rpm |

■ Square Wave Motion

| Function (Sent by host) | b[4:0] | Data (Bytes) |
|----------------------------------|--------|--------------|
| Square_Wave | 0x0b | 1~3 |
| SS_Frequency | 0x0d | 1~3 |

The servo motor makes a square wave motion with instantaneous acceleration and deceleration command. The amplitude is set by the Square_Wave function Data and the frequency is set by the SS_Frequency function Data Bytes. The motion is executed as soon as the Square_Wave function is received. Note that Square_Wave and Sin_Wave shares the same SS_Frequency data value. The square waveform is generated internally within the DYN servo drive.

■ Sine Wave Motion

| Function (Sent by host) | b[4:0] | Data (Bytes) |
|----------------------------------|--------|--------------|
| Sin_Wave | 0x0c | 1~3 |
| SS_Frequency | 0x0d | 1~3 |

The servo motor makes a sine wave motion with continuous acceleration and deceleration. The amplitude is set by the Sin_Wave function Data and the frequency is set by the SS_Frequency function Data Bytes. The motion is executed as soon as the Sin_Wave function is received. Note that Sin_Wave and Square_Wave shares the same SS_Frequency data value. The sine waveform is generated internally within the DYN servo drive.

7.5 Dynamic Target Position Update (DTPU)

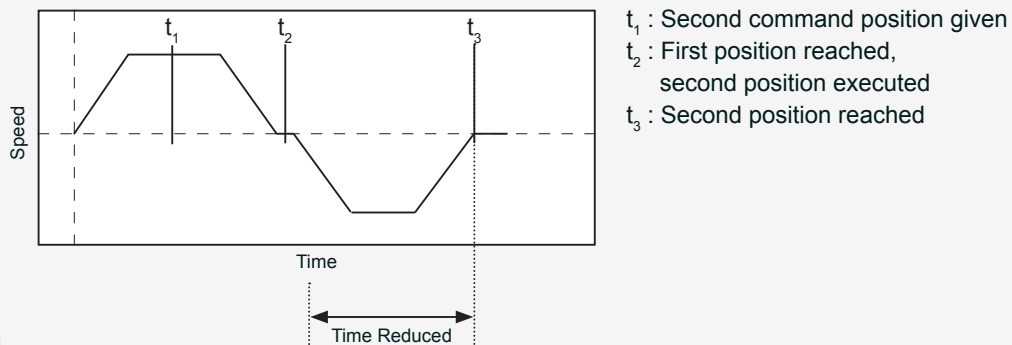
The DYN servo drive's built in S-Curve generator is able to update the target position instantaneously regardless of whether the current command position has completed or not. As soon as a new command position is received, the servo drive immediately updates the servomotor target to the newest position. 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.

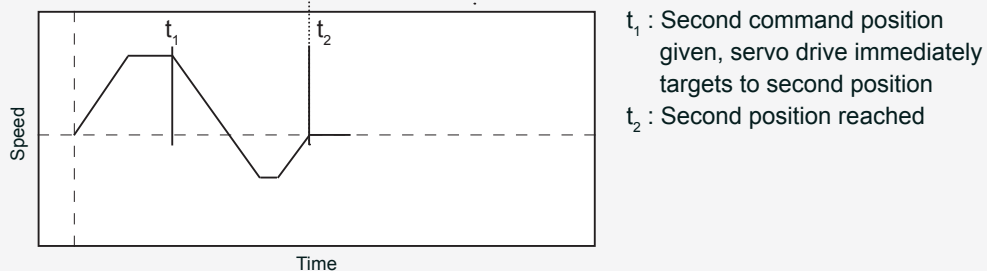
The servo drive also applies a curved acceleration command to the S-Curve to maintain smoothest servo motor motion. At each S-Curve "transition" point, the normally rigid path is curved into smooth speed transitions.

■ Efficiency

Without DTPU



With DTPU

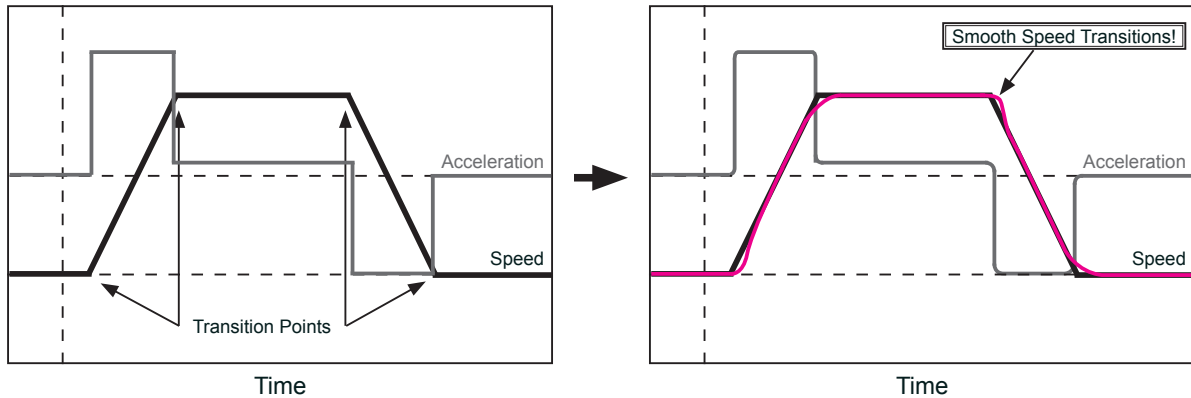


When the axis is command to a new position, the servo drive immediately updates the target position and generates new S-Curve profile to reach new target position. Without DTPU technology, the axis must first finish its current command before executing the next one, causing a delay in the overall positioning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

Curved Acceleration

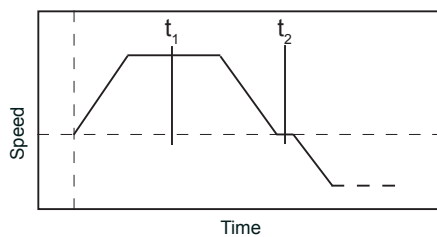
The DTPU algorithm also applies a curved acceleration to maintain smooth motion. At each S-Curve transition point, the acceleration/deceleration is curved at the edges so speed is smoothly changed. This decreases motor vibration. The smoothing is applied relative to total command movement so overall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



Safety

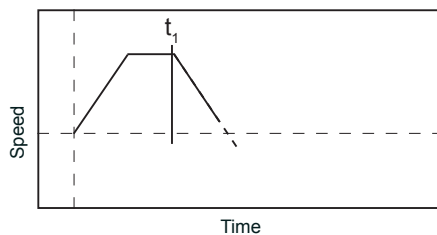
Dynamic Target Position Update DTPU allows the axis to be commanded as soon as a safety hazard or warning is detected. This means protection measures can be executed immediately. Without DTPU, the axis must finish the current positioning before executing protection measures.

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

7.6 Packet Structure Examples

| Sent Packet (to DYN drive) | | |
|---|---------------|--------------|
| Function | Function Code | Data (Bytes) |
| Set_Drive_Config | 0x07 | 1 |
| <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">Bn 0xxxxxxx</div> <div style="text-align: center;">Bn-1 10000111</div> <div style="text-align: center;">Bn-2 1 b6 b5 b4 b3 b2 b1 b0</div> <div style="text-align: center;">B0 1xxxxxxx</div> </div> <div style="text-align: center; margin-top: 10px;"> <p>Drive configuration setting</p> </div> | | |
| Sent Packet (from DYN drive) | | |
| None | | |

| Sent Packet (to DYN drive) | | |
|---|---------------|--------------|
| Function | Function Code | Data (Bytes) |
| Read_Drive_Config | 0x08 | 1 (dummy) |
| <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">Bn 0xxxxxxx</div> <div style="text-align: center;">Bn-1 10001000</div> <div style="text-align: center;">Bn-2 1 b6 b5 b4 b3 b2 b1 b0</div> <div style="text-align: center;">B0 1xxxxxxx</div> </div> <div style="text-align: center; margin-top: 10px;"> <p>Dummy bits</p> </div> | | |
| Received Packet (from DYN drive) | | |
| Is_Config | 0x1a | 1 |
| <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">Bn 0xxxxxxx</div> <div style="text-align: center;">Bn-1 10011010</div> <div style="text-align: center;">Bn-2 1 b6 b5 b4 b3 b2 b1 b0</div> <div style="text-align: center;">B0 1xxxxxxx</div> </div> <div style="text-align: center; margin-top: 10px;"> <p>Packet Length = 4 Function = 0x1a</p> </div> | | |

| Sent Packet (to DYN drive) | | |
|---|---------------|--------------|
| Function | Function Code | Data (Bytes) |
| Read_Drive_Status | 0x09 | 1 (dummy) |
| <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">Bn 0xxxxxxx</div> <div style="text-align: center;">Bn-1 10001000</div> <div style="text-align: center;">Bn-2 1 b6 b5 b4 b3 b2 b1 b0</div> <div style="text-align: center;">B0 1xxxxxxx</div> </div> <div style="text-align: center; margin-top: 10px;"> <p>Dummy bits</p> </div> | | |
| Received Packet (from DYN drive) | | |
| Is_Status | 0x19 | 1 |
| <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">Bn 0xxxxxxx</div> <div style="text-align: center;">Bn-1 10011001</div> <div style="text-align: center;">Bn-2 1 b6 b5 b4 b3 b2 b1 b0</div> <div style="text-align: center;">B0 1xxxxxxx</div> </div> <div style="text-align: center; margin-top: 10px;"> <p>Packet Length = 4 Function = 0x19</p> </div> | | |

7.7 Application Examples

■ EXAMPLE 1

Condition:

Make 3rd axis motor right now position be the absolute zero. position(= 0), ID = 3. One byte dummy data 0x00, Packet Length = 4.

Method:

B3 = 0x03
B2 = 0x80 + (PacketLength-4)*32 + Set_Origin = 0x80 + 0x00 = 0x80
B1 = 0x80 + 0x00 = 0x80
S = B3 + B2 + B1 = 0x03 + 0x80 + 0x80 = 0x103
B0 = 0x80 + Mod(S,128) = 0x83

As shown in the Sample Code, by calling the subroutine:

Send_Package(0x03,0), when Global_Func = (char)Set_Origin = 0x00.

The code will generate above B3~B0.

The motor power on position is the default absolute zero position, or it is the position set by using set absolute zero function (0x00).

■ EXAMPLE 2

Condition:

Make 3th axis motor back to absolute zero position(= 0), ID = 3. Move to position 0 = 0x00, One byte data, PacketLength = 4.

Method:

B3 = 0x03
B2 = 0x80 + (PacketLength-4)*32 + Go_Absolute_Pos = 0x80 + 0x01 = 0x81
B1 = 0x80 + 0x00 = 0x80
S = B3 + B2 + B1 = 0x03 + 0x81 + 0x80 = 0x104
B0 = 0x80 + Mod(S,128) = 0x84

■ EXAMPLE 3

Condition:

Make 3th axis motor move 120(steps) from right now position, ID = 3.

120 = 0x78 = 0x0111 1000 > 63, Two byte data, high 7bits 000 0000 = 0x00, lower 7bits = 111 1000 = 0x78. And use function Go_Relative_Pos (=0x03), Packet Length = 5.

Method:

B4 = 0x03
B3 = 0x80 + (PacketLength-4)*32 + Go_Relative_Pos = 0x80 + 0x03 = 0xa3
B2 = 0x80 + 0x00 = 0x80
B1 = 0x80 + 0x78 = 0xf8
S = B4 + B3 + B2 + B1 = 0x03 + 0xa3 + 0x80 + 0xf8 = 0x21e
B0 = 0x80 + Mod(S, 128) = 0x80 + 0x1e = 0x9e

■ EXAMPLE 4

Condition:

Make 3th axis motor move -120(steps) from right now position, ID = 3.

Method:

-120 = 0x88 = 0xff88 < -63, Two byte data.

0xff88 = 0x1111 1111 1000 1000:

Lower 7bits = 000 1000 = 0x08 Higher 7bits = 0111 1111 = 0x7f

Use function Go_Relative_Pos(=0x03), Packet Length = 5.

B4 = 0x03;

B3 = 0x80 + (PacketLength-4)*32 + Go_Relative_Pos = 0x80 + 0x04 = 0xa3.

B2 = 0x80 + 0x7f = 0xff

B1 = 0x80 + 0x08 = 0x88

S = B4 + B3 + B2 + B1 = 0x03 + 0xa3 + 0xff + 0x88 = 0x22d

B0 = 0x80 + Mod(S, 128) = 0x80 + 0x2d = 0xad

■ EXAMPLE 5

Condition:

Make 2th axis motor turn at 60rpm, ID = 2.

Method:

Speed is 60, One Byte data is enough, 60 = 0x3c. Packet Length = 4.

B3 = 0x02;

B2 = 0x80 + (PacketLength-4)*32 + Turn_ConstSpeed = 0x80 + 0x0a = 0x8a

B1 = 0x80 + 0x3c = 0xbc

S = B3 + B2 + B1 = 0x02 + 0x8a + 0xbc

B0 = 0x80 + Mod(S, 128) = 0xc8

■ EXAMPLE 6

Condition:

Make 2th axis motor turn at -60rpm, ID = 2. Speed is -60 = 0xc4 = 0x1100 0100 > -63, One byte data 7bits = 0x0100 0100 = 0x44. Packet Length = 4.

Method:

B3 = 0x02;

B2 = 0x80 + (PacketLength-4)*32 + Turn_ConstSpeed = 0x80 + 0x40 + 0x0a = 0x8a

B1 = 0x80 + 0x44 = 0xc4

S = B3 + B2 + B1 = 0x02 + 0x8a + 0xc4 = 0x150

B0 = 0x80 + Mod(S, 128) = 0x80 + 0x50 = 0xd0

■ EXAMPLE 7

Condition:

Make a line on X-Y Plane

Suppose right now position for three motors are $(X0, Y0, Z0) = (0, 0, 0)$, and the End point of straight line is $(X1, Y1, Z1) = (100, 200, 0)$

Method:

Always use General ID = 0x7f

The Feedrate = 3, could be from 1~127

Global_Func = (char)Make_LinearLine = 0x02;

Then send four packets to the Drives as:

Send_Package(ID, X1 - X0), i.e. Send_Package(0x7f, 100)

Send_Package(ID, Y1 - Y0), i.e. Send_Package(0x7f, 200)

Send_Package(ID, Z1 - Z0), i.e. Send_Package(0x7f, 0)

Send_Package(ID, FeedRate), i.e. Send_Package(0x7f, 3)

After the X-Y-Z three Drives received all four packets, they will start to move until they meet the end point of $(X1, Y1, Z1)$. Three motors will meet $(X1, Y1, Z1)$ at the same time.

During the linear or circular interpolation motion, the Read_Drive_Status (=0x09) can be used to read Drives status register to check whether b5 = 0 or not, b5 = 0 means the coordinated motion is finished.

Send_Package(ID, Y1 - Y0) is the subroutine in the SAMPLE CODE, it will generate a packet as above examples.

■ EXAMPLE 8

Condition:

Make a circular arc on X-Y Plane

Suppose right now position for three motors are $(X0, Y0) = (0, 0)$, and the End point of arc is $(X1, Y1) = (200, 0)$ in CW direction. It is easy to know the center of arc is $(Xc, Yc) = (100, 0)$

Method:

The Feedrate = 1, could be from 1~127 > 0, because in CW direction otherwise be negative value.

The planeNumber = 0 because it is in X-Y plane

TwoBytes = (PlaneNumber << 8) | FeedRate = 0*256 + 1 = 1

Use General ID = 0x7f

Global_Func = (char)Make_CircularArc = 0x04;

Then send five packets to the Drives as:

Send_Package(ID, X0 - Xc), i.e. Send_Package(0x7f, -100)

Send_Package(ID, Y0 - Yc), i.e. Send_Package(0x7f, 0)

Send_Package(ID, X1 - Xc), i.e. Send_Package(0x7f, 100)

Send_Package(ID, Y1 - Yc), i.e. Send_Package(0x7f, 0)

Send_Package(ID, TwoBytes), i.e. Send_Package(0x7f, 1)

After the X-Y-Z three Drives received all four packets, Only two of three motors will move and finally will meet $(X1, Y1)$ at the same time. During the linear or circular interpolation motion, the Read_Drive_Status (=0x09) can be used to read Drives status register to check whether b5 = 0 or not, b5 = 0 means the coordinated motion is finished.

Two equal half arcs must be made to make a circle.

The following three examples makes use of the sample code in *Section 7.7A Appendix : C++ Code for Serial Communication Protocol*. All contents of the sample code must be copied to the program.

■ EXAMPLE 9

| |
|---|
| Condition: |
| Read servo motor absolute position |
| Method: |
| <p>Call ReadMotorPosition32() subroutine function</p> <p>Motor position stored in Motor_Pos32 variable as:</p> <p>Motor_Pos32 = (long) [$-2^{27} : 2^{27}-1$] = [-134,217,728 : 134,217,727]</p> |

■ EXAMPLE 10

| |
|---|
| Condition: |
| Read servo motor torque current |
| Method: |
| <p>Call ReadMotorTorqueCurrent() subroutine function</p> <p>Motor torque current stored in MotorTorqueCurrent variable as:</p> <p>MotorTorqueCurrent = (short) [$-2^{15} : 2^{15}-1$] = [-32,767 : 32,766]</p> <p>MotorTorqueCurrent represents a relative number according to the RMS current output by servo drive. This value is different between each servo motor capacity and varies between the DYN2 and DYN4 servo drive. The customer can measure the change in MotorTorqueCurrent variable to monitor relative current draw. Use servo motor torque constant specification to calculate torque output.</p> |

■ EXAMPLE 11

| |
|---|
| Condition: |
| Read servo drive Main Gain parameter |
| Method: |
| <p>Call ReadMainGain() subroutine function</p> <p>DYN drive Main Gain stored in MainGain_Read variable</p> <p>Use the same subroutine format for all Parameter Read functions 0x18~ 0x1f.</p> |

7.8 RS485 Network

Several Drives can be connected by RS485 after every Drive on the RS485 net have been designated an individual, or broadcasting ID number.

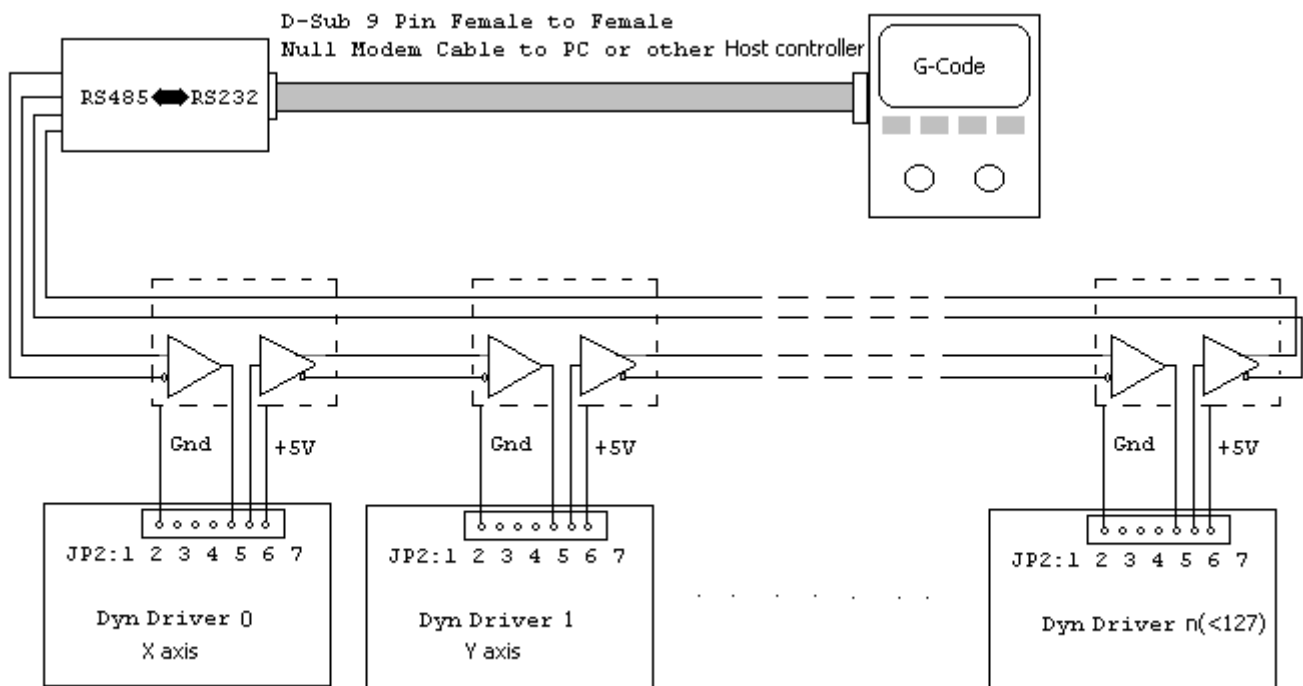
The RS485 check box must be checked if RS485 network is used which means there are at least two or more Drive on the net, then every servo drive status and configuration can be read or set according to the ID number on the servo setting dialog box. The ID number cannot be assigned to a particular Drive if RS485 network is connected.

The Servo Drive ID number CAN ONLY BE SET when there is only ONE drive connected, then assigned a new ID number to that drive without checking the RS485/232 Net check box (in the DMMDRV software).

The RS485 network is a serial network, if there is a packet in the network, one Drive will receive it first, if the packet's ID number is the same as the Drives, that packet will be received and processed by the Drive, otherwise that packet will be relayed to the next Drive.

The Drive ID is contained in the first byte of the packet. When a packet is received, the drive only reads the first byte, it will receive if ID is correct and relay to next drive if ID does not match. Data flow on the serial RS485 net is very fast and efficient.

Every drive has a RS485NET node which contains a RS485 buffer such as LTC491.



The following code shows an example to generate a data packet and call functions in RS232 serial protocol.



Note: in the description of RS232 communication protocol above (Section 7), the last byte of packet is always B0, but in the code of below, the first byte is always B0.

```
#define Go_Absolute_Pos          0x01
#define Is_AbsPos32              0x1b
#define General_Read             0x0e
#define Is_TrqCurrent            0x1e
#define Read_MainGain            0x18
#define Is_MainGain              0x10

char InputBuffer[256];           //Input buffer from RS232,
char OutputBuffer[256];         //Output buffer to RS232,
unsigned char InBfTopPointer, InBfBtmPointer; //input buffer pointers
unsigned char OutBfTopPointer, OutBfBtmPointer; //output buffer pointers
unsigned char Read_Package_Buffer[8], Read_Num, Read_Package_Length, Global_Func;
unsigned char MotorPosition32Ready_Flag, MotorTorqueCurrentReady_Flag, MainGainRead_Flag;
long Motor_Pos32;
int MotorTorqueCurrent, MainGain_Read;

void DlgRun::ReadPackage()
{
    unsigned char c, cif;

    ReadRS232Port();             // Include customer code to read from serial port

    while(There is data in the customer hardware RS232 receiving Buffer)
    {
        InputBuffer[InBfTopPointer] = HardwaerRS232ReceiveBuffer;    //Load InputBuffer with received packets
        InBfTopPointer++;
    }

    while(InBfBtmPointer!=InBfTopPointer)
    {
        c = InputBuffer[Comm.InBfBtmPointer];
        InBfBtmPointer++;
        cif = c&0x80;
        if(cif==0)
        {
            Read_Num = 0;
            Read_Package_Length = 0;
        }
        if(cif==0||Read_Num>0)
        {
            Read_Package_Buffer[Read_Num] = c;
            Read_Num++;
            if(Read_Num==2)
            {
                cif = c>>5;
                cif = cif&0x03;
                Read_Package_Length = 4 + cif;
                c = 0;
            }
            if(Read_Num==Read_Package_Length)
            {
                Get_Function();
                Read_Num = 0;
                Read_Package_Length = 0;
            }
        }
    }
}
```

```

void DlgRun::Get_Function(void)
{
    char ID, ReceivedFunction_Code, CRC_Check;
    ID = Read_Package_Buffer[0]&0x7f;
    ReceivedFunction_Code = Read_Package_Buffer[1]&0x1f;
    CRC_Check = 0;
    for(int i=0;i<Comm.Read_Package_Length-1;i++)
    {
        CRC_Check += Read_Package_Buffer[i];
    }
    CRC_Check ^= Read_Package_Buffer[Comm.Read_Package_Length-1];
    CRC_Check &= 0x7f;
    if(CRC_Check!=0){
        //MessageBox("There is CRC error!") - Customer code to indicate CRC error
    }
    else
    {
        switch(ReceivedFunction_Code){
            case Is_AbsPos32:
                Motor_Pos32 = Cal_SignValue(Read_Package_Buffer);
                MotorPosition32Ready_Flag = 0x00;
                break;
            case Is_TrqCurrent:
                MotorTorqueCurrent = Cal_SignValue(Read_Package_Buffer);
                MotorTorqueCurrentReady_Flag = 0x00;
                break;
            case Is_MainGain:
                MainGain_Read = Cal_SignValue(Read_Package_Buffer);
                MainGainRead_Flag = 0x00;
                break;
            default:;
        }
    }
}

/*Get data with sign - long*/
long DlgRun::Cal_SignValue(unsigned char One_Package[8])
{
    char Package_Length,OneChar,i;
    long Lcmd;
    OneChar = One_Package[1];
    OneChar = OneChar>>5;
    OneChar = OneChar&0x03;
    Package_Length = 4 + OneChar;
    OneChar = One_Package[2];
    OneChar = OneChar<<1;
    Lcmd = (long)OneChar;
    Lcmd = Lcmd>>1;
    for(i=3;i<Package_Length-1;i++)
    {
        OneChar = One_Package[i];
        OneChar &= 0x7f;
        Lcmd = Lcmd<<7;
        Lcmd += OneChar;
    }
    return(Lcmd);
}

```

/*First byte 0x7f, bit 6 represents sign */

/* Sign extended to 32bits */

/* Lcmd : -2²⁷ ~ 2²⁷ - 1 */

```

//***** Every Robot Instruction *****
// Send a package with a function by Global_Func
// Displacement: -2^27 ~ 2^27 - 1
// Note: in the description of RS232 communication protocol above (Section 7), the last byte of packet is //
always B0, but in the code of below, the first byte is always B0.

void DlgRun::Send_Package(char ID , long Displacement)
{
    unsigned char B[8],Package_Length,Function_Code;
    long TempLong;
    B[1] = B[2] = B[3] = B[4] = B[5] = (unsigned char)0x80;
    B[0] = ID&0x7f;
    Function_Code = Global_Func & 0x1f;
    TempLong = Displacement & 0x0ffffff; //Max 28bits
    B[5] += (unsigned char)TempLong&0x0000007f;
    TempLong = TempLong>>7;
    B[4] += (unsigned char)TempLong&0x0000007f;
    TempLong = TempLong>>7;
    B[3] += (unsigned char)TempLong&0x0000007f;
    TempLong = TempLong>>7;
    B[2] += (unsigned char)TempLong&0x0000007f;
    Package_Length = 7;
    TempLong = Displacement;
    TempLong = TempLong >> 20;
    if(( TempLong == 0x00000000) || ( TempLong == 0xffffffff))
    { //Three byte data
        B[2] = B[3];
        B[3] = B[4];
        B[4] = B[5];
        Package_Length = 6;
    }
    TempLong = Displacement;
    TempLong = TempLong >> 13;
    if(( TempLong == 0x00000000) || ( TempLong == 0xffffffff))
    { //Two byte data
        B[2] = B[3];
        B[3] = B[4];
        Package_Length = 5;
    }
    TempLong = Displacement;
    TempLong = TempLong >> 6;
    if(( TempLong == 0x00000000) || ( TempLong == 0xffffffff))
    { //One byte data
        B[2] = B[3];
        Package_Length = 4;
    }
    B[1] += (Package_Length-4)*32 + Function_Code;
    Make_CRC_Send(Package_Length,B);
}

```

```

void DlgRun::Make_CRC_Send(unsigned char Plength,unsigned char B[8])
{
    unsigned char Error_Check = 0;
    for(int i=0;i<Plength-1;i++)
    {
        OutputBuffer[OutBfTopPointer] = B[i];
        OutBfTopPointer++;
        Error_Check += B[i];
    }
    Error_Check = Error_Check|0x80;
    OutputBuffer[OutBfTopPointer] = Error_Check;
    OutBfTopPointer++;

    while(OutBfBtmPointer != OutBfTopPointer)
    {
        RS232_HardwareShiftRegister = OutputBuffer[OutBfBtmPointer];
        SendRS232Port();           // Include customer code to send to RS232 port
        OutBfBtmPointer++;         // Change to next byte in OutputBuffer to send
    }
}

void DlgRun::ReadMotorTorqueCurrent(void)
{ /*Below are the codes for reading the motor torque current */

    char ID = 0;                    //Read motor torque current
    Global_Func = General_Read;      //Suppose read 0 axis motor
    Send_Package(ID , Is_TrqCurrent);

    //Function code is General_Read, but one byte data is : Is_TrqCurrent
    //Then the drive will return a packet, Function code is Is_TrqCurrent
    //and the data is 16bits Motor torque current.

    MotorTorqueCurrentReady_Flag = 0xff;
    While(MotorTorqueCurrentReady_Flag != 0x00)
    ReadPackage();

    //MotorTorqueCurrentReady_Flag is cleared inside ReadPackage() or inside
    //Get_Function() exactly after the MotorTorqueCurrent is updated.
}

```

```

void DlgRun::ReadMotorPosition32(void)
{
    /*Below are the codes for reading the motor shaft 32bits absolute position */

    char ID = 0;
    Global_Func = General_Read;
    Send_Package(ID , Is_AbsPos32);

    //Read motor 32bits position
    //Suppose read 0 axis motor

    // Function code is General_Read, but one byte data is : Is_AbsPos32
    // Then the drive will return a packet, Function code is Is_AbsPos32
    // and the data is 28bits motor position32.

    MotorPosition32Ready_Flag = 0xff;
    While(MotorPosition32Ready_Flag != 0x00)
    ReadPackage();

    // MotorPosition32Ready_Flag is cleared inside ReadPackage() or inside
    // Get_Function() exactly after the Motor_Pos32 is updated.
}

void MoveMotorToAbsolutePosition32(char MotorID,long Pos32)
{
    char Axis_Num = MotorID;
    Global_Func = (char)Go_Absolute_Pos;
    Send_Package(Axis_Num,Pos32);
}

void ReadMainGain(char MotorID)
{
    char Axis_Num = MotorID;
    Global_Func = (char)Read_MainGain;
    Send_Package(Axis_Num, Is_MainGain);

    MainGainRead_Flag = 0xff;
    while(MainGainRead_Flag != 0x00)
    {
        ReadPackage();
    }
}

```

```

void main(void)
{
    /* (1) Move motor 2 to absolute position of 321,456 - Method 1*/
    char Axis_Num = 2;
    Global_Func = (char)Go_Absolute_Pos;
    long pos = 321456;
    Send_Package(Axis_Num,Pos);

    /* (2) Move motor 2 to absolute position of 321,456 - Method 2 - Using subroutine function*/
    MoveMotorToAbsolutePosition32(2,321456);

    /* (3) Code for reading the motor shaft 32bits absolute position - Method 1
        This method uses a while delay to wait for Send_Package() function to complete
    */
    int i;
    InBfTopPointer = InBfBtmPointer = 0;           //reset input buffer pointers
    OutBfTopPointer = OutBfBtmPointer = 0;         //reset output buffer pointers

    for(i=0;i<8;i++)
        Read_Package_Buffer[i] = 0;

    Read_Num = Read_Package_Length = 0;

    //Reading motor 32bits position
    char ID = 0; //Suppose read 0 axis motor
    Global_Func = General_Read;
    Send_Package(ID , Is_AbsPos32);

    while(i<10000)                                //10~20ms waiting
    {
        i++;
    }

    ReadPackage();                                //Motor absolute position stored in Motor_Pos32 variable

    /* (4) Reading the motor shaft 32bits absolute position - Method 2 using subroutine function*/
    ReadMotorPosition32();                        //Motor absolute position stored in Motor_Pos32 variable

    /* (5) Reading the motor current using subroutine function*/
    ReadMotorTorqueCurrent(); //Motor torque current stored in MotorTorqueCurrent variable

    /* (6) Reading the main gain of 8th axis servo drive using subroutine function*/
    ReadMainGain(8);                             //Main Gain stored in MainGain_Read variable
}

```

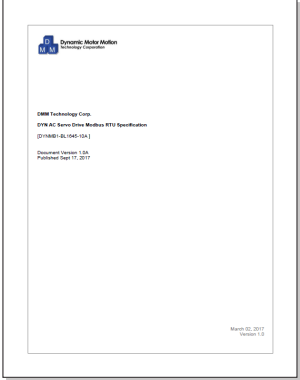

Sample Code Notes:

(1) The sample code uses a ring buffer structure to input and output data packet bytes. Two separate ring buffers are using in the code as *char InputBuffer[256]* and *char OutputBuffer[256]*.

Two position pointers are used in each buffer structure to index the data inside the buffer structure. For example, when a data packet is received from the servo drive, each byte received is sequentially saved into the InputBuffer with the InBfTopPointer incremented each time. This is done until the host hardware RS232 receiver buffer is empty, meaning all packet bytes have been read and stored. Data is processed as first-in-first-out (FIFO) queue and starts at the index of InBfBtmPointer. InBfBtmPointer is incremented each time a byte is processed until InBfBtmPointer=InBfTopPointer, meaning all packet bytes have been processed.

8 Modbus RTU (RS485) Communication


The DYN4-□□□B2-00 servo drive models are compatible with Modbus RTU communication over 2-Wire RS485.
Please refer to the following manual for Modbus communication specification:

| | |
|--------------------------------|--|
| Document Number: DYNMB1-BL1645 | DYN AC Servo Drive Modbus RTU Specification <div></div> |
|--------------------------------|--|

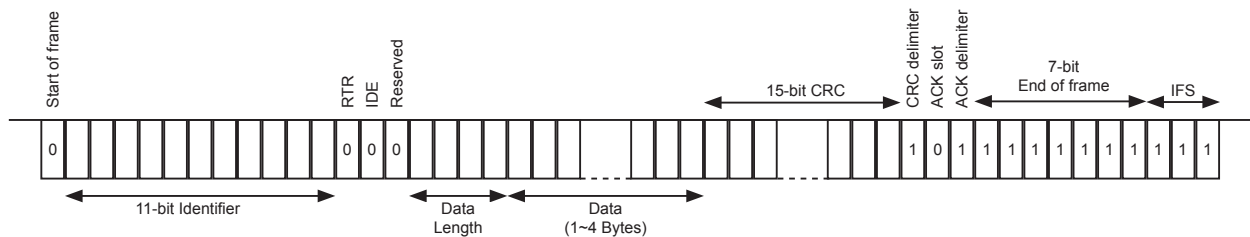
9 CAN Communication

The DYN4-□□□C2-00 servo drive models are compatible with CAN 2.0A specification. The data frame format is a proprietary DYN servo drive format with efficient data packaging and high transmission rates up to 1Mbit/s for fastest cycle time.

Please refer to the CAN communication manual for detailed specifications:

| | |
|------------------------------------|---|
| Document Number: DYNCAN1-BL314-12A | DYN AC Servo Drive CAN Specification  |
|------------------------------------|---|

DYN servo drive CAN Protocol Data Framing:



11-bit Identifier Consists both Drive ID and Command Function Code:

b4~b0 = 5-bit Function Code

b5~b10 = Drive ID 0~64
0 = Broadcast

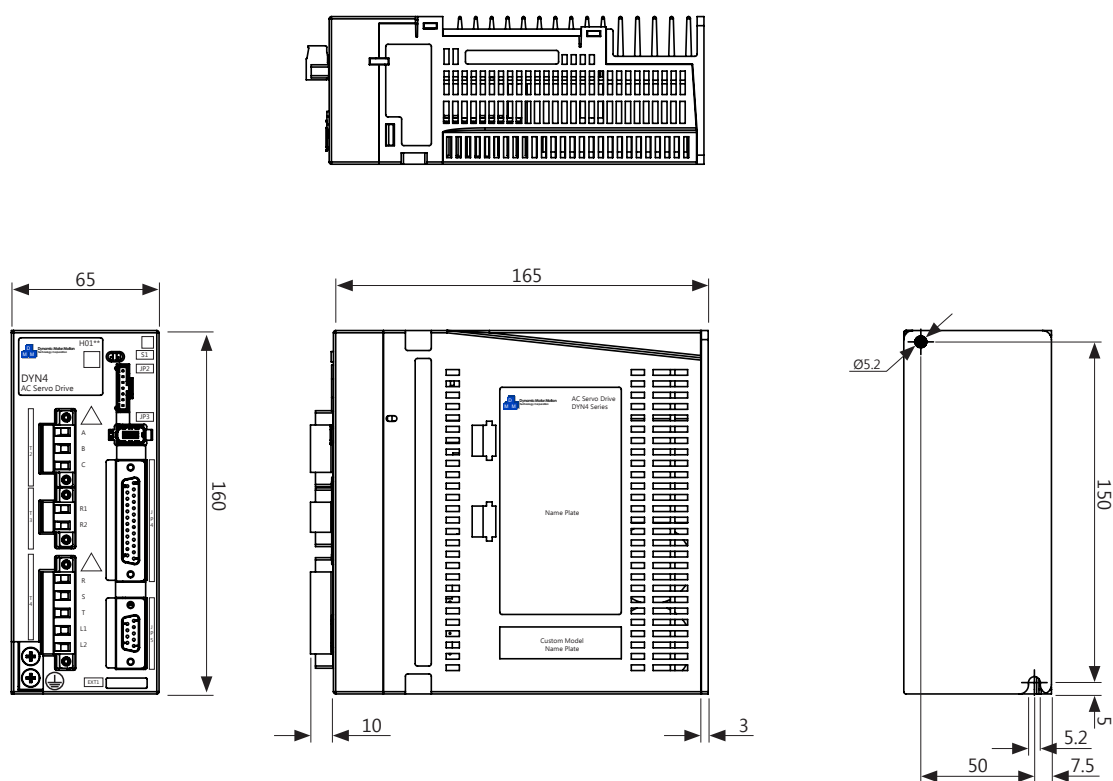
Function Code:

| # | CAN Command | 5-bit Function Code | Data Length (Bytes) |
|-----|---------------------|---------------------|---------------------|
| 0 | Set_Origin | 0x00 | 0 |
| 1 | Go_Absolute_Pos_PTP | 0x01 | 1~4 |
| 2 | Make_LinearLine | 0x02 | 1~4 |
| 3 | Go_Relative_Pos_PTP | 0x03 | 1~4 |
| ... | ... | ... | ... |

Appendix

Appendix A - Servo Drive Dimensions

Dimensions common between all DYN4 servo drive L01, H01 and T01 frame sizes.



■ Cable Spacing

Leave at least 60mm space in front of the servo drive for cable spacing. Ensure the cables are not pulling on the connectors or pins.

1) The wires used to connect AC power into logic power L1/L2 and main control power R/S/T should be at least 12AWG (3.5mm² cross-section area) and have a temperature rating of 75 degree Celsius. Using smaller gauge wires can restrict current flow into the drive and cause dips/spikes in the voltage line.

2) Power ON both logic L1/L2 and control R/S/T circuits simultaneously. Normally, logic L1/L2 can be power up first, then control R/S/T afterwards. But if the input voltage source is unstable, powering R/S/T draws high in-rush current, which can spike the line voltage for L1/L2 and cause over-voltage error.

3) In addition to the noise filter connected to the logic input, also connect an EMI filter (NF2) at the control R/S/T input to minimize motor PWM noise from affecting other drives or other devices from common and differential mode noise.

For three-phase input at control R/S/T, select the appropriate filter depending on Delta or Wye configuration. Use separate noise filters for each servo drive.

Do not share the same NF1 Logic filter between multiple drives. Ensure the noise filter is properly grounded.

4) If the servo drive commonly faults with over-voltage error during operation, connect a regenerative resistor into R1/R2 terminal. The regenerative circuit can also be activated when the input voltage is too high so take notice of the temperature of the regenerative resistor. If the temperature is unusually high, especially when there is little or no motion on the motor, refer to Point 2, and connect a transformer or input line reactor to the power supply.

5) The servo drive's max AC voltage input is 240VAC allowed to fluctuate 10%, or up to 264VAC. If the voltage exceeds 260VAC at the input, use a transformer or an Input AC line reactor to stabilize the voltage at 240VAC. Select line reactor current according to rated current of motor.

| Servo Motor Model | Motor Rated Current | Recommend Reactor |
|-------------------|---------------------|---|
| 880-DST (750W) | 4.4A | Automation Direct: LR-20P5-1PH (Single Phase) LR-21P0 (Three-Phase) MTE: RL-00402 (Three-Phase) |
| 120-DST (1.8kW) | 16.7A | Automation Direct: LR-23P0-1PH (Single-Phase) LR-25P0 (Three-Phase) MTE: RL-01802 (Three-Phase) |

6) Use the same power supply voltage for logic L1/L2 and control R/S/T. For example, when using a transformer with 120VAC at the primary voltage and 240VAC at the secondary voltage, connect the 240VAC to both L1/L2 and R/S/T. Do not connect 120VAC into L1/L2 and 240VAC into R/S/T.

The DYN4 servo drive parameters can be used to fine tune the Analog reference input specification. If the analog input is within the deadzone, the servo drive will read the command as zero.

1) For Analog input, the MaxSpeed parameter is the dead zone:

$$\{(MaxSpd-1)*11/8\}*12000mV/4096 = 0 \text{ to } 506 \text{ mV Dead Zone, } 1 \text{ for } 0, 127 \text{ for } 506mV$$

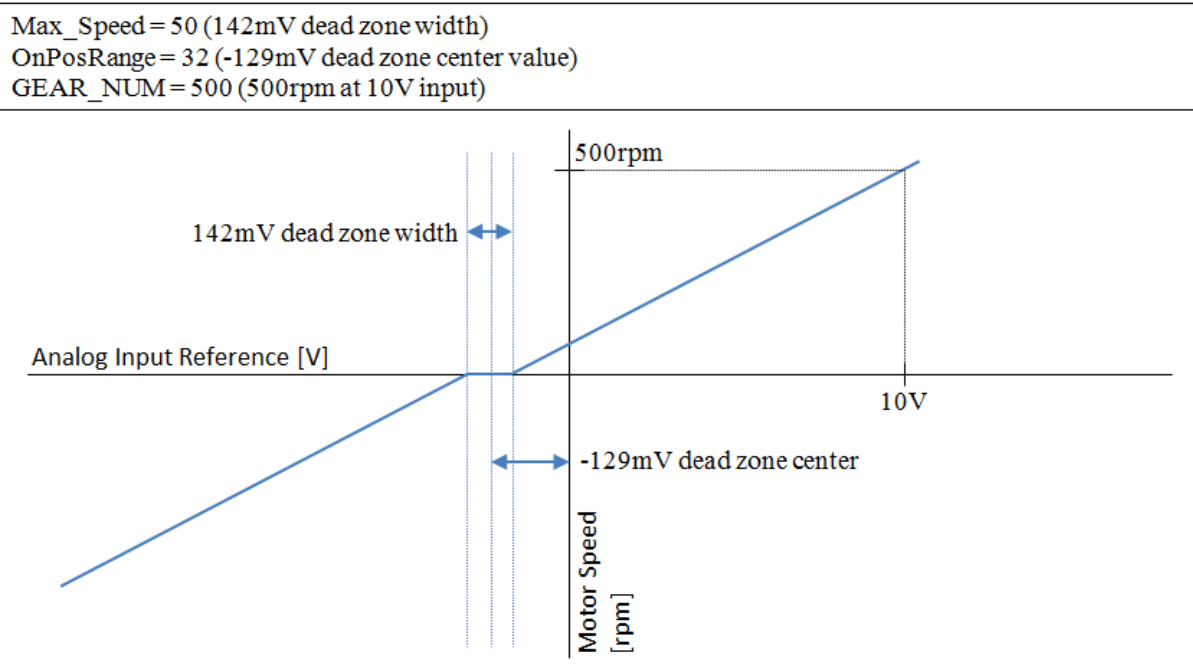
2) For Analog input, the OnPosRange parameter is the dead zone center value:

$$\{(OnPosRange - 64)*11/8\}*12000mV/4096 = -253 \text{ to } 253 \text{ mV, Dead Zone Center}$$

Ex. OnPosRange = 64, center will be zero

3) Analog speed input reference scaling. For analog input speed servo mode, GEAR_NUM parameter is used to scale the analog reference speed or torque. GEAR_NUM is the motor speed at 10V reference input. For example, if GEAR_NUM is set to 1500, then the motor runs at 1500rpm at 10V input, and 750rpm at 5V input.

4) Reversing rotation direction in Analog input mode. Normally, servo motor rotates in CW direction with positive reference voltage. By applying +5V to JP4 Pin22 and 0V (Gnd) to Pin10, the rotation direction is reversed to CCW. This direction change can be made on-the-fly during normal operation.



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 invoice.

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 DYN4 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 DYN4 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 customer.

■ Disclaimer

DMM Technology Corp. constantly strive to improve its product performance and reliability. The contents of this manual outlines the latest features and specifications of the DYN4 AC Servo Drive and may be changed at any time to reflect corrections, improvements or changes to the product or information in this manual.

■ Servo Drive Revision

| Frame Size | L01 | H01 | T01 |
|------------------|----------------|----------------|----------------|
| Hardware Version | PTH1-L01 | PTH1-47 HL01 | PTH1-T01 L1 |
| Software Version | PTS1-L01 | PTS1-26 HL01 | PTS1-T01 L1 |
| Release Date | September 2018 | September 2018 | September 2018 |
| Version Notes | - | - | - |

■ Document Revisions ■

| Published | Revision | Int Ref. | Section | Revised Content |
|----------------|----------|----------|---|---|
| December 2018 | A1.0A | ZM7 | -- 1.2 1.5 1.6 2.3.1 2.4 3.5 5.1 5.2 | - CE codes updated - Model Designation Updated - Multi-Turn Encoder option added - Updated motor model numbers - Revision to I/O specifications - JP5 encoder output updated - Wiring diagrams updated - LINE_NUM parameter increased to 4095 - Auto tuning information added |
| September 2017 | A1.8A | ZM5 | 1.2 1.3 2.5 8 9 Appendix B | - Added B and C Command type model number - Updated CE Certification - Added single phase input example - Modbus RTU (RS485) Communication - CAN Communication - Added Application Note# AP15-48 - DYN4 Servo Drive - AC Power Supply Guidelines |
| July 2016 | A1.7a | ZM1 | 1.2 1.3 1.6 2.5 | - Added L01 and T01 model information - Revised wiring diagram - Added encoder cable shield grounding notice |
| December 2015 | A1.5b | 05U | 7.4.2 2.5 2.3.2 2.3 5.1 4.5 3.4.1 1.5 All | Added RS232 specifications - Added multiple drive connection - Regenerative Resistor notes Sourcing circuit diagram JP4 I/O corrections and additions - Added Max Acceleration and Max Speed parameter reset after power ON - Parameter detail added Changed PULSE_NUM to LINE_NUM New DMMDRV software program information Added ABS-14-00 14-bit encoder option information Various design and layout improvements |
| March 2015 | A1.5 | DR1 | 7 All | - Added DYN232M Section - Formatting improvements |
| January 2015 | A1.2 | M22 | 4.1 2.2 | - Revised command pulse specifications - Added JP3 encoder feedback pinout |

DYN4 Series

AC Servo Drive

AC SERVO DRIVE

INSTRUCTION MANUAL

TYPE A - GENERAL PURPOSE PULSE / ANALOG / DYN232M

TYPE B - MODBUS

MODEL: DYN4 - □□□ A, DYN4 - □□□ B, DYN4 - □□□ C

MANUAL CODE: DYN4MS-ZM7-A10A

REVISION: A1.0A

Electronic Version

Copyright © 2018 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



Dynamic Motor Motion
Technology Corporation