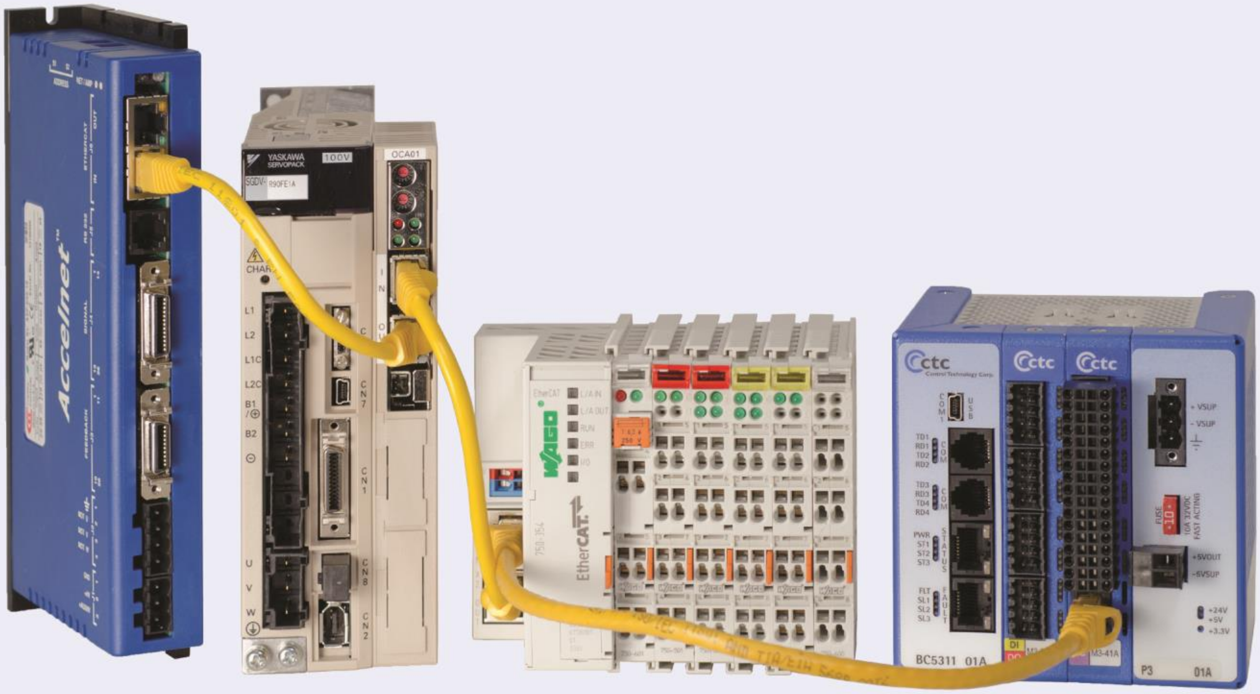
ctc
Control Technology Corp.

EtherCAT Applications Guide



Control Technology Corporation
Hopkinton, MA • ctc-control.com

CONTROL TECHNOLOGY CORPORATION

M3-41 EtherCAT Applications Guide

EtherCAT Applications Guide

April 23, 2017

Blank

⚠ **WARNING:** Use of CTC Controllers and software is to be done only by experienced and qualified personnel who are responsible for the application and use of control equipment like the CTC controllers. These individuals must satisfy themselves that all necessary steps have been taken to assure that each application and use meets all performance and safety requirements, including any applicable laws, regulations, codes and/or standards. The information in this document is given as a general guide and all examples are for illustrative purposes only and are not intended for use in the actual application of CTC product. CTC products are not designed, sold, or marketed for use in any particular application or installation; this responsibility resides solely with the user. CTC does not assume any responsibility or liability, intellectual or otherwise for the use of CTC products.

The information in this document is current as of the following Hardware and Firmware revision levels. Some features may not be supported in earlier revisions. See www.ctc-control.com for the availability of firmware updates or contact CTC Technical Support.

Model Number	Hardware Revision	Firmware Revision
5300/Incentive M3-41	All Revisions	>= V050090R70.01 >= V1.72

Blank

TABLE OF CONTENTS

[1] OVERVIEW 13

[2] BOOT SEQUENCE (CONTROLLER) 21

[3] DRIVES & I/O MAPPING 25

DRIVES & LOCAL I/O 26

[4] DRIVES: MODES OF OPERATION & GENERAL PROGRAMMING 27

CYCLIC SYNCHRONOUS POSITION MODE (CSP) 28
PROFILE VELOCITY MODE 29
INTERPOLATED POSITION MODE 30
PROFILE POSITION MODE 30
HOMING MODE 31
GEARING/TRACKING & LOCAL QUADRATURE ENCODERS 32
MOVE ON A GEAR 34
CAMMING MOVES AND OPTIONAL TIMEOUTS 37
SEGMENTED MOVES 38
SLEWED MOVE 39
LINEAR INTERPOLATION (2D) 39
LINEAR INTERPOLATION (3D) 41
CIRCULAR INTERPOLATION (2D) 43
CAMMING & SPLINES (2D/3D) – LINEAR, CUBIC & QUADRATIC 48
TORQUE CONTROL 51
RESTARTING ETHERCAT PROGRAMMATICALLY 53
SPECIAL REGISTER ACCESS 53
GLOBAL MSB REGISTERS AND FLAGS 54
ACCESSING PROPERTIES OF ANOTHER AXIS 54
DRIVE TYPE & AXIS NUMBER 55
DRIVE OBJECT ACCESS (SDO) 55

[5] REGISTRATION, ABSOLUTE POSITIONING, & DISTRIBUTED CLOCK 57

REGISTRATION 57
ABSOLUTE VERSUS INCREMENTAL POSITIONING MODES 59
DISTRIBUTED CLOCK & DC SYNC 60
ETHERCAT MASTER CONTROL LOOP CYCLE TIME 62

[6] ETHERCAT IO, PLS, & PWM 65

INPUTS/OUTPUTS 65
ETHERCAT IO CONFIGURATION 68
MAPPABLE INPUT IO & COUNTERS 69
PULSE & PWM GENERATION 70
PLS OUTPUTS 72

[7] ERROR HANDLING 77

RETRY LOGIC 77
DRIVE DIAGNOSTIC VARIABLES AND REGISTERS 77
MSB ‘WSTATUS’ VARIABLE BIT DEFINITIONS 78
MSB ‘ERRORTYPE’ VARIABLE VALUE DEFINITIONS 79
MSB ‘LAST_ALSTATUSCODE’ VARIABLE VALUE DEFINITIONS 81

[8] QUICKBUILDER ETHERCAT EXPLORER ... 83

QUICKBUILDER ETHERCAT EXPLORER STATUS WINDOW ... 83
ETHERCAT EXPLORER PROPERTIES ... 87
LOG BUFFER TIMINGS ... 89
ETHERCAT MASTER ENI CONFIGURATION FILES ... 90
USER OPTIONS ... 91

[9] TELNET COMMANDS ... 95

STATUS COMMANDS ... 95
NETWORK COMMANDS ... 95
MESSAGE LOG COMMANDS ... 96
CONFIGURATION FILE COMMANDS ... 96
FIRMWARE UPDATE COMMANDS ... 97
LICENSE COMMANDS ... 97

[10] INCENTIVE PC RUNTIME ... 99

INCENTIVE RUNTIME ... 100
QUICKBUILDER PROGRAMMING FOR WINDOWS® ... 102
INCENTIVE INSTALLATION OVERVIEW ... 103
STARTUP AND NETWORK CONFIGURATION ... 105
MAC ADDRESS FOR LARGE SYSTEMS ... 111
FILE SYSTEM ... 112
DEMO MODE AND LICENSING ... 116
TenAsys INtime Licensing ... 116
Incentive Licensing ... 118
WINDOWS® UPDATES ... 118
SERIAL PORTS ... 119
ETHERCAT ETHERNET ADAPTER ... 122
SYSTEM MANAGEMENT INTERRUPT DETECTION ... 123
PLATFORM EVALUATION ... 125
CTC INCENTIVE® .NET API ... 127
.NET API SAMPLE PROGRAM OVERVIEW ... 130
.NET API OPENING A CONNECTION LOCALLY AND REMOTE ... 132
SIMPLE API PROGRAMMING CONCEPTS ... 135
PLClogic Class ... 135
Axis Class ... 138
RuntimeManagement Class ... 140
HOT SWAP OF ETHERCAT DEVICES ... 143
AxisSupervisor Class – online/offline Methods ... 144
Offline/Online Example ... 150
REMOVAL OF CTPLC_1 FOR STANDALONE OPERATION ... 151
QUICKBUILDER PROGRAMMING AND ATOMICITY ... 155
SOME COMMON ISSUES AND RESOLUTIONS ... 156
Shutdown and Restart Leave Power Switch Lit ... 156
Bridged Network Fails after Windows Shutdown ... 156

[A] ABB ... 157

DC SYNC ... 157
DRIVE CONFIGURATION ... 158

[B] ADVANCED MOTION CONTROLS ... 159

DRIVE INFORMATION & FIRMWARE ... 159
STATION ALIAS ... 160
DC SYNC ... 160
INPUTS/OUTPUTS ... 160

Communications Error 163
DriveWare 164
EtherCAT Explorer View 175

[C] COPLEY (ACCELNET & XENUS PLUS) 177

Station Alias 177
EtherCAT Explorer View (Xenus Dual Axis) 179

[D] ELMO (NOT RELEASED) 181

Drive Information & Firmware 181

[E] EMERSON (CONTROL TECHNIQUES) 183

Drive Information & Firmware 183
Inputs/Outputs 184
DC Sync 185
Drive Enable Command 185
Station Alias 186
Menu to Object Mapping 186
RPM Limiting 186

[F] IAI 187

Parameter Information from Test Setup 187
Special Variable Mapping 190
Control Output Mapping 191
Control Status Mapping 193
Operation 195
Errors and Alarms 199

[G] KOLLMORGEN 201

Drive Information & Firmware 201
Positioning Mode 202
Powerup Delay 202

[H] LINMOT 203

Overview 203
I/O 204
Motor Voltage Levels 204
Motor Wizard 205
Control Parameter A/B 206
INTF file upgrade (.HX3) 206
Test MSB 207
Homing MSB 209
LinMot-Talk Control Panel 209
EtherCAT Explorer View 211

[I] MITSUBISHI 213

Overview 213
CN3 Connector 214
MR Configurator2 215
MR Configurator2 Miscellaneous 220
EtherCAT Explorer View 223

[J] NUMATICS (EMERSON) 225

Valve Modules 225
Input Mapping (Required for New Controllers) 225
Firmware Updates 226

POWER CONNECTOR 228

[K] OMRON GX-JC06 ETHERCAT JUNCTION SLAVES 229

CABLING 229
EXAMPLE WITH M3-41 MODULE 230

[L] SANYO DENKI 231

DRIVE INFORMATION & FIRMWARE 231
STATION ALIAS 232
SANYO DENKI DRIVE MODE TRANSITIONS 233
SANYO DENKI POWER-UP DELAY 233
R ADVANCED MODEL – SETUP SOFTWARE 234
ETHERCAT EXPLORER VIEW 238

[M] SMC CORPORATION 239

EX600 FIELDBUS SYSTEM 239
DC SYNC 240
SDO CONFIGURATION 240

[N] TURCK 243

SYNCHRONIZATION VIA HARDWARE USING THE CFG-SWITCH 243
DIAGNOSTIC LEDS 244
ETHERCAT CONNECTOR 245
RFID 245
RFID PROPERTY VARIABLES 246
TAG INFO 250
TRANS INFO 251
STRINGS 251
PROGRAMMING EXAMPLES 252

[O] YASKAWA 257

STATION ALIAS 257
YASKAWA POSITION LAG & PERR 257
DRIVE IO CONNECTOR MAPPING 259
MSB 'ERRORREGISTER' VARIABLE VALUE DEFINITIONS 259
MSB 'ERRORCODE' VARIABLE VALUE DEFINITIONS 259
ETHERCAT EXPLORER VIEW 261

[P] GENERAL ANOMALIES AND TIPS 263

NETWORK SWITCHES 263
MSBS 263
PPR/MPPR 263

[Q] TEST SUITE 265

MSB TEST MODULE 265

[R] INCENTIVE ETHERCAT MASTER & M3-41A FIRMWARE REVISIONS 271

V1.72 (INCENTIVE ONLY 271
V1.71 271
V1.70 271
V1.67 271
V1.66 272
V1.65 272
V1.64 272
V1.63 272
V1.62 272

V1.61 273
V1.60 273
V1.59 273
V1.58 273
V1.57 273
V1.56 273
V1.55 274
V1.54 274

[S] INCENTIVE PCLOGIC PROCESS 275

R70.01 275
R69.99 275

Blank

CHAPTER

1

[1] Overview

This manual is intended to be used in conjunction with the *Model 5300 QuickMotion Reference Guide* and an understanding of that manual is assumed. The model 5300 M3-40 motion module supports local drives and I/O; this manual discusses CTC motion control as it pertains to the M3-41 hardware and software module. It is an advanced EtherCAT Master providing distributed motion control using the CAN application protocol over EtherCAT. This includes both servo drives, RFID readers, and I/O devices.

Unlike EtherCAT Masters from other vendors, the M3-41 attempts to isolate the user from the complexity of the EtherCAT environment by automatically scanning the network and configuring supported devices. The programming interface uses the same high-level language that the M3-40 series of modules uses: QuickBuilder MSBs (Motion Sequence Blocks). You no longer have to deal with a complex configurator, poking drive objects, or figuring out how an interface works. Each of the supported motion and/or I/O devices has been verified with the M3-41, and all setup and initialization is done for you. This greatly simplifies an EtherCAT installation, enabling you to concentrate on motion control and your system, not a complicated configurator. Multiple EtherCAT Master Network modules (M3-41) can be intermixed with other networks such as BACnet®, Modbus®, and other modules offered within the embedded 5300 controller family.

The M3-41 EtherCAT Master is available both as a hardware device, module within the 5300 Controller, or as a soft device, executing in real-time on a Windows® based platform. In both environments the same application programming environment is used, QuickBuilder. Additionally, a .Net API called Incentive, which exposes the entire MSB language is available on a Windows® platform. The biggest difference between the two is the PC environment is limited to EtherCAT only IO while the embedded 5300 controller has numerous local IO and network possibilities, in addition to EtherCAT. Execution on a Windows based PC presents the developer with an open and diverse architecture with which to implement their automation solution. Below is a quick comparison of the two environments:

Feature	Embedded 5300 PLC	Soft 5300PC PLC, Dual Core Processor	Soft 5300PC PLC, Quad Core Processor
Executes QuickBuilder	Yes	Yes	Yes
Programmable using ‘C’ steps	Yes	Yes	Yes
Modbus, CTC Binary, UDP, TCP	Yes	Yes	Yes
Serial Port Support	Yes	Yes	Yes

Feature	Embedded 5300 PLC	Soft 5300PC PLC, Dual Core Processor	Soft 5300PC PLC, Quad Core Processor
QuickBuilder Performance	1 X	1X to 4 X	2.3 to 8 X
Local IO Expansion modules	Yes 2/4/8	No	No
Axes/EtherCAT Network	16	8 (Atom) 16 (i7)	64 (I210) 32 (other)
Max EtherCAT Networks	4	1	2
EtherCAT Segmentation	No	Yes	Yes
Fast Packet Retry on Loss	Yes	No	No
Turck RFID Channels/Network	16	8 (Atom) 16 (i7)	32
Scan Rates	500us-4ms	500us-4ms	500us-4ms
BACnet Support	Yes	TBD	TBD
Digital IN/Digital OUT	1024	1024	1024
Analog IN/Analog OUT	256	256	256

Many factors affect the maximum number of axis/network with the Soft 5300PC. The usage of an Intel I210 network adapter also significantly boosts performance. This adapter is able to periodically transmit packets via hardware whereas others (Realtek) place the overhead on the 5300PC software, thus limiting performance.

The following motion control devices are currently supported, with more available in the future:

	Cyclic Sync Position	*Interpolated*	*Homing*	*Profile Position*	*Profile Velocity*	*Registration*
ABB MicroFlex e150	✓	Not supported by manuf	Not supported by manuf	Not supported by manuf	Not supported by manuf	✓ Probe 1 & 2
ADVANCED Motion Controls (Digiflex DPE)	✓	Not supported by manuf.	✓	✓	✓	
Copley Accelnet (AEP-055-18)	✓	✓	✓	✓	✓	
Copley Xenus Plus 2 Axis (XE2-230-20)	✓	✓	✓	✓	✓	
Elmo Gold[1] (Not	✓	Not	✓	✓	✓	

	Cyclic Sync Position	Interpolated	Homing	Profile Position	Profile Velocity	Registration
Released)		supported				
Emerson[2] Digitax ST & Unidrive SP (single axis)	✓	Not supported	✓	Not supported by manuf.	Not supported by manuf.	
IAI ACON Controller[3]	Not supported by manuf.	Not supported by manuf.	✓	✓	Not supported by manuf.	
Kollmorgen AKD[4]	✓	Not supported	✓	Not supported	✓	✓ Probe 1
LINMOT C1250 & E1450[5]	✓	Not supported by manuf.	Not supported by manuf	Not supported by manuf	Not supported by manuf.	
Mitsubishi J4[6]	✓	Not supported by manuf.	✓	Not supported by manuf	Not supported by manuf	✓ Probe 1 & 2
Sanyo Denki SANMotion RS2E	✓	Not supported by manuf.	✓	✓	✓	✓ Probe 1 & 2
Yaskawa Sigma 5 (Rotary and Linear)	✓	✓	✓	✓	✓	✓ Probe 1 & 2
Virtual Axis[7]	✓					

[1] Elmo support is not currently released but is functional for test purposes.

[2] Emerson must have DC Sync enabled prior to 'drive enable' for all operations.

[3] IAI ACON Controllers do not support Profile Position mode but use a proprietary setting similar to   it with simulation provided by the M3-41 module. Only Full Direct Value Mode is supported (3).

[4] Support for Kollmorgen Interpolated Position mode was removed to allow PDO space for registration. Cyclic Sync Position mode is the preferred mode.

[5] Due to timing issues with LinMOT drives a cycle time of 2 mS or slower must be used or a PVT overflow error may occur within the drive.

[6] Probe 1 & 2 Registration has been implemented but not fully tested for this drive. Mitsubishi cannot run any slower than 2 ms scan rate (manufacturer limitation).

[7] A Virtual Axis does not impact the number of online drives licensed although does count towards the limitation of 16 axis/M3-41 module.

<u>Firmware Versions:</u>

- Kollmorgen V1.8.0.3 (must be this or greater to support Cyclic Sync Position mode)
- Elmo V1.1.6.4
- Emerson SM-EtherCAT V1.07.01, Unidrive SP02X1 V1.15, Digitax DST1201 V1.06
- Yaskawa V3.05
- ABB MicroFlex e150 build 5711.4.0
- LinMOT – See appendix.

<u>Supported I/O Devices:</u>

Beckhoff (8 bit digital IO boundaries only)

- EK1100 – EtherCAT Coupler
- Digital Inputs – EL1018
- Digital Outputs – EL2008
- Analog Inputs – EL3102, EL3112, EL 3122, EL3142, EL3152, EL3162
- Analog Outputs – EL4032, EL4102, EL4112, EL4122, EL4132

Omron

- GX-JC06 – EtherCAT Junction Slave, 6 port.
- Pending: NX-ECC201, NX-ECC202, and NX-ECC203 EtherCAT Couplers with following NX- modules:
  - ID4342 DIN-8 NPN 24VDC
  - ID4442 DIN-8 PNP 24VDC
  - ID5142-1 DIN-16 N/PNP 24VDC
  - ID5142-5 DIN-16 N/PNP 24VDC
  - ID5342 DIN-16 NPN 24VDC
  - ID5442 DIN-16 PNP 24VDC
  - ID6142-5 DIN-32 N/PNP 24VDC
  - ID6142-6 DIN-32 N/PNP 24VDC
  - OD4121 DOUT-8 TNPN 12-24VDC
  - OD4256 DOUT-8 TPNP 12-24VDC
  - OD5121-1 DOUT-16 TNPN 12-24DC
  - OD5121-5 DOUT-16 TNPN 12-24DC
  - OD5121 DOUT-16 TNPN 12-24VDC
  - OD5256-1 DOUT-16 TPNP 24VDC
  - OD5256-5 DOUT-16 TPNP 24VDC
  - OD5256 DOUT-16 TPNP 24VDC
  - OD6121 DOUT-32 TNPN 12-24DC
  - OD6121-6 DOUT-32 TNPN 12-24DC
  - OD6256-5 DOUT-32 TPNP 24VDC
  - PC0010 IOG 16 term

  Free-run mode only.

Numatics Incorporated (Emerson) 501/G3 Series (1.1 Build 42194, Boot 1.1 Build 41544)

240-310	EtherCAT Module
219-828	Valve Driver Output Module
425186-001	Atlas Valve Driver Output Module (P599AE42518800x)
240-203	16DI PNP Terminal Strip
240-204	16DI NPN Terminal Strip
240-205	16DI PNP M12 x 8
240-206	8DI PNP M12 x 8
240-207	16DO PNP M12 x 8
240-208	8DO PNP M12 x 8
240-209	16DI NPN M12 x 8
240-210	8DI NPN M12 x 8
240-211	8DI/8DO PNP M12 x 8
240-300	8DO High Current PNP M12 x 4
240-316	8DI PNP Terminal Strip
240-323	16DI PNP 19 Pin
240-330	16DO PNP Terminal Strip

*SMC Corporation EX600 (8 bit digital IO boundaries only, **not first slave due to clock issues**)*

- Digital Inputs – EX600-DX*B, C, C1, D, E, F modules
- Digital Outputs – EX600-DY*B, E, F modules
- Digital Input/Outputs – EX600-DM*E, F modules
- Analog Inputs – EX600-AXA module
- Analog Outputs – EX600-AYA module
- Analog Input/Outputs – EX600-AMB module
- Valves – EX600-SEC*, 8, 16, and 24 valves

Wago 750-354 EtherCAT Coupler

- Digital Inputs – 750-4XX modules (non-8 bit boundaries supported)
- Digital Outputs – 750-5XX modules (non-8 bit boundaries supported)
- Analog Inputs – 750-452, 453, 454, 455, 456, 457, 459, 460, 461, 462, 464, 465, 466, 467, 468, 469, 470, 472, 473, 474, 475, 476, 477, 478, 479, 480, 630
- Analog Outputs – 750-550, 551, 552, 553, 554, 555, 556, 557, 559, 560, 562, 563, 565

Turck BL20-E-GW-EC EtherCAT Coupler

- Digital Inputs (non-8 bit boundaries supported)
  - BL20-2DI-24VDC-P (2 digital inputs, 24VDC PNP switching)
  - BL20-2DI-24VDC-N (2 digital inputs, 24VDC NPN switching)
  - BL20-2DI-120/230VAC (2 digital inputs, 120/230VAC)
  - BL20-4DI-24VDC-P (4 digital inputs, 24VDC PNP switching)
  - BL20-4DI-24VDC-N (4 digital inputs, 24VDC NPN switching)
  - BL-20-E-8DI-24VDC-P (8 digital inputs, 24VDC PNP switching)
  - BL-20-E-16DI-24VDC-P (16 digital inputs, 24VDC PNP switching)

  - BL-20-16DI-24VDC-P P (16 digital inputs, 24VDC PNP switching, optocouplers)
  - BL-20-32DI-24VDC-P (32 digital inputs, 24VDC PNP switching, optocouplers)
- Digital Outputs (non-8 bit boundaries supported)
  - BL20-2DO-24VDC-0.5A-P (2 digital outputs, 24VDC PNP switching, 0.5 Amp)
  - BL20-2DO-24VDC-2A-P (2 digital outputs, 24VDC PNP switching, 2 Amp)
  - BL20-2DO-120/230VAC-0.5A (2 digital outputs, 120/230VAC)
  - BL20-2DO-R-NO (2 relay outputs, normally open)
  - BL20-2DO-R-NC (2 relay outputs, normally closed)
  - BL20-2DO-R-CO (2 relay outputs)
  - BL20-4DO-24VDC-0.5A-N (4 digital outputs, 24VDC NPN switching)
  - BL20-4DO-24VDC-0.5A-P (4 digital outputs, 24VDC PNP switching)
  - BL-20-E-8DO-24VDC-0.5A-P (8 digital outputs, 24VDC PNP switching)
  - BL-20-E-16DO-24VDC-0.5A-P (16 digital outputs, 24VDC PNP switching)
  - BL-20-16DO-24VDC-0.5A-P (16 digital outputs, 24VDC PNP switching, optocouplers)
  - BL-20-32DO-24VDC-0.5A-P (32 digital outputs, 24VDC PNP switching, optocouplers)
- Analog Inputs
  - BL20-E-8AI-U/I-4PT/NI (8 analog inputs, U/I configurable, -10/0..+10VDC & 0/4..20MA)
  - BL20-2AI-U (2 analog inputs, U/I configurable, -10/0..+10VDC & 0/4..20MA)
  - BL20-2AI-I (2 analog inputs, 0/4..20MA)
  - BL20-1AI-I (1 analog input, 0/4..20MA)
  - BL20-1AI-U (1 analog input, -10/0..+10VDC)
  - BL20-2AI-PT/NI-2/3 Temp (2 analog inputs for temperature measurement)
  - BL20-4AI-U/I (4 analog inputs, U/I configurable)
  - BL20-2AI-THERMO-PI (2 analog inputs for thermocouples)
- Analog Outputs
  - BL20-E-4AO-U/I (4 analog outputs, configurable, -10/0..+10VDC & 0/4..20MA)
  - BL20-1AO-I (1 analog output, 0/4..20MA)
  - BL20-2AO-I (2 analog outputs, 0/4..20MA)
  - BL20-2AO-I (2 analog outputs, -10/0..+10VDC)
- RFID Reader
  - BL20-2RFID-S (2 channel RFID reader)

Only digital input and digital output modules evenly divisible by 8 are supported on all slaves except Wago and Turck; this keeps the EtherCAT packet aligned for faster operation. Usage of non 8 bit divisible digital IO on Wago and Turck will affect performance due to the bit shifting required during each control loop. This typically reduces the total axes supported by 1, depending on the number of IO points. If hundreds of mis-aligned IO are used the performance impact will be greater and will require specific

application testing. Additionally SMC Corp devices must not be the first slave device, they do not support ARMW EtherCAT packets required for clock synchronization.

Blank

CHAPTER

2

[2] Boot Sequence (Controller)

The M3-41 is reset during power up or hardware reset of the model 5300 controller. Prior to the controller running its QuickBuilder application program, the M3-41 module scans the EtherCAT network and auto-configures all the drives and I/O it discovers. These resources are then reported back to the controller so it can continue its boot sequence. Once booted, no devices can be added or removed from the network without resetting the model 5300 controller.

The boot sequence begins with the M3-41 card sending a broadcast on the EtherCAT network to determine how many slaves are present, at which point the following occurs:

1. If the Power Up Delay QuickBuilder option is set, the module will initialize it's peripherals, Ethernet, etc., and then delay the specified number of seconds.
2. Each slave is interrogated and specific information is read from its EEPROM.
3. Topology, available mailboxes, and distributed clocks are determined.
4. The slave is placed in the PRE_OP state.
5. Non-volatile memory is checked to see if a configuration file is present.
    a. Configuration File present:
        i. Its contents are checked against what is observed on the network. The number and type of slaves, (VendorID and ProductCode), and position on the network (PhysAdr/Configured Address) must match for booting to continue.
        ii. If they do not match, the network waits 5 seconds and rescans, starting at step 1. This is repeated up to 4 times, after which the boot cycle is aborted and it is up to the user to intervene, by modifying either the network or the configuration file. If the 'Retry Forever' User Option is active, the retry count is ignored and the module continues to rescan the network until the configuration file is verified against that of the network. Once a match is found the boot operation continues to step 6. If no match is found, scanning continues.
    b. Configuration File not present:
        i. Operation continues to step 6.

6. If the QuickBuilder Verify Delay option is set, and this is the first time here, the module will delay the required number of seconds and then branch back to Step 2, re-broadcasting the network for the required devices. The second time here the delay will once again occur but it will then branch to Step 7.
7. Slave-specific information is sent to the slave to configure it, and the PDOs (Process Data Objects) are initialized as required for M3-41 operation. For drives, properties such as velocity, position, I/O, status, and control words are set up.
8. The PDO information is mapped based on the required input and output size. During Master scanning each slave will place its data at the appropriate offset position within the EtherCAT packet, as it makes its way along the network bus.
9. The slave is placed in the SAFE_OP state.
10. The first slave device that can support a distributed clock is determined and delay times for each slave are calculated and loaded into the slave internal objects. Distributed clock is always used, referencing the first slave that supports it. When the master scan loop executes, its time is modified each cycle to match that of the referenced slave. The Master reads the first slave clock and writes to all other slaves (0x910 object) on every scan cycle. The Master adjusts its scan cycle using a programmable interrupt timer that resides within an FPGA, with a granularity of 10 nS.
11. EtherCAT Master drivers are now installed for all drives and I/O discovered on the network. This includes any virtual drives defined. Dual-ported memory tables are updated to notify the main controller and its QuickBuilder program what resources are available.
    a. All servo drives should be installed in sequence. The first becomes axis 1, the second is axis 2, etc. This assumes that each drive's address is set to 0.
    b. If the address switches are used, they may be set from 1 to 16, each representing an axis to which MSBs are assigned. No two drives can have the same address (except for 0).
    c. Some drives have problems with their address switches, in which case the Station Alias setting must be programmed into the EEPROM using an EtherCAT Configurator, such as the Beckhoff ET9000.
12. The EtherCAT Master begins periodic scanning, adjusting the time as needed to sync to the slave device supplying the distributed clock.
13. The slaves are now placed in the OPERATIONAL state.
14. If all slaves reach the OPERATIONAL state, the QuickBuilder MSBs will begin executing, one for each of the slave servo drives discovered. I/O devices become available for access from the main controller, appearing as regular I/O (both analog and digital).
15. The MSB should now turn DC Sync on via the MSB instruction if it will be used. This is required for Emerson/Control Techniques and ABB drives.
16. A 'drive enable' command must be executed within the servo-controlled MSB in order for the power-up commands to be initiated to the slave's amplifier. <u>If a drive input is</u>

programmed as a hardware disable, and it is active, the MSB will hang executing the 'drive enable' command until the hardware enable is active.

17. A 'drive disable' command removes power from the amplifier; the MSB will still execute.

Blank

CHAPTER

3

[3] Drives & I/O Mapping

QuickBuilder assigns drive and I/O as it sees them, sequentially on the local backplane. EtherCAT is simply an extension of the backplane — as each device is seen, it is added to the local I/O and drive table. Thus the first device on the network has its I/O added to the end of the local 5300 rack I/O table as it is placed online. EtherCAT devices are typically addressed by their sequence of daisy chained wiring. By default each EtherCAT device has an automatically assigned 'Configured Address'. The lower part of this address is masked with 0x00ff and the result becomes the default address. Axis numbers are assigned as they appear on the network. The problem with this approach is that the addresses can change as devices are moved in the daisy chain of wiring. This may be fine with I/O, but does not work for mapping MSBs to the drives they are controlling. Insertion of a new drive will shift all the following axis numbers up by one. To resolve this, if the 'Station Alias' is nonzero it will override the automatic numbering and be used as the Axis assignment.

I/O may be intermixed with drives, in any order.

The I/O will be ignored in the numbering of the axes. Using the Station Alias enables you to insert other devices and change wiring without worrying about your MSBs controlling the wrong drive. Most Station Aliases are set using the drive's dip switches. Due to a firmware anomaly, Kollmorgen devices typically must have their addresses set with the EtherCAT configurator and programmed to their individual EEPROMs.

Optionally, virtual drives can be defined, starting as the first or added after the online drives. These drives run in Cyclic Sync Position mode only and execute the same as online drives. In many cases a virtual drive will be configured as a master and an online drive will track to its position. A virtual drive counts towards the 16 axis module limit since it calculated all target information identical to a real drive. The eCAT_driveType variable will contain a 9 for a Virtual axis.

The EtherCAT Master can only assign I/O and axis numbers based upon what devices respond on the network. Configuration files must be used in a production environment to ensure all the required devices are online prior to executing their controller MSBs. Differing devices power up at different times and may not initially respond to the EtherCAT Master online broadcast. Having a configuration file to compare against informs the master that it must wait for devices to come online prior to proceeding with the boot operation. Reference the EtherCAT Explorer chapter for information on how to automatically create a configuration file.

Drives & Local I/O

Some drives support inputs and outputs at the remote drive level. The MSB property ‘dins’ represents the raw inputs provided by the drive, up to 32 inputs (object 0x60FD.0). The first 10 inputs may be accessed using ‘din1’ to ‘din10’ bit properties; as with the M3-40 modules.

Outputs operate as they do on the M3-40 module, limited to 8 outputs at the remote drive level (object 0x60FE.1). Use the ‘setout’ and ‘clrout’ MSB instructions for access, where the first output is 1.

Local I/O is also present on the M3-41 module. This module has 6 inputs and 2 outputs which are global to all MSBs. The outputs are referenced as 9 and 10 when using the ‘setout’/’clrout’ instructions. The MSB property ‘global_inputs’ is used to read the 6 inputs, with the first bit being the first input. The MSB property ‘global_outputs’ can be used in addition to ‘setout’/’clrout’ for read/write operations of the local outputs.

Local global Inputs, P1 connector pins:

P1 -11	DIN1
P1-13	DIN2
P1-15	DIN3
P1 -12	DIN4
P1-14	DIN5
P1-16	DIN6

Local global Outputs, P1 connector pins:

P1 -9	DOUT1
P1-10	DOUT2

1. ‘global_inputs’ and ‘global_outputs’ axis properties may be accessed by QuickBuilder using the Axis name/property method: axisname.property. These two properties will contain the same value on all EtherCAT axes.

2. Chapter 6 discusses additional IO capabilities available from the MSB language using various IO arrays. These arrays give access not only to drive and module based IO but remote EtherCAT IO blocks such as those from Wago, Turck and Beckhoff. Some of the features include PLS, PWM, pulse, and atomic multi-bit access of 32 drive inputs/outputs, local and remote IO.

3. M3-41 soft PC Version only supports drive I/O.

CHAPTER

4

[4] Drives: Modes of Operation & General Programming

CTC's programming environment allows you to place a drive in a number of different modes, each offering unique features. The MSB 'cmode' variable controls the active mode that a drive runs in when motion commands are executed. The variable can be set programmatically and/or its initial value set via the QuickBuilder axis property sheet. The following values are currently supported for programming entry:

```
$CYCLIC_SYNC_POSITION_MODE   0
$PROFILE_VELOCITY_MODE       1
$INTERPOLATED_POSITION_MODE  3
$PROFILE_POSITION_MODE       4
$PROFILE_TORQUE_MODE         5 (not supported, reference Chapter 4, Torque Control)
$HOMING_MODE                 8
```

When setting 'cmode', either the constant predefined name can be used (starting with $) or the immediate numeric value. When using the constant make sure it is spelled correctly or it will be defined as a user variable with a default value of 0.

By default all drives are placed in Cyclic Sync Position mode at power up unless overridden by the QuickBuilder axis property sheet. This allows direct control of all motion by the M3-41 module. Initially the current position becomes the base position and is cyclically written to each drive as a holding position until commanded differently. The amplifier is only turned on once the 'drive enable' command is executed. This results in a special power-up sequence. The 'drive enable' command waits until the drive is powered up before it allows additional commands to be executed.

A typical simple move is displayed below:

```
cmode = $CYCLIC_SYNC_POSITION_MODE;  // CSP mode (default)
[top]
move at 1 for 2;
wait for in position;
delay 1000 ms;
move at 1 for -2;
wait for in position;
delay 1000 ms;
goto top;
```

In the above example, the drive will move at 1 rev/sec for 2 revolutions and then move back 2 revolutions. The move is a relative move, since it does not designate an actual position. The delay is arbitrary and for viewing purposes only.

To run in Profile Position mode, simply change cmode to a 4 or $PROFILE_POSITION_MODE and execute the same commands.

Cyclic Synchronous Position Mode (CSP)

Cyclic sync position mode provides linear interpolation, where the drive will always insert a delay of one position command. It gives the EtherCAT Master the greatest flexibility since it is directing the drive exactly where to go on each scan cycle (delayed one scan cycle by the drive). The trajectory is calculated on the fly by the M3-41 module. It also takes the most overhead, as each drive requires around 50µS for this calculation. It is the preferred mode for M3-41 operation.

Example:

```
// This is a background MSB.  Make sure inposw is set for the drive,
// typically .001 for 1048576 ppr.  Also set the ppr and mppr.  For
// Yaskawa this is typically 1048576.  Enable the drive, turning power
// on to the amplifier.  The current position will be constantly
// updated so the drive does not move.

drive enable;
zero feedback position;

// CSP mode is the default but set anyway for documentation purposes
// and await any drive settling.
cmode = $CYCLIC_SYNC_POSITION_MODE;
wait for in position;

[top]
// Begin the move, 1 rev/second for 2 revolutions
move at 1 for 2;
wait for in position;
// Delay 1 second once in position
delay 1000 ms;
// Do a relative move back 2 revolutions at 1 rev/second
move at 1 for -2;
wait for in position;
// Delay 1 second once in position
delay 1000 ms;
// Do it again, forever...
goto top;
```

When moving out of a Profile mode and back into CSP mode, make sure the following is executed first:

```
cmode = $CYCLIC_SYNC_POSITION_MODE;
wait for in position;
```

Reference the DC Sync section of this manual for additional information when using CSP mode.

Profile Velocity Mode

In Profile Velocity mode, the speed is output in accordance with the Profile acceleration and Profile deceleration. The drive attempts to maintain the velocity commanded. CSP mode can also be used for constant velocity, but when using Profile Mode, control loop time is optimized since the EtherCAT Master does not have to constantly calculate the trajectory for that drive. The following is a simple example of Profile Velocity Mode:

```
vmax = 100;       // Max velocity.
stoprate = 1;     // This is the rate (rev/sec) used on Yaskawa and
                  // Copley drives for decel when STOP.
invel_t = 1;      // Time required, in milliseconds to be at target
                  // velocity before considered AT TARGET.
invel_w = .01;    // Must be at target velocity +/- (.01 X target vel)
                  // with drive AT TARGET to satisfy move.
                  // If target is 0 then +/- .01 rev/sec

[top]

cmode = $PROFILE_VELOCITY_MODE;     // Profile Velocity mode

// When in velocity mode distance is just sign of direction.
move at 2 for 1;
wait for in position;   // This is when attain requested velocity
delay 5000 ms;

// Now speed up to 20 rev/sec in the same direction.
move at 20 for 1;
wait for in position;
delay 1000 ms;

// If the motor is not tuned may never get to here but
// this is the proper way to stop in velocity mode.
move at 0 for 1;
wait for in position;

// Ensure stopped before changing to another mode, like CSP.
stop;
wait for in position;
goto top;
```

The EtherCAT Master uses the properties ‘invel_t’ (at velocity time) and ‘invel_w’ (at velocity window) to monitor when the drive is actually AT TARGET in Profile Velocity mode.

Interpolated Position Mode

Interpolated position mode is used to control multiple coordinated axes or a single axis with the need for time-interpolation of setpoint data. In interpolated position mode, the trajectory is calculated by the EtherCAT Master and passed to the amplifier's interpolated position buffer as a set of points. The amplifier reads the points from the buffer and performs linear or cubic interpolation between them. This mode is provided for compatibility with some drives with the preferred mode being CSP. The following is a simple example of Interpolated Position Mode:

```
// This is a background MSB.  Make sure inposw is set for the drive,
// typically .001 for 1048576 ppr. Also set the ppr and mppr.  For
// Yaskawa this is typically 1048576.

// Enable the drive, turning power on to the amplifier.  The current
// position will be constantly updated so the drive does not move.

drive enable;
zero feedback position;

// Make sure we are stopped and are in CSP mode
cmode = $CYCLIC_SYNC_POSITION_MODE;
wait for in position;

// Drop into Interpolated Position Mode
cmode = $INTERPOLATED_POSITION_MODE;
[top]
// Begin the move, 1 rev/second for 2 revolutions
move at 1 for 2;
wait for in position;
// Delay 1 second once in position
delay 1000 ms;
// Do a relative move back 2 revolutions at 1 rev/second
move at 1 for -2;
wait for in position;
// Delay 1 second once in position
delay 1000 ms;
// Do it again, forever...
goto top;
```

Profile Position Mode

In Profile Position mode the drive is given a velocity, and acceleration, and final position to move to and it calculates the trajectory. There is lower overhead on the EtherCAT Master when using Profile Position Mode than for Cyclic Sync Position, since no trajectory needs to be calculated but it does impact the cyclic nature of other drives running in CSP mode given the initial SDO transmissions for profile setup. The following is a simple example of Profile Position Mode:

```
// Place the drive in Profile Position mode (Note Profile Position is
// not supported by Kollmorgen).
cmode = $PROFILE_POSITION_MODE;
[top]
```

```
// Set Profile Velocity (0x6081) to 5 rev/sec, acc (0x6083), and dec
// (0x6084).  Request move of 2 revolutions
move at 5 for 2;
// Wait until drive says we are in position
wait for in position;
delay 1000 ms;     // Delay for visual
// Set Profile Velocity (0x6081) to 5 rev/sec, acc (0x6083), and dec
// (0x6084).  Request move of -2 revolutions, relatively back to 0.
move at 5 for -2;
// Wait until drive says we are in position
wait for in position;
delay 1000 ms;     // Delay for visual

goto top;          // Continue forever
```

The properties of inpos_w variable (in position window) and inpos_t variable (in position time) can be set prior to initiating the command. ‘inpos_t’ maps to object 0x6068 and ‘inpos_w’ maps directly to the drive property ‘inposw’. ‘inpos_t’ is used in Profile Position mode to set the length of time a drive must be in the commanded position before it is determined to be ‘in position’ (inpos variable).

Homing Mode

In Homing mode the drive finds the home position based on a supplied motion profile and the designated homing method (0x6098 object). Most homing methods are generic but not all are supported by every drive. Consult the drive manufacturer’s manual to determine your drive’s homing methods.

Before requesting a Homing Move, set the following parameters:

- ***inpos_t*** – The number of milliseconds the drive should settle after finding home before the move is considered complete.
- ***homing_speed1*** – Some homing modes require multiple speeds, with this one being the speed to the switch.
- ***homing_speed2*** – Some homing modes require multiple speeds, with this one being the speed to zero or the index pulse.
- ***homing_method*** – The method that the drive manufacturer designates for the desired move. For example, 33 homes to the index pulse, while 34 does the same thing but in the opposite direction. There are numerous modes available, all controlled by the drive itself. The MSB is simply setting the move up for the drive to take control.

Once the above parameters are defined, ‘cmode’ is set to $HOMING_MODE (8) to command Homing mode and a move absolute command to 0 is initiated. The acceleration value in the move command will be used for the homing acceleration and deceleration rate. The move will not start until the ‘move’ command is executed. For example: ‘**move to** 0 **using** 10000,10000;’ initiates the drive homing move with an acceleration/deceleration profile of 10000 rev/sec^2. This is the value written to object 0x609A, Homing Acceleration. Note that some drives, such as Mitsubishi call this an acceleration/deceleration constant where upon the raw value is written. For example for a time constant of 0 on Mitsubishi the command would be: ‘**move to** 0 **using** 0, 0;’. Check the manufacturers manual for the meaning of object 0x609a. Note that Mitsubishi currently does not support the object but may in the future.

The following is a simple example of Homing Mode:

```
      drive enable;
      zero feedback position;

      // Let the drive home to the index pulse
      [home]
      inpos_t = 250;    // 250 millisecond settling (can make it anything want
                        // but this is the time the drive will wait at position
                        // before notifying QuickBuilder MSB that it is home.
      homing_method = 34;     // Yaskawa method setting for home to index pulse
      homing_speed1 = 1; // homing_speed1 is not used but set for default anyways
      homing_speed2 = 1; // in mode 34 only the homing_speed2 is used
      cmode = $HOMING_MODE;         // Homing mode for drive
      move to 0 using 10000,10000;  // Tell the drive to initiate the move with
                                    // accel of 10000.
      wait for in position;   // The drive will stop once it sees the index
                              // pulse and tpos = fpos at that position
                              // therefore we will have an offset past home in
                              // tpos/fpos, not really absolute 0.

      // Using CSP mode we can move back to absolute home position or 'zero
      // feedback' to 0 out tpos/fpos.

      // Drop back to CSP mode so we command the drive
      cmode = $CYCLIC_SYNC_POSITION_MODE;
      move at .1 to 0 using 10000,10000;  // Do any kind of absolute move back
                                          // to 0 to remove our offset.
      wait for in position;               // Wait until move is complete

      // We are now at home, 0.
```

The properties of inpos_w variable (in position window) and inpos_t variable (in position time) can be set prior to initiating the command. ‘inpos_t’ maps to object 0x6068 and ‘inpos_w’ maps directly to the drive property ‘inposw’. ‘inpos_t’ is used in Homing mode to set the settling time a drive must be in the commanded position before it is determined to be ‘in position’ (inpos variable).

Gearing/Tracking & Local Quadrature Encoders

Gearing and tracking modes work the same as with the M3-40 module, with a few enhancements to allow for the larger number of axes.

Any axis can track another axis by simply dropping into tracking mode. To reference which axis to track, use the variable ‘master_feedback’, which by default is 1. Set this variable to the axis you wish to track, set whether to reference fpos or tpos of the master axis, and then drop into tracking mode. The following is a simple example of Tracking Mode:

```
      cmode = $CYCLIC_SYNC_POSITION_MODE;
      // Establish which axis we are for, axisnum is our axis number
      // and is a new available property.
      AxisNum = axisnum;

      // ******** TRACKING MODE **********
      [SetTracking]
      zero master counters;
      master_feedback = 1;  // Set we will track axis 1
      // Set the feedback mode first so when enter tracking it is referencing
      // correct master.  'set master feedback' references the other axis
      // fposc or feedback position.  'set master target1' references the other
      // axis tpos position.
      set master feedback;
      set mode tracking;            // Enter tracking mode
      gear at 1:4;

      [stall]
      goto stall;
```

As described in the example there are two major ways to track the master axis:

set master feedback – tracks fpos (encoder feedback) of the master.
set master target1 – tracks tpos (target position) of the master.

With an analog servo fpos (encoder feedback position) is typically used. When using EtherCAT tpos (target position) may be more accurate, depending on the drive's lag. Lag is the time it takes for the drive to take a commanded position and make it current. Different drives have different averaging or smoothing algorithms as they calculate their next profile. For example 'fpos' will also lag more than 'tpos' since it will take the drive time to catch up to the commanded position. The benefit of tracking 'fpos' is when the master is in a high torque situation, with a larger position error, the slave will track the actual position rather than the desired position (tpos). Which to use will be application and drive dependent, with the preference towards 'fpos'.

Another option available on the M3-41 hardware module is the provision for directly connecting up to 3 local quadrature encoders. These encoders can be used as master references by the EtherCAT axis. To reference these encoders, the master_feedback variable is set to 1001, 1002, or 1003, for each of the respective encoder inputs. Once master_feedback is referencing a local encoder, its present value will appear in 'mpos' and 'mposc' MSB variables. In addition, all local encoder counts can be accessed using the 'ctr' array, index 5 to 7 from an MSB, or ctr5, ctr6, and ctr7 from QuickBuilder.

Master_feedback = 1001, ctr[5]/ctr5, P1 connector pins:

P1-19 A0+
P1-20 A0-
P1-21 B0-
P1-22 B0+

Master_feedback = 1002, ctr[6]/ctr6, P1 connector pins:

P1-23 A1+
P1-24 A1-
P1-25 B1-

P1-26 B1+

Master_feedback = 1003, ctr[7]/ctr7, P1 connector pins:

P1-27 A2+
P1-28 A2-
P1-29 B2-
P1-30 B2+

Power and ground for the encoder can be sourced from the +5V/GND connector on the model 5300 power supply. Additionally, when using 'set master target1', it is best to set the lower numbered axis as the master otherwise the position loop update will be delayed by one control loop cycle on the slave device. In most instances this delay is irrelevant.

Tracking operation can be modified with two variables, perrlimit and vcmd. During tracking operations the amount the slave axis master position has moved, each servo cycle, will be multiplied by 'vcmd'. By default this is 1.0 but may be modified to any positive value. A second variable 'perrlimit' can be used to dump the error when it reaches the value set in 'perrlimit'. This is useful in winding operations. Setting 'perrlimit' to 0 disables this feature and is the default.

Move on a Gear

One of the new features introduced on the M3-41 module is the ability to be geared to a master axis and then do a 'move' command (CSP mode only) which will transpose that move commands profile on top of that tracking the master axis. This is useful in a number of applications, particularly where the slave may be tracking with a master but needs to catch up to that axis and maintain gearing.

Specific property variables:

tracking_pstate – Read only, contains the current execution state of the drive during a 'move on a gear' operation. '*pstate'* must be in TRACKING mode for this property to be valid.

tracking_tpos – Read only, same as '*tpos'* property but used only for 'move on a gear' operation. It will depict the relative position reference with an initial position of 0. This is then added to the tracked position as calculated from the master position.

tracking_tposc = Read only, number of counts required for the 'move on a gear' move for this increment of its profile.

tracking_sign – Read only, 0 if not used, 1 for positive rotation, -1 for negative rotation when adding '*tracking_tposc*' to '*tposc'* to derive resulting motion.

tracking_status – Read only, contains the current state of the move as referenced by the 'QS2_Status' property.

```
NOT_INITIALIZED          = 0
STOPPED_READY            = 1
```

```
      ACCELERATING            = 3
      AT_MAX_SPEED            = 4
      DECEL_TO_NEW_MAX_SPEED  = 5
      DECEL_TO_STOP           = 6
```

The below example is two axes, where the second axis is the master:

MSB axis 2, Master Axis (can be virtual axis):

```
/**************** ENABLE DRIVE *******************/
delay 1000 ms;
 [Drive_Enable]
drive enable;                 // Enable the drive
set common bit 1 true;        // Tell slave drive is enabled and it can gear
wait common bit 0 true;       // Wait for slave to be geared

[loop]

move at 10 for 1000000;       // make a long move
wait for in position;
delay 1000;

goto loop;  // repeat
```

MSB axis 1, Slave Axis (start this axis first):

```
/**************** ENABLE DRIVE *******************/
// Clear common bits used to handshake with the master
set common bit 0 false;
set common bit 1 false;

[Drive_Enable]
drive enable;
wait common bit 1 true;       // Wait for master to tell us its drive is enabled
zero master counters;
master_feedback = 2;          // Reference axis 2 for fposc so we can track
set master feedback;
set mode tracking;            // Enable tracking mode

// Set our gear ratio
gear at 1:1;

set common bit 0 true;        // tell master we are all geared and ready to go

[reverse]
delay 3000 ms;                // Track 1:1 with master at speed of master
move at 5 for 50;             // Simulate catching up to Master.
                              // If Master at 10 rev/s then doing 15 rev/sec
                              // during move, ‘wait for in position’ not usable.
[stall]
if tracking_pstate != 2 goto stall; // wait for move to COMPLETE
goto reverse;
```

Only ‘move’ commands can be executed while in TRACKING mode. Slew and camming are not supported by the slave axis.

Axis property ‘pstate’ and ‘tracking_pstate’, where ‘pstate’ is equal to TRACKING mode for ‘move on a gear’ to occur, both can have the following values:

```
IDLE = 0,           // Ready to run new move command
RUNNING = 1,        // Processing sub-steps
COMPLETE = 2,       // Done running, awaiting IDLE to begin new move
STOP = 3,           // Stop
SLEWSTOP = 4,       // Slewed stop
SLEWING = 5,        // Slewing
PRESPLINE = 6,      // Pre 'SPLINE' move
PRECAM = 7,         // Pre 'CAM' move
PRECAM_REVERSE=8,   // Run cam table in reverse
CONT_CAM=9,         // Continue cam from where stopped (assumes did jog)
INSPLINE=10,        // In 'SPLINE' move
INCAM=11,           // In 'CAM' move
TABLESTOP=12,       // Stop table
TRACKING=13,        // Geared mode, state of ‘pstate’ when
                    //     ‘tracking_pstate’ valid
PRETRACKING=14,     // Initialization for TRACKING (geared) mode
EXIT_TRACKING=15,
ECAT_COMPLETE_PENDING=16,
ECAT_PROFILE_POS_INIT=17,
ECAT_PROFILE_POS_STARTING1=18,
ECAT_PROFILE_POS_STARTING1A=19,
ECAT_PROFILE_POS_STARTING2=20,
ECAT_PROFILE_POS_RUNNING=21,
ECAT_PROFILE_POS_WAIT_INPOS=22,
ECAT_PROFILE_VEL_INIT=23,
ECAT_PROFILE_VEL_WAIT_DELAY1=24,
ECAT_PROFILE_VEL_WAIT_DELAY2=25,
ECAT_PROFILE_VEL_WAIT_DELAY3=26,
ECAT_PROFILE_VEL_WAIT=27,
ECAT_PROFILE_TORQUE_INIT=28,
ECAT_PROFILE_INIT_CSP=29,
ECAT_PROFILE_INIT_INTERPOLATED=30,
ECAT_PROFILE_WAIT_CSP1=31,
ECAT_PROFILE_WAIT_CSP2=32,
ECAT_MODE_WAIT_CSP=33,
ECAT_MODE_WAIT_INTERPOLATED=34,
ECAT_MODE_WAIT_PROFILE=35, //Idling, awaiting Profile request to be processed.
ECAT_PROFILE_INIT_QSTOP=36,
ECAT_PROFILE_WAIT_QSTOP1=37,
ECAT_PROFILE_WAIT_QSTOP2=38,
ECAT_PROFILE_WAIT_QSTOP=39,
ECAT_PROFILE_AT_VEL=40,
ECAT_PROFILE_AT_TORQUE=41,
ECAT_HOMING_INIT=42,
ECAT_HOMING_STARTING1=43,
ECAT_HOMING_STARTING1A=44,
ECAT_HOMING_STARTING2=45,
ECAT_HOMING_RUNNING=46,
ECAT_HOMING_WAIT_INPOS_KOLLMORGEN=47,
ECAT_HOMING_WAIT_INPOS=48,
ECAT_HOMING_WAIT_INPOS_IAI=49,
ECAT_PROFILE_POS_WAIT_ABORTING=50,

ECAT_OFFLINE = -1
```

Camming Moves and Optional Timeouts

Camming Moves are supported in Cyclic Sync Position and Interpolated Position modes. You are allowed up to 2000 rows (master/slave entries) per table, with up to 6 tables available (numbered 0 to 5). Consult the *QuickMotion Reference Guide* for more detailed information. An MSB example is as below:

```
drive enable;
zero feedback position;
cmode = $CYCLIC_SYNC_POSITION_MODE;

// This MSB effectively sets up a 1:1 gear ratio with the Master
// The first item in the table is master revolutions/second.
// The second entry is that of the slave, this MSB.
table 1 clear;          // clear out the old data

table 1 addseries       // load new data to table 1
0.000 ,     0.0000:     //set up a 1:1 ratio
1.000 ,     1.0000:
2.000 ,     3.000 :
3.000 ,     5.0000:
5.000 ,     0.0000;     // CAMS wrap back to zero, like a mechanical cam

table 1 precompute;     // compute the cam (about 1/4 sec per 1000 pts)
// The master can reference another drive with the 'set master feedback1'
// command and setting the master_feedback variable to the drive desired.
// It can also reference its own created master, as shown below.
// This master is virtual and will increment by 1000 counts every
// Control Loop tick (1 ms).  This way only one drive is needed for
// testing camming.
set master virtual;           // We will use our own virtual master
// Set up the number of counts to increment per control loop.
move master at 1000 forever;
// This value will be divided by ppr for actual master revs.
zero feedback position;
[top]
// With the table repeat count set to 0 it will continually recycle
// through the table, forever, until a ‘stop table’ command is executed.
// This one will do 6 test cycles, repeating the cam table.
table 1 start linear cam 1.0, 1.0, 6;
wait for in position;   // Wait for CAM table to be done and exit

[loop]
activeCAM_row = 0;      // This contains where the table left off when it
                        // exited and must be zero'd as it
                        // is where it will start next time through.
delay 3000;             // Stall for now...
```

```
goto loop;
```

Below, a small section of the code is re-written to show how timeouts can be used in case a move does not occur becauseof an error condition. Timeouts can be used anywhere, not just in camming.

```
[top]
mytime = 10000;         // About 10 seconds
set timeout mytime;
on timeout goto timedout;  // We will expect the CAM move to finish in 10
                           // seconds or abort.
// With the table repeat count set to 0 it will continually recycle
// through the table, forever, until a 'stop table' command is executed.
// This one will do 2 test cycles, repeating the cam table.
table 1 start linear cam 1.0, 1.0, 2;
wait for in position;   // Wait for CAM table to be done or timeout
set timeout 0;          // Cancel the timer since we finished the move

[loop]
activeCAM_row = 0;      // This contains where the table left off when it
                        // exited and must be zero'd as it
                        // is where it will start next time through.
delay 3000;             // Stall for now...
goto loop;

[timedout]
// Timed out on camming table move
stop table;       // Stop the camming
[stall]
i = i+1;          // Increment i so know timeout worked from QuickView
goto stall;
```

Segmented Moves

Segmented Moves are supported in Cyclic Sync Position and Interpolated Position modes. You are allowed up to 20 segments. As shown in the example below, first clear the segment table, then add each segment, followed by a 'start' command. An MSB example:

```
// Enable the drive and clear our position
drive enable;
zero feedback position;
cmode = $CYCLIC_SYNC_POSITION_MODE;

// Initialize the move variables
vel1=5;
vel2=11;
rate1=50;
rate2=5;
dist1=10;
dist2=60;

stop_dist=1;

// Start the move loop
```

```
[top]
segmove 1 clear;        // Clear any prior segments stored

// Add each of the desired segments for the move desired
segmove 1 accdec to vel1 using rate1;
segmove 1 slew until dist1;
segmove 1 accdec to vel2 using rate2;
segmove 1 slew until dist2;
segmove 1 accdec to 0 for stop_dist;
// Start the move now
segmove 1 start relative;
// Wait for it to run the segments
wait for in position;
// Delay a bit and then do it again
delay 1000 ms;
goto top;
```

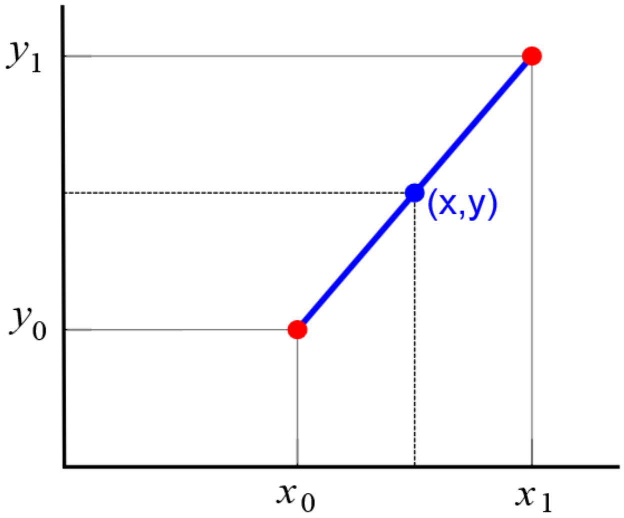
Slewed Move

A slewed move can be used for jogging as well as other desired motions. It is basically a move at some velocity that is reached in the time specified. An example of a slewed move:

```
drive enable;
zero feedback position;

cmode = $CYCLIC_SYNC_POSITION_MODE;
Speed = .5;             // Set our desired slew speed

[Slew]
slew begin;             // Enter slewing mode
slew at Speed in 1; // Slew to speed in 1 second
delay 5000 ms;          // Maintain speed for 5 seconds
slew at 0 in 0.1; // Slew to a stop in .1 seconds
slew end;               // Drop out of slewing mode
delay 1000 ms;          // Pause for 1 second
Speed = Speed * -1;     // Change directions and slew the other way

goto Slew;
```

Linear Interpolation (2D)

Two dimensional (2D) linear interpolation allows any two drives to arrive at the same point, at the same time. This is referred to as the target X, Y position, when programming with MSB's, with the move from the present position to the target being interpolated.



The controlling axis is the X axis MSB while the axis that will be controlled in unison is the Y axis. A number of properties are available for an interpolated move:

axisY – The axis number, from 1 to N, which will be the Y axis, commanded from the X axis. The Y axis must be set for interpolation to occur.
vectorY – The desired Y position on an X/Y grid in user units, based upon revolutions.
Note that this value is overwritten after a circular interpolated move for diagnostic purposes.

Read only variables for viewing in the debug watch window:

angle – Read only, calculated angle of the last vector move.
magnitude – Read only, calculated size of the last vector move.
velX – Read only, calculated velocity along the X axis of the last vector move.
velY – Read only, calculated velocity along the Y axis of the last vector move.
accX – Read only, calculated acceleration along the X axis of the last vector move.
accY – Read only, calculated acceleration along the Y axis of the last vector move.
decX – Read only, calculated deceleration along the X axis of the last vector move.
decY – Read only, calculated deceleration along the Y axis of the last vector move.

Any of the normal 'move' commands can be used. The velocity and acceleration parameters are that of the vector while the position information is for the X axis position. The Y axis MSB should not attempt to control the axis while the X axis has it in motion or an error will result. Common bits can be used to synchronize the tasks if needed. Additionally, both drives must be in CSP mode.

Sample Program:

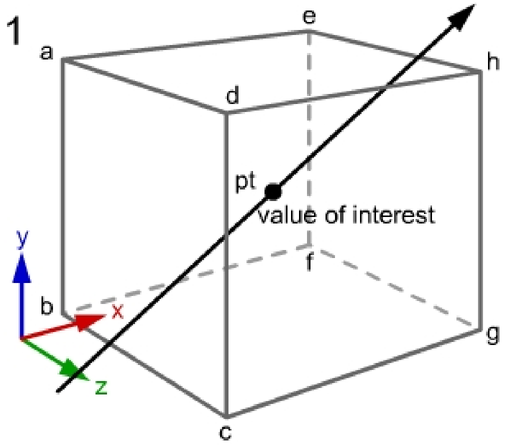
```
[beginTest]
axisY = 2;         // set the Y axis
delay 1000 ms;     // make sure the axis is running first
speed = 40;        // Set the vector velocity in rev/sec

[interpolate]
```

```
// For simplicity user units are 1:1 with revolutions (uun/uud)
cmode = $CYCLIC_SYNC_POSITION_MODE; // CSP mode
vectorY = 10;     // Y position of 10 revolutions
// Below will make a 45 degree angle, 100 is the vector accel/decel
move at speed for 10 using 100,100;  // vector move where 'for' is the
                                     // X position.
wait for in position;   // Wait for move to finish on both X & Y axis

// move along the X Axis only
vectorY = 0;
move at speed for -10 using 100, 100;
wait for in position;

// Move along the Y Axis only, should be back to start after the move.
vectorY = -10;
move at speed for 0 using 100, 100;
wait for in position;

// should be back home now, do again
goto interpolate;
```

Splines may be used to with multiple X/Y vectors to provide smooth motion through multiple 2D/3D positions residing in a table. Linear, cubic, and quadratic splines are supported.

Linear Interpolation (3D)

Three dimensional (3D) linear interpolation allows any three drives to arrive at the same point, at the same time. This is referred to as the target X, Y, and Z position, when programming with MSB's, with the move from the present position to the target being interpolated.



As with 2D linear interpolation, the controlling axis is the X axis MSB while the axes that will be controlled in unison are the Y and Z axis. A number of properties are available for an interpolated move:

axisY – The axis number, from 1 to N, which will be the Y axis, commanded from the X axis. The Y axis must be set for either 2D or 3D interpolation to occur.

vectorY – The desired Y position on an X/Y/Z grid in user units, based upon revolutions. Note that this value is overwritten after a circular interpolated move for diagnostic purposes.
axisZ – The axis number, from 1 to N, which will be the Z axis, commanded from the X axis.
vectorZ – The desired Z position on an X/Y/Z grid in user units, based upon revolutions. Note that this value is overwritten after a circular interpolated move for diagnostic purposes.

Read only variables for viewing in the debug watch window:

angle – Read only, calculated angle of the last vector move.
magnitude – Read only, calculated size of the last vector move.
velX – Read only, calculated velocity along the X axis of the last vector move.
velY – Read only, calculated velocity along the Y axis of the last vector move.
velZ – Read only, calculated velocity along the Y axis of the last vector move.
accX – Read only, calculated acceleration along the X axis of the last vector move.
accY – Read only, calculated acceleration along the Y axis of the last vector move.
accZ – Read only, calculated acceleration along the Z axis of the last vector move.
decX – Read only, calculated deceleration along the X axis of the last vector move.
decY – Read only, calculated deceleration along the Y axis of the last vector move.
decZ – Read only, calculated deceleration along the Z axis of the last vector move.

Any of the normal ‘move’ commands can be used. The velocity and acceleration parameters are that of the vector while the position information is for the X axis position. The Y/Z axis MSB’s should not attempt to control their axis while the X axis has it in motion or an error will result. Common bits can be used to synchronize the tasks if needed. Additionally, all 3 drives must be in CSP mode.

Sample Program:

```
[beginTest]
axisY = 2;        // set the Y axis, required for interpolation
axisZ = 3;        // set the Z axis
delay 1000 ms;    // make sure the axis is running first
speed = 40;       // Set the vector velocity in rev/sec

[interpolate]
// For simplicity user units are 1:1 with revolutions (uun/uud)
cmode = $CYCLIC_SYNC_POSITION_MODE; // CSP mode
vectorY = 10;     // Y position of 10 revolutions
vectorZ = 20;     // Z position of 20 revolutions
// Below will make a 45 degree X/Y angle, 100 is the vector accel/decel
move at speed for 10 using 100,100;  // vector move where 'for' is the
                                     // X position.
wait for in position;   // Wait for move to finish on X, Y & Z axis

// move along the X Axis only
vectorY = 0;
vectorZ = 0;
move at speed for -10 using 100, 100;
wait for in position;

// Move along the Y Axis only, should be back to start after the move.
vectorY = -10;
```

```
        vectorZ = 0;
        move at speed for 0 using 100, 100;
        wait for in position;

        // Move along the Z Axis only, should be back to start after the move.
        vectorY = 0;
        vectorZ = -20;
        move at speed for 0 using 100, 100;
        wait for in position;

        // should be back home now, do again
        goto interpolate;
```

When running 3D linear interpolation all the vector parameters are based upon the 2D X/Y axis calculations. The Z axis will attempt the same type of move based upon the required distance and calculated time to target, ignoring the vector motion parameters. Since both X and Y axis arrives at the target at the same time, moving the Z axis in this way ensures it arrives simultaneously.

Circular Interpolation (2D)

Two dimensional (2D) circular interpolation allows any two drives to work in coordinated motion to draw an arc or full circle. This functions similarly to that of 2D linear interpolation but with the addition of a few parameters to define the arc.

The controlling axis is the X axis MSB while the axis that will be controlled in unison is the Y axis. A number of properties are available for an interpolated move:

axisY – The axis number, from 1 to N, which will be the Y axis, commanded from the X axis. The Y axis must be set for interpolation to occur.
radius – The radius in user units of the arc to be drawn. A negative radius flips the arc.
angleSweep – The desired amount of angular motion that is to occur relative to the radius center point. A positive angleSweep rotates clockwise, negative, counter clockwise.
angleStart – The angle at which motion should start where 0 is vertical on the Y axis, minus angle moves left, and positive angle moves right. The 'angleSweep' variable is added to this angle.
vectorZ – The calculated center of the arc for the X axis will be stored here for diagnostic reference, in machine units.
vectorY – The calculated center of the arc for the Y axis will be stored here for diagnostic reference, in machine units. Make sure you update vectorY after a circular move if the next move is linear interpolation.

Read only variables for viewing in the debug watch window:

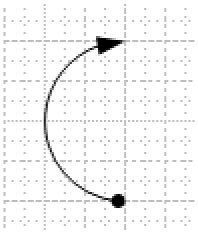
angle – Read only, initialized to 0 and records the calculated angle as it sweeps.
velVector – Read only, velocity in radians/second that is being used for the calculated profile.
accVector – Read only, acceleration in radians/second^2 that is being used for the calculated profile.
decVector – Read only, deceleration in radians/second^2 that is being used for the calculated profile.

Any of the normal 'move' commands can be used. The velocity and acceleration parameters are in user units and represent the feed rate at a point on the arc. These parameters are then converted into radian units. The Y axis MSB should not attempt to control the axis while the X axis has it in motion or an error will result. Common bits can be used to synchronize the tasks if needed. Additionally, both drives must be in CSP mode.

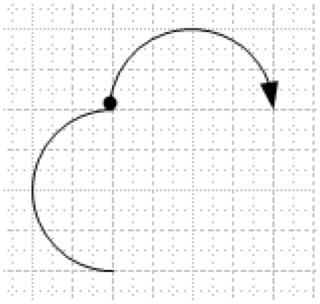
Below are two sets of arcs drawn, one specifying the feed rate and the second time based, with all the needed velocity, acceleration, and deceleration dynamically calculated:

Example absolute moves:

```
[beginTest]

axisY = 2;  // set that we will coordinate with axis 2

[top]

// CW rotation
radius = .5;              // 1/2 inch radius, 5 rev/inch., uud = 5.
angleSweep = 180; // 180 degree sweep
angleStart = -90; // negative 90 degree offset on Y axis
// move at 1.5 inches/sec feed rate with accel/decel of 4 inches/sec2
// velocity / radius = rad/sec
move at 1.5 to 0 using 4,4;
wait for in position;
```

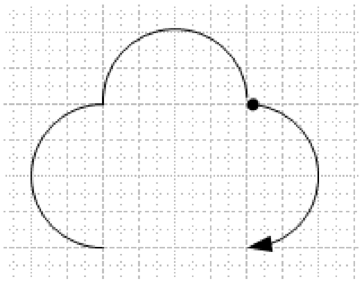


```
// CW rotation
radius = .5;              // 1/2 inch radius.
angleSweep = 180;
angleStart = 0;           // no offset on Y axis
// move at 1.5 inches/sec feed rate with accel/decel of 4 inches/sec2
// velocity / radius = rad/sec
move at 1.5 to 0 using 4,4;
wait for in position;
```

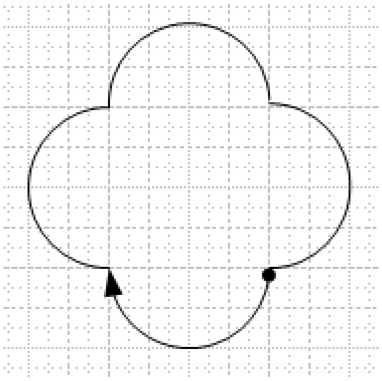


```
// CW rotation
radius = .5;              // 1/2 inch radius.
angleSweep = 180;
angleStart = 90;          // 90 degree offset on radius
```

```
// move at 1.5 inches/sec feed rate with accel/decel of 4 inches/sec2
// velocity / radius = rad/sec
move at 1.5 to 0 using 4,4;
wait for in position;
```



```
// CW rotation & flip
radius = -.5;            // 1/2 inch radius but flip the curve, CW.
angleSweep = 180;
angleStart = 0;          // no offset to the radius
// move at 1.5 inches/sec feed rate with accel/decel of 4 inches/sec2
// velocity / radius = rad/sec
move at 1.5 to 0 using 4,4;
wait for in position;
```

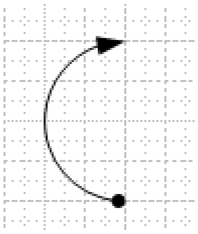


```
[stall]
delay 4000 ms;
// Do it again, forever...
goto top;
```

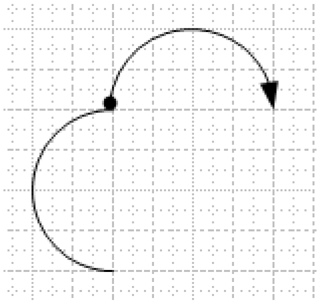
Example time based moves:

```
[beginTest]

axisY = 2;  // set that we will coordinate with axis 2

[top]

// CW rotation
radius = .5;             // 1/2 inch radius, 5 rev/inch., uud = 5.
angleSweep = 180; // 180 degree sweep
angleStart = -90; // negative 90 degree offset on Y axis
// draw arc in 1 second and calculate required velocity and acceleration
move in 1 to 0;
```

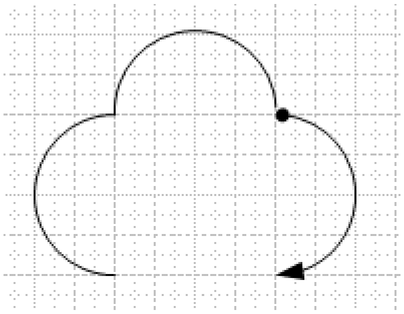
```
wait for in position;
```



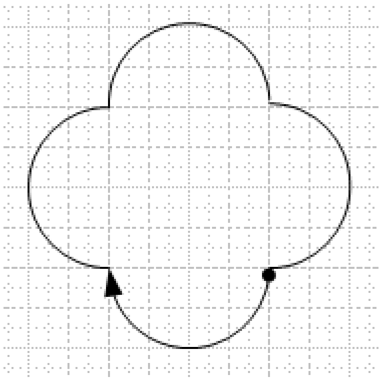
```
// CW rotation
radius = .5;            // 1/2 inch radius.
angleSweep = 180;
angleStart = 0;         // no offset on Y axis
// draw arc in 1 second and calculate required velocity and acceleration
move in 1 to 0;
wait for in position;
```



```
// CW rotation
radius = .5;            // 1/2 inch radius.
angleSweep = 180;
angleStart = 90;        // 90 degree offset on radius
// draw arc in 1 second and calculate required velocity and acceleration
move in 1 to 0;
wait for in position;
```



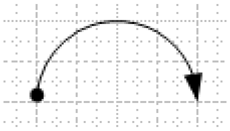
```
// CW rotation & flip
radius = -.5;           // 1/2 inch radius but flip the curve, CW.
angleSweep = 180;
angleStart = 0;         // no offset to the radius
// draw arc in 1 second and calculate required velocity and acceleration
move in 1 to 0;
wait for in position;
```



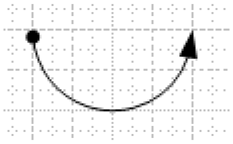
```
[stall]
delay 4000 ms;
// Do it again, forever...
goto top;
```

Arc summary with positive/negative radius and sweep angles:

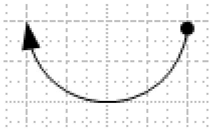
Radius = .5, angleSweep = 180, angleStart = 0



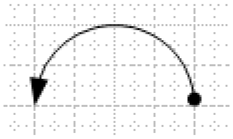
Radius = .5, angleSweep = -180, angleStart = 0



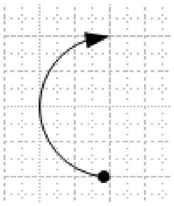
Radius = -.5, angleSweep = 180, angleStart = 0



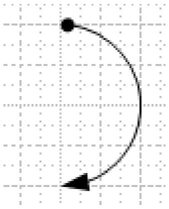
Radius = -.5, angleSweep = -180, angleStart = 0



Radius = .5, angleSweep = 180, angleStart = -90



Radius = .5, angleSweep = 180, angleStart = 90



Camming & Splines (2D/3D) – Linear, Cubic & Quadratic

Camming tables in QuickMotion are two-dimensional arrays of floating-point data. There are 8 tables available for use with the M3-41, numbered 0 through 7, each having up to 2000 rows and always 2 columns. These columns are named “x” and “y”. Although their primary use is to hold data for *spline*- and *CAM*-based motion, they can be used to hold arbitrary data such as positions for recipe-based motion. Although limited to 8 tables per axis, these tables can also be swapped out dynamically and refreshed with new data when loaded from the controller file system.

Spline tables use the “x” column as time and the “y” column as a *relative* position. *CAM* tables use the “x” column as a *relative master* position and the “y” column as a *relative slave* position.

Since *spline* and *CAM* tables use *relative* position data, the first point pair in these tables must be 0.0, 0.0 (time/master-position of 0, position/slave-position of 0). The exception to this is with CAM tables where the y component can be non-zero, thereby establishing an offset. In addition, for any tables used for *spline* and *CAM* operations, all “x” values must be increasing, that is: a given row’s “x” must be greater than the previous row’s “x”. Also, the minimum number of rows (pairs) in these tables is 3.

It is recommended that CAM tables and instructions be used whenever possible. Significant enhancements have been made to camming which have currently not been carried forward to splines. Some of this consists of the ability to start on non-zero y column values, ability to start anywhere within a table, and forward and reverse table traversing.

Points in a *spline* or *CAM* table are also referred to as *knots*, as they represent critical loci that must be passed through when interpolation occurs.

For example, in the following *spline* table:

0.0	0.0
1.5	2.0

2.0	2.5
3.0	3.0
4.0	2.0
5.0	0.0

There are 6 knots. Since this is a *spline* table, the last 5 knots are interpreted as follows:

- At time = 1.5 seconds, the position of the axis should be 2.0 user-units beyond where the axis started this spline move.
- At time = 2.0 seconds, the position of the axis should be 2.5 user-units beyond where the axis started this spline move.
- At time = 3.0 seconds, the position of the axis should be 3.0 user-units beyond where the axis started this spline move.
- At time = 4.0 seconds, the position of the axis should be 2.0 user-units beyond where the axis started this spline move.
- At time = 5.0 seconds, the position of the axis should be back where the axis started this spline move.

The position of the axis between these "knots" is determined by the interpolation method specified by the MSB code when the table is *started.*

The three available interpolation methods in QM for *spline* (and *CAM* tables) are:

Linear - A straight-line joins each knot.

Quadratic - A piecewise 2nd degree polynomial is fitted between this knot and the next; the first derivative of the first point is forced to 0.

Cubic - A piecewise 3rd degree polynomial is fitted between this knot and the next two knots; the first and second derivatives of the first point is forced to 0.

Splines and Camming work identical to the M3-40 module. The M3-41 adds the ability to synchronize multiple axes, up to 3, using splines and camming tables. As with 2D & 3D Linear Interpolation, the axisY and axisZ properties may be assigned to the desired axis with which to synchronize the controlling X axis.

axisY – The axis number, from 1 to N, which will be the Y axis, commanded from the X axis. The Y axis must be set for either 2D or 3D interpolation to occur.

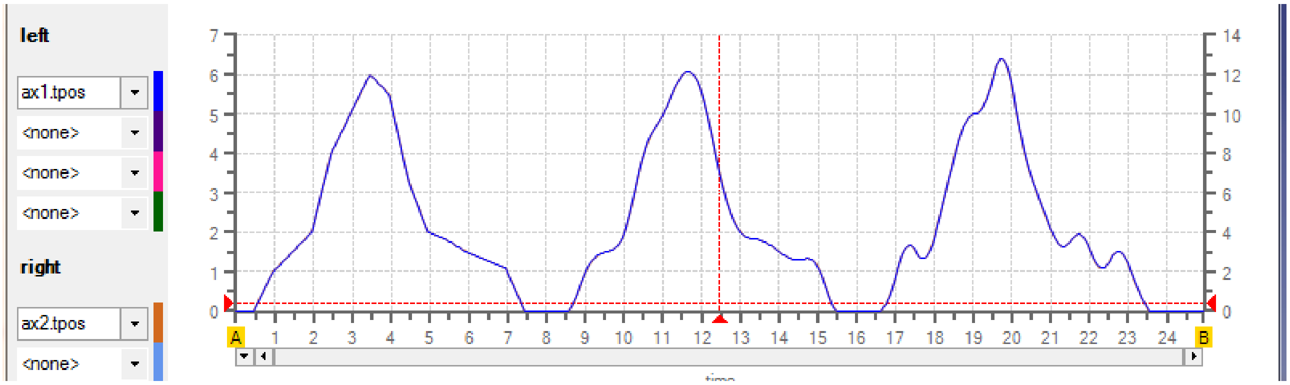
axisZ – The axis number, from 1 to N, which will be the Z axis, commanded from the X axis in 3D operations.

In order to operate in 2D mode the first table has its data assigned, for 3D, the first table # +1 is used for the Y axis, first table #+2 for the Z axis. The Z axis cannot be used without the Y axis. When the table pre-compute is executed on the first table, it will automatically computer the other tables based upon the axisY and axisZ contents. The same is true for the 'table start', start the first table and the other axis will start simultaneously. Note that there must be the exact same number of entries in all tables, using the same time references for splines, or master position for camming. Also all axis must be in Cyclic Synchronous Position (CSP) mode. The following is an example of a 2D spline operation:

```
[beginTest]
// Demonstrates the three types of splines:  linear, cubic,
// and quadratic

// Set the Axis to use for Y, this can be done anytime
// prior to the precompute.
axisY = 2;

// Clear the X and Y axis
table 0 clear;
table 1 clear;

// Add the spline data points for the X axis.
// First is time, send if position.
table 0 addseries
0.000 ,     0.0000      :                 // simple spline table
0.500 ,     1.0000      :
1.000 ,     1.5000      :
1.500 ,     2.0000      :
2.000 ,     4.0000      :
2.500 ,     5.0000      :
3.000 ,     6.0000      :
3.500 ,     5.5000      :
4.000 ,     3.3000      :
4.500 ,     2.0000      :
5.000 ,     1.8000      :
5.500 ,     1.5000      :
6.000 ,     1.3000      :
6.500 ,     1.1000      :
7.000 ,     0.000 ;

// The Y table will be X table number + 1
// Now enter the spline data points for the Y axis.
// If Z was used it would be table #3, there
// may be up to 6 tables, 0 to 5.
table 1 addseries
0.000 ,     0.0000      :                 // simple spline table
0.500 ,     2.0000      :
1.000 ,     3.0000      :
1.500 ,     4.0000      :
2.000 ,     8.0000      :
2.500 ,     10.0000     :
3.000 ,     12.0000     :
3.500 ,     11.000      :
4.000 ,     6.6000      :
4.500 ,     4.0000      :
5.000 ,     3.6000      :
5.500 ,     3.0000      :
6.000 ,     2.6000      :
6.500 ,     2.2000      :
7.000 ,     0.0000      ;
```

```
// Now calculate the move, note that since
// axisY is set the table for X and X#+1
// will be done by this single instruction.
// If axisZ was set then table X#+2 would
// also be done.
table 0 precompute;          // precomputes 1 & 2

[top]
// zero our position since table is 0 based
zero feedback position;

// Do the X & Y linear move
table 0 start linear  1.0, 1.0, 1;
wait for in position;
delay 1000;
// Do the X & Y cubic move
table 0 start cubic   1.0, 1.0, 1;
wait for in position;
delay 1000;
// Do the X & Y quadratic move
table 0 start quadratic   1.0, 1.0, 1;
wait for in position;

// pause and do again...
delay 3000;
goto top;
```

The resulting target positions calculated for each move type, on both axes, are shown below. The first is a linear move, second cubic, and third quadratic, note that both axis are exactly synced and overlaying one another:



Torque Control

Some applications, such as a press, may want to dynamically control the maximum torque applied. Yaskawa, Sanyo Denki, and Emerson drives support a variable called 'tmax' which when set represents the % of maximum torque. At power up tmax is initialized to whatever the drive default is, for example Emerson is 175% or 'tmax' = 175. The variable 'rmstrq' represents the current torque requirements in %, maximum being that set by 'tmax'. Note that the property sheet of Quickbuilder is ignored and the actual default powerup/reset value from the drive is read and written to 'tmax'.

If you exceed the 'tmax' value and stall the drive, motion will stop and 'inpos' will never be set due to the failure of 'fpos' (feedback position) to equal 'tpos' (target position). MSB programs must not use the 'wait for in position' command or they will hang forever. Either the 'pstate' (COMPLETE, 2) or 'QS2_status' (STOPPED_READY, 1) may be monitored for the moves completion.

EtherCAT drive specific object access is as follows:

Emerson

tmax – 0x2004.7 / 10.0 (Symmetrical Current Limit PR4.07)
Initialized to value in 0x2004.7 or 175.0 if not available.
rmstrq – 0x2004.4 / 10.0 (Current Demand PR4.04)

Sanyo Denki

tmax – 60E0.0 / 10.0 (positive torque %) & 60E1.0 / 10.0 (negative torque %), 0 to 500%.
Initialized to 500.0 during initialization.
rmstrq – 0x6077.0 / 10.0 (Actual torque)

Yaskawa

tmax – 0x6072.0 / 10.0 0 (Max torque)
Initialized to value in 0x6072.0 or 300.0 if not available.
rmstrq – 0x6077.0 / 10.0 0 (Actual torque)

LinMOT

tmax – 0x13A6.0 / 10.0 (Max Current A) & 0x13BA.0 / 10.0 (Max Current B)
Limit is expressed in Amps, not %. Initialized to value in 0x13A6.0 or 4.0 if not available.
rmstrq – 0x1b93.0 / 1000.0 (Current Demand /1000.0 = Amps)

Mitsubishi

tmax – 60E0.0 / 10.0 (positive torque %) & 60E1.0 / 10.0 (negative torque %), 0 to 100%
Initialized to 100.0 during initialization, not read from the drive.
rmstrq – 0x6077.0 (Actual torque)

IAI

tmax – Initialized to value from properties of drive in QuickBuilder.
Pressing Current Limit set to (tmax/100.0) * 255.0
Load Current Threshold set to (tlim/100.0) * 255.0
tlim initialized to 0.

Below drives do not support the 'tmax' function but do provide 'rmstrq'.

Kollmorgon

tmax – not supported
rmstrq – 0x6077.0 (Actual torque)

Advance Motion Controls

tmax – not supported
rmstrq – 0x6077.0 (Actual torque)

Copley

tmax – not supported
rmstrq – 0x6077.0 (Actual torque)

Elmo

tmax – not supported
rmstrq – 0x6077.0 (Actual torque)

'tmax' must be set after 'drive enable' in order to take effect. Prior to 'drive enable' the current value in the drive is read and 'tmax' is initialized to that value or the constants detailed above for each drive. This allows the offline PDO scan to occur without changing the existing value in the drive.

Restarting EtherCAT Programmatically

A request to restart the EtherCAT network can be made programmatically by setting the number of the network module you wish to access (1 = first) in register 12333, followed by writing a 21930 (0x55AA) to register 13464. The network will be taken off line, rescanned, and MSBs restarted. Monitor the online status (13464) prior to access. Note that the I/O count will not be updated in the controller; a restart to the network should not be used to add or remove devices without cycling power on the controller.

12333 – (R/W): Network module bank select register; each EtherCAT and other supported network modules are selectable for further access. '1' is the first module.

13464 – (R/W): Network online status; 1 = online, 0 = offline. Writing a 21930 (0x55AA) to this register causes the network to reset and, in the case of EtherCAT, rescan all I/O and drives. MSBs will be restarted.

The network can also be restarted via telnet and the QuickBuilder EtherCAT Explorer. When restarting the network, any axis properties that were changed programmatically from those shown on the property sheet should be manually initialized. The restart does not re-initialize the property values; those are only set during a hardware reset or power cycle.

Special Register Access

12333 – (R/W): Network module bank select register; each EtherCAT and other supported network modules are selectable for further access. '1' is the first module.

13464 – (R/W): Network online status; 1 = online, 0 = offline. Writing a 21930 (0x55AA) to this register will cause the network to reset and in the case of EtherCAT, rescan all I/O and drives. MSBs will be restarted.

13025 – (RO): Number of digital inputs in the system

13026 – (RO): Number of digital outputs in the system

13027 – (RO): Number of analog inputs in the system

13028 – (RO): Number of analog outputs in the system

13029 – (RO): Number of motors (axes) in the system

Registers 13025 to 13029 are useful for the application to verify the number of I/O it expects to find in the system. Some EtherCAT Masters use a specially configured XML file to simplify I/O count verification. It is up to the application to confirm the correct number of I/O and axes during startup and notify the user of any discrepancies.

Global MSB Registers and Flags

There are 48 double precision user variables per MSB. These variables are local to an MSB and not available to the MSBs executing on a different axis. Global registers, which are local to each EtherCAT Master module, can be used as a shared variable resource amongst the axes. There are 32 registers named ‘global_reg1’ to ‘global_reg32’. Storage operations consisting of arithmetic operations are atomic to each MSB and these registers can be accessed from QuickBuilder just like axis properties. Like user variables, global registers are double precision variables.

Global flag registers are also available, operating the same as global registers. There are 5 flags registers named ‘global_flags1’ to ‘global_flags5’. Unlike variables the flag registers are 32 bit integers. Atomic math operations like ‘|’, ‘&’, and arithmetic bit shifting (<<, and >>) may be used. For example:

```
global_flags1 = global_flags1 << 1;   // shift bits left by 1.
```

QuickBuilder access of global registers and flags are atomic for read operations but not write. If atomic operation is needed it is recommended common bits be used as semaphore flags and is application dependent.

Accessing Properties of another Axis

Properties such as fpos, tpos, etc., are local to an axis and not shared with other axis MSBs. This limitation can be overridden by using the ‘axisptr’ property of an MSB. This property controls what axis the MSB will retrieve its property value on a read and write operation. It is typically set to the value of ‘axisnum’, which is the axis number of that axis executing the MSB. Setting this axis number to any other value will override what axis the property is retrieved from. An example follows:

```
axisptr = 3;        // Monitor axis 3 until the drive is enabled and running
[stall]
if enabled > 0 goto online;
goto stall;         // Wait until available
[online]
axisptr = axisnum;        // Future property access will be for out axis.
```

Any property value can be accessed but make sure you do not attempt to mix property access of more than one axis in a statement since all properties will reference that set by the axisptr. If you wish to read, or write, a value a user variable should be used. For example the following reads fpos on axis 3:

```
axisptr = 3;        // Access axis 3 properties
myvar = fpos;       // read fpos from axis 3
axisptr = axisnum;  // set property access back to our axis
```

Drive Type & Axis Number

The type of drive and axis number that an MSB is executing can be referenced programmatically via the ‘eCAT_driveType’ and ‘axisnum’ property variables. ‘axisnum’ contains the axis number, where 1 is the first. ‘eCAT_driveType’ is defined as follows:

$DRIVE_COPLEY	2
$DRIVE_YASKAWA	3
$DRIVE_ELMO	4
$DRIVE_KOLLMORGEN	5
$DRIVE_SANYO_DENKI	6
$DRIVE_EMERSON	7
$DRIVE_AMC	8
$DRIVE_VIRTUAL	9
$DRIVE_IAI_ACON_MODE3	11
$DRIVE_ABB_MICROFLEX	12

As with ‘cmode’, either the constant name beginning with ‘$’ or the actual numeric value may be used in an expression.

Example:

```
if eCAT_driveType == $DRIVE_AMC goto AMC;
if eCAT_driveType == $DRIVE_VIRTUAL goto Virtual;
```

Drive Object Access (SDO)

An MSB program can directly access any remote drive object, assuming the drive is in the correct state for the write operation. This is implemented by allowing SDO (Service Data Object) reads and writes. The command syntax is as follows:

```
sdo write <value>, <slave>, <object #>, <object index>, <object size, 1 to 4>;
sdo read <result storage>, <slave>, <object #>, <object index>, <object size, 1 to 4>;
```

If an error occurs, the MSB will enter a fault state. The Object size is the size of the data, in bytes, as specified by the manufacturer. The <slave> is the destination of the read or write: -1 for the current slave the MSB is operating on, or the slave index, starting at 1, as it appears in the discovery tree.

Example:

```
// 0x609a.00 is the homing acceleration object.  This is not used
// on Yaskawa so we can use it for general storage
sdo write counter, -1, 0x609a, 1, 4;
// Read the object value back, should be the same...
sdo read counter, -1, 0x609a, 1, 4;
```

Many objects cannot be accessed while the drive is in the operational state, resulting in a state error message and thus a fault. Also Virtual Drives support intermittent sdo read/writes to a specific slave, not to -1 since it is not actually online.

⚠ Although SDO access to slaves is supported it is suggested it be used sparingly as it limits the bandwidth available to the cyclic PDO transmissions. It may also result in warning messages of PDO re-transmission due to timeouts caused by SDO messaging. This is especially true when using a 500 µS EtherCAT control loop time.

CHAPTER

5

[5] Registration, Absolute Positioning, & Distributed Clock

Registration

Registration allows the present servo position to be captured when an input is triggered. The standard MSB commands can be used for registration as on the M3-40 card. Registration is only supported on certain drives: Kollmorgen, Sanyo Denki, and Yaskawa. These drives support the Touch Probe Function object, 0x60B8, as well as the Touch Probe Status, 0x60B9, and Touch Probe Value 1, 0x60BA. The Sanyo Denki and Yaskawa both support Touch Probe Value 2, 0x60BC for an additional registration input.

The QuickMotion language isolates the programmer from the interaction with these objects making registration simple to implement. The ‘set capture’ command selects the drive input probe 1 or 2. ‘set capwin’ sets the window within which the capture is allowed to occur, the reference for comparison, as well as optionally arming the touch probe input. The MSB instructions are summarized below:

syntax

set capture *transition* **of input** *input*

parameters

transition	*rise*, ~~*fall* **or** *edge* (any)~~ (drive dependent but only rise supported)
input	drive touch probe 1 or 2, reference specific drive for proper input wiring

This statement initializes the parameters to be used for all captures on this axis, specifying the input (*capInput*) to use. The following variables are computed and available after a successful capture:

capposc	capture position in encoder counts
cappos	capture position in user units
capTriggered	flag set to 1 when capture occurs

Note: *capposc* and *cappos* are only valid when *capTriggered* is a 1. Once armed *capposc*/*cappos* will reflect the value latched when the capture input goes active but is not necessarily within the defined capture window. *capTriggered* verifies the capture window against the latched *capposc*/*cappos*, prior to setting. If more than one running MSB on an M3-41 module arms the *same* input for capture, unexpected capture results may occur.

Only one input may be armed for capture at a time *per axis*. If another input is presently armed when this command is issued, the other input is effectively *disarmed*

syntax

```
set capwin range start, end using reference { arm }
```

parameters

`start` Start window position to compare against *reference*. *Reference* >= *start*.

`end` End window position to compare against *reference*. If equals *start* then no window exists and capture will occur based on input. *Reference* <= *end*.

`reference` the encoder count scaled reference variable to compare to:

fposc	feedback position
mposc1 - **mposc5**	master position counters #1 through #5
mposc	master position counter
smodc	slave position (modulo)
smark	slave marked position
tmc1 tmc2	temporary master counters #1 & #2
tsc1 tsc2	temporary slave counters #1 & #2
sdc	slave decrement counter
fposc1	feedback position of axis 1 (fposcA)
fposc2	feedback position of axis 2 (fposcB)
tmodc	Temporary master counter mod mmc
sfposc	Secondary feedback position of axis
tposc	Target position of axis
ctr0	din1 mapped input counter
ctr1	din2 mapped input counter
ctr2	din3 mapped input counter
ctr3	din4 mapped input counter
ctr4	din5 mapped input counter
ctr5	Local quadrature encoder 1
ctr6	Local quadrature encoder 2
ctr7	Local quadrature encoder 3

`arm` If included will arm the capture, if not arm will need to be done by a Wait or On command.

This statement initializes a window to be monitored for valid captures to occur, anything outside this window is considered invalid and ignored. If the capture occurs outside this window it will automatically be re-armed. If ‘arm’ is specified this statement will automatically arm the capture prior to completing this instruction. The *capwinStart* variable is the start of range and the *capwinEnd* variable is the end of range, inclusive. The ‘*capMod’* variable is used to perform a modulus on the reference value prior to comparison to the *start/end*, the remainder after the modulus is the value compared. This helps in situations that may experience rollover.

Below is an example program to latch the position whenever Touch Probe 1 occurs during a set of moves:

```
[beginTest]
// Setup registration
// First set which probe to use.
// Yaskawa & Sanyo Denki support Touch Probe 1 & 2.
// Kollmorgen supports only Touch Probe 1.
// Yaskawa fires on input being active, since active low this is falling edge
set capture rise of input 1;  // Select probe 1 (enter 2 for probe 2)
// Clear the window so capture will happen moment probe occurs
set capwin range 0, 0 using fposc arm;    // Arm at same time
// capTriggered will be set to a 1 when the capture occurs.

[run]
// Begin the move, 1 rev/second for 2 revolutions
if capTriggered != 1 goto notTriggered;
// Clear the window so capture will happen moment probe occurs
set capwin range 0, 0 using fposc arm;    // Clear the range, arm at same time
                                          // 'wait capture' will arm as well.
[notTriggered]
move at 1 for 2;
wait for in position;
// Delay 1 second once in position
delay 3000 ms;
// Do a relative move back 2 revolutions at 1 rev/second
move at 1 for -2;
wait for in position;
// Delay 1 second once in position
delay 3000 ms;
// Do it again, forever...
goto run;
```

‘touchProbeStatus’ MSB variable maps directly to object 0x60b9.

Absolute versus Incremental Positioning Modes

By default the M3-41 uses Incremental Positioning mode. At power-up it records the current absolute position and zeros it; thereby fpos and tpos are at 0. If the encoder is battery backed, you can use Absolute Positioning mode to maintain position. This mode sets fpos and tpos to the current actual position (0x6064). You can later clear it to 0 by using the ‘zero feedback position’ command or do moves based on an offset from tpos. Regardless, Absolute Positioning mode allows you to maintain position after a power cycle.

Enable Absolute Positioning mode by setting the encoder_mode axis property variable to a 1, prior to executing the 'drive enable' command. A value of 0 is for an incremental encoder. This variable can also be automatically set by using the axis property pull-down menu option: absolute.

For example, with Yaskawa and a circular table, this is done by using SigmaWin+. Perform this operation with the following sequence:

1. Make sure the drive is set for Absolute Encoder (Pn002.2). If not, set it, exit the program, and re-enter (needed to enable menu functions).
2. Set the Servopack for Multi-turn (Pn205), setting the number of revolutions of the motor to one revolution of the circular table. Write to the drive and cycle power (note that a setting of zero is 1:1).
3. Setup->Multiturn Limit Setup is invoked. The number must match the number previously set as this sets it in the motor; cycle power.
4. Setup->Reset Absolute Encoder, cycle power. This establishes a position within the Multi-turn window. Example: If the Multi-turn value (Pn205) is set to '0', resetting the absolute encoder places the motor's position somewhere between '0' and '1048576' counts; cycle power.
5. Setup->Search Origin is done to locate the actual rotation reference position of the motor. The function allows you to jog the axis into the encoder's marker position, also called the "Point of Origin."

Using the Multi-turn functionality of the Yaskawa drive ensures that whenever the power is cycled, the actual position (0x6064) will be based on one rotation of the circular table. While running, the actual position (0x6064) will increment/decrement normally as a 32-bit number.

If you wish to zero the home position, the Absolute Encoder Home Offset object (0x607C) can be set. The value in this object is added to the Absolute Position and is the value placed in the actual position (0x6064) object, thus zeroing position. In most drives this is the 0x607C object. The value written is the complement of the position when at home, upon power-up. This causes fpos and the actual position (0x6064) object to appear to be 0 at home. For Yaskawa drives, use the SigmaWin+ utility to initialize 0x607C, since it is a non-volatile object that must be set prior to the drive being operational.

Distributed Clock & DC Sync

By default the distributed clock is always read from the first DC slave and distributed to all the slaves in the system using the EtherCAT ARMW command. The first slave in the EtherCAT cabling that supports it will be assigned the role of distributed clock master and the M3-41 will read the time from this slave on every control loop cycle and write it to all following slaves on the network.

Both 32 & 64 bit reference clocks are supported with the release of M3-41 V1.46 and above. Earlier revisions only supported 64 bit.

Currently 64-bit clocks are available in the following supported devices:

- Beckhoff EK1100 couplers
- Wago 750-354 couplers
- Omron GX-JC06 EtherCAT Junction Slaves
- Elmo Gold
- Sanyo Denki drives.
- ABB e150 Drives
- SMC Corp not supported, must not be first slave device.

Currently 32-bit clocks are available in the following supported devices:

- Yaskawa Sigma 5
- Copley Accelnet
- SMC Corp not supported, must not be first slave device.

With the Beckhoff EK1100, no other modules are needed, just the coupler as the first node. Wago requires an IO module as well as an end module.

Currently the only drive that requires DC Sync all the time are the Emerson/Control Techniques, ABB, Mitsubishi, and Sanyo Denki drives; all others can run in free-run mode and will interpolate the commanded position. There are two possible syncs: Sync0 and Sync1. Sync0 alone is used more commonly than both Sync0 and Sync1. The example below shows how to enable both. To disable Sync1 in the example, set its shift time from Sync0 to 0.

Only Yaskawa Sigma 5 supports Sync0 & Sync1. Other drives are limited by the manufacturer. AMC drives do not support DC Sync. Fully tested and supported drives with DC Sync are Yaskawa, Copley, Sanyo Denki, Mitsibushi, and Emerson. Mitsibushi and Sanyo Denki must have DC Sync enabled prior to being operational thus they are always set to 1mS clock and 250uS offset on Sync0.

The master will attempt to sync to the clock of the slave that is assigned the role of distributed clock master, but expect drift to occur due to variations in the operating system and interrupt overhead. The more drives in the system, the more drift. The cyclic data will be consistent (control loop time), but the point at which the slave receives the data will drift anywhere from a few nanoseconds up to an estimated 75µS, constantly re-syncing, with an average jitter of about 2 µS. On most drives this is not a problem as the drive will interpolate. Many drives, such as the Yaskawa, sample at very high rates (62.5 µS). It is best to set the dc sync to the same value as the Master PDO control loop update time.

```
// Activate DC Sync0 each cycle time with no Sync1, always do before
// 'drive enable'
// dcsync <slave node or -1 for current>,
//    <Sync0 Cycle Time in nanoseconds, ns>,
//    <Sync1 shift from Sync0, ns>,
//    <Sync0 shift from Cycle Time, ns>,
//    <Sync start delay in ns>
// Set all parameters to 0 except the slave node to deactivate.
// Below is a 1mS Sync0 cycle time with no Sync1, control loop is 1 mS.
// Sync0 starts at cycle time and is not shifted and there is a
// 100mS delay before it all starts the first sequence.
      delay 2000 ms;    // Needed for restarts so have idle time on clock off.
      dcsync -1, 1000000, 0, 0, 100000000;
      delay 105 ms;     // starts 100 milliseconds into the future

// Enable the drive, turning power on to the amplifier.  The current position
// will be constantly updated so the drive does not move.

      drive enable;
```

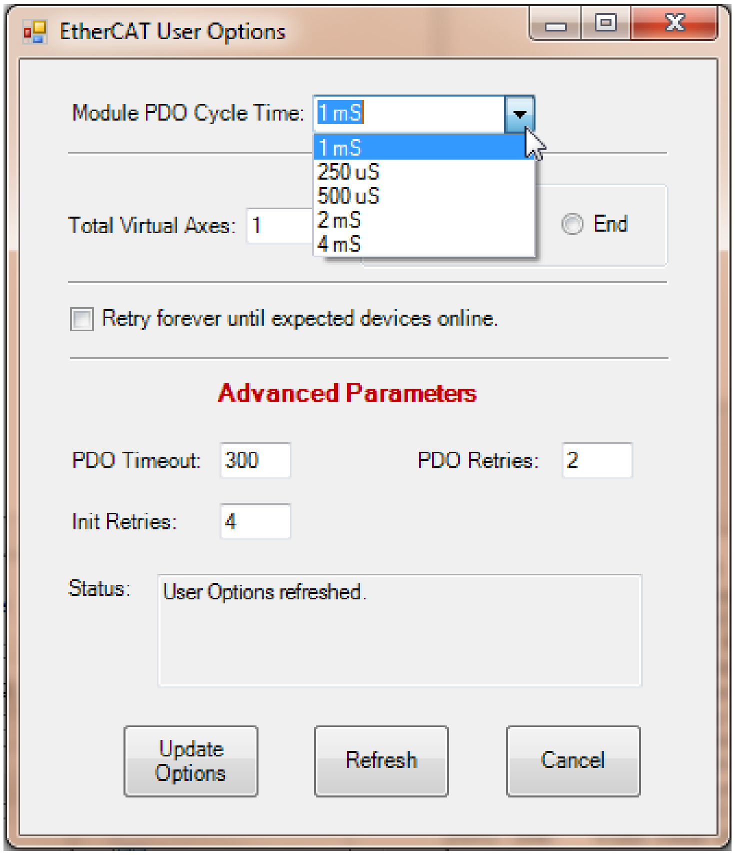
It is best to use DC Sync when using Cyclic Synchronous Position mode, especially with machining operations. Failure to enable can cause a small amount of infrequent servo noise as the drive interpolates the commanded position.

Some devices will not operate correctly unless the DC Sync is enabled (e.g., Emerson/Control Techniques). It is also best to set the DC Sync prior to enabling the drive since some drives, such as Emerson, require this.

When using DC Sync, or any multiple drive systems, it is best to verify all drives are enabled prior to executing a MOVE command. The DC Sync command can take time to execute since it must place the drive in a non-operational state, initiate numerous commands and then make it operational again. Only one DC Sync command will execute at a time, with other axis locked out until completion. In a large system this can cause several seconds of delay where the first drive is ready to execute but the last drive is still enabling DC Sync and the drive. Reference Chapter 4, “*Accessing Properties of Another Access”*.

EtherCAT Master Control Loop Cycle Time

The M3-41 can support an EtherCAT scan time of 500 µS, 1 mS, 2 mS, or 4 mS. The User Options form (Chapter 7), available within the EtherCAT Explorer, is used to set the desired speed. The network has to be restarted or controller rebooted to change the control loop time.



EtherCAT User Options

Module PDO Cycle Time: 1 mS

Total Virtual Axes: 1

☐ Retry forever until expected devices online.

Advanced Parameters

PDO Timeout: 300

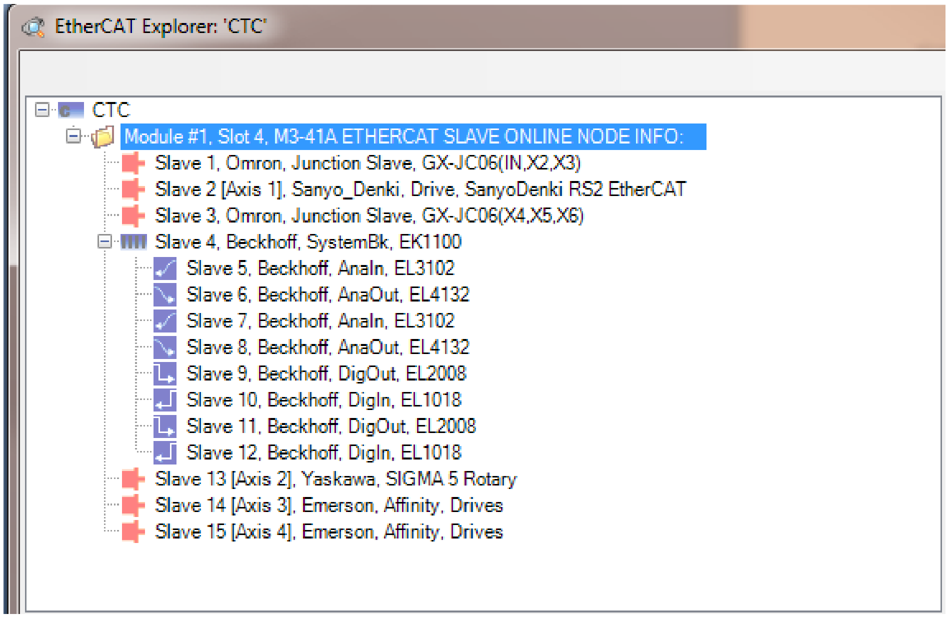
PDO Retries: 2

Init Retries: 4

Status: User Options refreshed.

For most applications it is best to use the default 1 mS scan time. Drives tend to lag regardless of how fast the position information is updated (CSP mode), this is dependent on the smoothing algorithm resident in all drives. **When running 500 µS it is recommended that no more than 4 drives be used**, some applications may support up to 6 but it is dependent on optional IO and the application program being executed. Applications which do not use the ‘host read/write’ commands can support more drives.

One way to determine if your application is approaching the limit is to view the Log Buffers. The following logs are from the Test Suite application in the Appendix, running two Emerson, one Sanyo Denki, and one Yaskawa drive with a DC Sync of 500 µS:



```
CTC
  Module #1, Slot 4, M3-41A ETHERCAT SLAVE ONLINE NODE INFO:
    Slave 1, Omron, Junction Slave, GX-JC06(IN,X2,X3)
    Slave 2 [Axis 1], Sanyo_Denki, Drive, SanyoDenki RS2 EtherCAT
    Slave 3, Omron, Junction Slave, GX-JC06(X4,X5,X6)
    Slave 4, Beckhoff, SystemBk, EK1100
      Slave 5, Beckhoff, AnaIn, EL3102
      Slave 6, Beckhoff, AnaOut, EL4132
      Slave 7, Beckhoff, AnaIn, EL3102
      Slave 8, Beckhoff, AnaOut, EL4132
      Slave 9, Beckhoff, DigOut, EL2008
      Slave 10, Beckhoff, DigIn, EL1018
      Slave 11, Beckhoff, DigOut, EL2008
      Slave 12, Beckhoff, DigIn, EL1018
    Slave 13 [Axis 2], Yaskawa, SIGMA 5 Rotary
    Slave 14 [Axis 3], Emerson, Affinity, Drives
    Slave 15 [Axis 4], Emerson, Affinity, Drives
```

```
*** Module #1, Slot 4 ***
M3-41A ETHERCAT MASTER
INFO: Time:    2308.792, Scanning = 2, Cycle 0.5000 mS, [Overhead 0.3326 mS, Min 0.1584 mS, Max 0.4522 mS],
        Adjusted Tick 0.4992 mS, Correction -820 ns, Max Tick 0.5000 mS, [Idle Time 0.0860 mS, Min 0.0031 mS, Max 0.3465].
        [Sync Time 0.0070 mS, Min 0.0000 mS, Max 0.2500, Avg 0.0095].
```

Reference the ‘Overhead’ information with the Max at .4522 mS. This means that you had 500 – 452 or 48 µS of spare control loop time, worst case. Once you exceed 500 a warning will occur, 550 an error. It is recommend that this number stay below .425 mS. This particular application timing was from a stress test with 4 drives, each of which were reading and writing local controller registers (host read/write) while at the same time executing motion commands. Also note the Error Time of a Max .250 mS, with an average jitter of 9.5 µS, this is excessive but still functional.

The following is the same test with no host read/write instructions:

```
*** Module #1, Slot 4 ***
M3-41A ETHERCAT MASTER
INFO: Time:    2815.575, Scanning = 2, Cycle 0.5000 mS, [Overhead 0.3168 mS, Min 0.2004 mS, Max 0.4157 mS],
        Adjusted Tick 0.5000 mS, Correction -3 ns, Max Tick 0.5001 mS, [Idle Time 0.1534 mS, Min 0.0042 mS, Max 0.2542].
        [Sync Time 0.0005 mS, Min 0.0000 mS, Max 0.0651, Avg 0.0026].
```

Note how the ‘Overhead’ Max has dropped to .415 mS and the Error Time Max is .065 mS with an average jitter just 2.6 µS. This application will run fine at 500 µS scan time.

DC Sync should be set to match the cycle loop time, for example, 500000 nS, not 1000000 nS if 500 µS is enabled. This is especially critical on Emerson drives. It is also recommended that common bits be used instead of the host read/write instruction, wherever possible.

CHAPTER 6

[6] EtherCAT IO, PLS, & PWM

Inputs/Outputs

EtherCAT IO can be accessed by QuickBuilder application programs as though it is local IO, appearing transparently as digital and analog inputs/outputs. QuickBuilder MSB's have additional capabilities to access IO that is resident on a drive, the local M3-41 module, and/or remotely on a slave device such as a Wago, Turck or Beckhoff IO block. IO can not only be accessed by an MSB controlling a physical drive but a virtual drive, in fact, an EtherCAT network with no drives can be designed with nothing but virtual drives controlling the IO allowing for a very flexible system.

A number of MSB IO arrays are available with the indexes into these arrays determining the source or destination of the operation.

<index>: (may be an immediate numeric or indirect via a variable reference)

Bit oriented indexes: an array index of 1 to 32 references each input or output bit available on the drive whose axis is assigned to the MSB. An array index of 501 to 1000 is reserved for local module IO (global_inputs/global_outputs), where 501 is the first. An array index of 1001 to 2025 is reserved for remote IO device blocks, such as Wago, Turck, Beckhoff, and SMC, where 1001 is the first.

Byte oriented indexes: an array index of 1 to 4 references each input or output byte available on the drive whose axis is assigned to the MSB. An array index of 501 to 1000 is reserved for local module IO bytes (global_inputs/global_outputs). An array index of 1001 to 2025 is reserved for remote IO device blocks, such as Wago, Turck, Beckhoff, and SMC.

Word (32 bit) oriented indexes: An array index of 1001 to 2025 is reserved for remote analog IO device blocks, such as Wago, Turck, Beckhoff, and SMC.

Arrays:

inputs[ ] – Digital input bits, where 1 is the first input.
inputs8[ ] – Digital input bytes, atomic to the byte level, where 1 is the first byte.
outputs[ ] – Digital output bits, where 1 is the first output.
outputs8[ ] - Digital output bytes, atomic to the byte level, where 1 is the first byte.
ains[ ] – Analog inputs, 32 bits. If the device being accessed is 16 bits, 0x0000 is added to the high bytes.
aouts[ ] – Analog outputs, 32 bits. If the device being accessed is 16 bits the data will be truncated.

Example using Output bytes:

```
// Attached device is a Wago IO block with 40 digital outputs,
// 40 digital inputs, 8 analog outputs, and 8 analog inputs.
//
// Shift 8 bits on the output
//
// Write a byte output, index can be immediate numeric or variable
// reference.
[test1]
 [_begin]
outputs8[1001] = 0x01;  // 1001 is the first remote 8 bit output block,
                        // write a 1 to first bit
[loop]
delay 250 ms;
// Shift the bit up by one and update the output
outputs8[1001] = outputs8[1001] << 1;
if (outputs8[1001] != 0) goto loop;
goto _begin;
```

Example using Output bits:

```
// Attached device is a Wago IO block with 40 digital outputs,
// 40 digital inputs, 8 analog outputs, and 8 analog inputs.
//
// Shift 8 bits on the output
//
// Write each output a bit at a time, shifting active output up by
// 1 each time.
[test2]
index = 1001;
[_begin1]
outputs[index] = 0x01;  // 1001 is the first remote output block,
                        // write a 1 to first bit
[loop1]
delay 250 ms;
// Turn the current output off
outputs[index] = 0;
// Point to next output bit
index = index+1;
// turn the next output on
if (index != 1009) goto _begin1;
```

```
      index = 1001;      // start at beginning again
      goto _begin1;
```

Example using analog input/output:

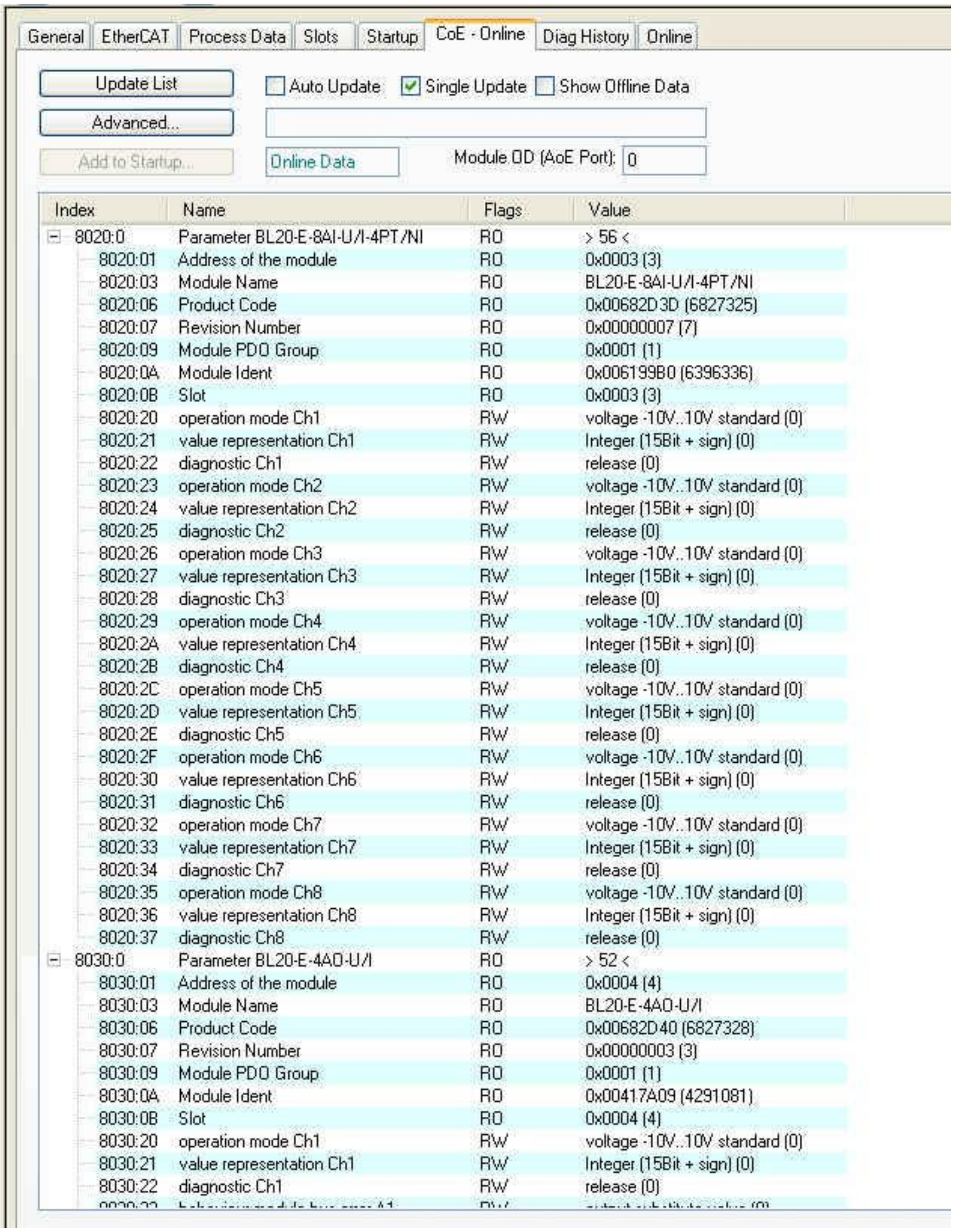
```
      // Test remote Analog input/outputs #5
      [test3]
      index = 1005;
      val = 0;
      [loop3]
      // update the analog output value
      aouts[index] = val;
      // Allow analog out to stabalize
      delay 250 ms;
      // Analog output is looped back to Analog input via external wire
      myval = ains[index];
      val = val + 100;        // increase analog output value
      if (val <= 10000) goto loop3;
      goto test3;  // reset value
```

Example using digital input:

```
      // Mapped global input example
      // Map drive input 1 to global input 1 so can use falloff or
      // riseof commands to monitor its state
      Set mapped input 1 to input 501;
      [top]
      On fallof 1 goto falledge;
      goto top;
      [falledge]
      // do whatever desired when falling edge detected
      // and await level state again…
      // … do something until back to high again …
      if inputs[1] goto top;
      goto falledge;  // waiting for signal to go high again
```

EtherCAT IO Configuration

Some EtherCAT IO, such as Wago and Turck have modules that are configurable. If a special mode, other than the default is needed it is suggested that either that mode be saved to the devices EEPROM using the Beckhoff EtherCAT Configurator or a specific SDO write command be issued to the device's configuration object by a virtual axis, at initialization. Refer to the specific manufacture's manual for object address information. For Wago and Turck this is typically the 0x8000 * (slot number -1) * 0x0010 object, with each index offering a specific property. Below shows a Turck IO device with an analog input module in slot 3 and analog output in slot 4.



□ Auto Update ☑ Single Update □ Show Offline Data

Online Data

Module OD (AoE Port): 0

Index	Name	Flags	Value
⊟ 8020:0	Parameter BL20-E-8AI-U/I-4PT/NI	RO	> 56 <
8020:01	Address of the module	RO	0x0003 (3)
8020:03	Module Name	RO	BL20-E-8AI-U/I-4PT/NI
8020:06	Product Code	RO	0x00682D3D (6827325)
8020:07	Revision Number	RO	0x00000007 (7)
8020:09	Module PDO Group	RO	0x0001 (1)
8020:0A	Module Ident	RO	0x006199B0 (6396336)
8020:0B	Slot	RO	0x0003 (3)
8020:20	operation mode Ch1	RW	voltage -10V..10V standard (0)
8020:21	value representation Ch1	RW	Integer (15Bit + sign) (0)
8020:22	diagnostic Ch1	RW	release (0)
8020:23	operation mode Ch2	RW	voltage -10V..10V standard (0)
8020:24	value representation Ch2	RW	Integer (15Bit + sign) (0)
8020:25	diagnostic Ch2	RW	release (0)
8020:26	operation mode Ch3	RW	voltage -10V..10V standard (0)
8020:27	value representation Ch3	RW	Integer (15Bit + sign) (0)
8020:28	diagnostic Ch3	RW	release (0)
8020:29	operation mode Ch4	RW	voltage -10V..10V standard (0)
8020:2A	value representation Ch4	RW	Integer (15Bit + sign) (0)
8020:2B	diagnostic Ch4	RW	release (0)
8020:2C	operation mode Ch5	RW	voltage -10V..10V standard (0)
8020:2D	value representation Ch5	RW	Integer (15Bit + sign) (0)
8020:2E	diagnostic Ch5	RW	release (0)
8020:2F	operation mode Ch6	RW	voltage -10V..10V standard (0)
8020:30	value representation Ch6	RW	Integer (15Bit + sign) (0)
8020:31	diagnostic Ch6	RW	release (0)
8020:32	operation mode Ch7	RW	voltage -10V..10V standard (0)
8020:33	value representation Ch7	RW	Integer (15Bit + sign) (0)
8020:34	diagnostic Ch7	RW	release (0)
8020:35	operation mode Ch8	RW	voltage -10V..10V standard (0)
8020:36	value representation Ch8	RW	Integer (15Bit + sign) (0)
8020:37	diagnostic Ch8	RW	release (0)
⊟ 8030:0	Parameter BL20-E-4AO-U/I	RO	> 52 <
8030:01	Address of the module	RO	0x0004 (4)
8030:03	Module Name	RO	BL20-E-4AO-U/I
8030:06	Product Code	RO	0x00682D40 (6827328)
8030:07	Revision Number	RO	0x00000003 (3)
8030:09	Module PDO Group	RO	0x0001 (1)
8030:0A	Module Ident	RO	0x00417A09 (4291081)
8030:0B	Slot	RO	0x0004 (4)
8030:20	operation mode Ch1	RW	voltage -10V..10V standard (0)
8030:21	value representation Ch1	RW	Integer (15Bit + sign) (0)
8030:22	diagnostic Ch1	RW	release (0)

Mappable Input IO & Counters

The first five inputs of each axis are by default mapped to the available drive inputs, with up to 32, depending on those available for the specific drive. These inputs have a number of MSB commands available for special handling (reference the QuickMotion Reference Guide):

```
on asynchevent asynchhandler
wait for transition of input { or condition }
pls output using reference definitions
set capture transition of input input
variable = ctr[n]
ctr[n] = expression
ctr[n] = offset
variable = dins
variable = din1
variable = din2
variable = din3
variable = din4
variable = din5
```

An additional feature is the ability to remap the first five inputs to any other available input. This can then be used to drive the above commands, including counters, ctr0 to ctr4. The 32 bit property ‘dins’ will have its first 5 bits originating from the mapped location, with the remainder from the drive. If the first 5 bits are needed from the drive they can be read using the input[ ] array.

The command to remap an input is as follows:

syntax

```
set mapped input index to input input { count edge }
```

parameters

`index`
The index of the mapped input assigned the input to control.
1-5 index

`input`
the input to assign to the index
0 to disable mapping and restore to default drive input, counter disabled
1-32 on drive
501 to 1000 local to M3-41 module
1001 to 2025 remote EtherCAT IO block

`edge`
rising or *falling*, optional, with default being *rising*

Note: index 1 is assigned to ctr0, 2 to ctr1, etc., also counter cleared upon execution.

Example:

```
// Map dins bits 0 to 4 to remote EtherCAT IO inputs 25 to 29.
// The counters ctr0 to ctr4 will be observed using QuickBuilder
// watch window since remote output #25 is connected to input
// #25, it will count on the rising edge once per second.
set mapped input 1 to input 1025 count rising;
set mapped input 2 to input 1026 count falling;
set mapped input 3 to input 1027 count rising;
set mapped input 4 to input 1028 count rising;
set mapped input 5 to input 1029 count rising;
[loop]
outputs[1025] = 1;
delay 500 ms;
outputs[1025] = 0;
delay 500 ms;
goto loop;
```

Pulse & PWM Generation

Timed pulses and PWM can be generated using the EtherCAT control loop time as a time tick. The MSB language has a number of instructions in this regard:

syntax

pulse_ext *output* **for** *n*

parameters

output the output to pulse
1-32 on drive
501 to 1000 local to M3-41 module
1001 to 2025 remote EtherCAT IO block

n the time to pulse the output, expressed as control loop ticks
(up to 5 pulsed outputs may be active at one time)

This statement causes the specified *output* to pulse for the specified duration of EtherCAT control loop *ticks*. The *output* follows the same access numbering convention as the outputs[ ] array. If the output is already *on* when this statement executes, the output state is unchanged, however it will be turned *off* after the specified time.

If another statement changes the state of the output to off before the allotted duration, the generation of the pulse is aborted. The generated pulse will be synced & set active on the next control loop tick.

Example:

```
// Test pulse and output on
[test4]
pulse_ext 1002 for 500; // turn output 2 on for 500 ms
delay 1000 ms;          // this should make it appear as on 1/2, off 1/2
second.
goto test4;
```

syntax

generate output *output* **rate** *freq*

parameters

output
the output to pulse
1-32 on drive
501 to 1000 local to M3-41 module
1001 to 2025 remote EtherCAT IO block

freq
the frequency (in control loop ticks) to generate 50% duty cycle pulses; rounded to an integer. Specifies the on time.

This statement begins or ends generation of pulses using a specific output. When a frequency of 0 is specified, no pulse generation occurs. This effectively turns the output back into a general-purpose output. A 50% duty cycle is generated, thus a rate of 1 would turn the output on during one control loop cycle and off on the next. A 1 mS control loop would yield a 500 HZ output square wave.

Example:

```
// Test square wave on output #1
[test5]
// Turn output 1 on for 75 ticks, off for 75 ticks
generate output 1001 rate 75;
// Turn output 2 on for 125 ticks, off for 125 ticks
generate output 1002 rate 125;
// Turn output 3 on for 250 ticks, off for 250 ticks
generate output 1003 rate 250;
// Turn output 4 on for 500 ticks, off for 500 ticks
generate output 1004 rate 500;
// Turn output 5 on for 1000 ticks, off for 1000 ticks
generate output 1005 rate 1000;
[loop5]
goto loop5;       // stall, pulses will continue forever...
```

syntax

pwm **output** *output* on *tickson* off *ticksoff* cycles *n*

parameters

output
the output to pulse
1-32 on drive
501 to 1000 local to M3-41 module
1001 to 2025 remote EtherCAT IO block

tickson
the number of control loop ticks to activate the output.

ticksoff
the number of control loop ticks to de-activate the output.

n
The number of complete PWM cycles to do prior to stopping where 0 terminates an active PWM cycle and restores the state to an inactive output and -1 runs forever.

This statement will generate a certain number of variable width output *cycles* with the desired *on* and *off* time in control loop *ticks*. If the *cycles* or *on* time are 0 the PWM will terminate immediately and the output set inactive. If the *cycles* is '-1' the PWM will run forever or until stopped by another instruction with cycles/on time set to 0.

Example with immediate references:

```
// Test PWM output on output 2
[test6]
pwm output 1002 on 125 off 1000 cycles -1;
[loop6]
goto loop6;              // stall, pwm will continue forever...
```

Example with variable references:

```
// Test PWM output on output 2
[test6]
index = 1002;
ontime = 125;
offtime = 1000;
pwm_cycles = -1;
pwm output index on ontime off offtime cycles pwm_cycles;
[loop6]
goto loop6;              // stall, pwm will continue forever...
```

Example with embedded math:

```
// Test PWM output on output 2
[test6]
ontime = 1000;
pwm output 1002 on (ontime/8) off 1000 cycles -1;
[loop6]
goto loop6;              // stall, pwm will continue forever...
```

PLS Outputs

PLS, or Programmable Limit Switch, is an output which can be configured to become active at a high rate of speed based upon some monitored event or state. Typically PLS outputs must become active faster than instructions in an MSB can execute testing the required conditions. For example with a 500 uS control loop the PLS output conditions and state would be tested every 500 uS, much faster than the MSB could execute. The M3-41 allows any output that is resident on a drive, local to the M3-41 module, or a general EtherCAT output to be used as a PLS output. The *output* follows the same access numbering convention as the outputs[ ] array. The available commands are as follows:

syntax

```
set pls index to output output
```

parameters

`index` The index of the PLS assigned the output to control.
1-5 index

`output`	the output to activate under PLS control 1-32 on drive 501 to 1000 local to M3-41 module 1001 to 2025 remote EtherCAT IO block

There exist five possible PLS functions that may be active per axis. Each of these PLS functions must be assigned the output they are to trigger when the specified event occurs. This instruction assigns the output number to each of the PLS index functions. The index requires this assignment in order to know what output to operate on for all the other PLS instructions. Note that the use of this command will disable and active PLS on the specified index and clear the 'mod' to 0 (not used).

syntax

`set pls` *`index`* `mod` *`mod`*

parameters

`index`	The index of the PLS assigned the output to control. 1-5 index
`mod`	The modulus to apply to the PLS reference, whereby the remainder is the resulting value, prior to comparison of a window value. This allows for taking into account rollover. For example if there are 65536 counts per revolution and the reference window is defined for the first revolution, then each revolution would be able to reference the same set of window values. Default is 0 meaning the reference value is not changed.

There exist five possible PLS functions that may be active per axis. Each of these PLS functions may be assigned a modulus value to apply to the reference value prior to window comparison. For example if 'fposc' is the reference and its value is 1025674 and the modules was 65536 (possibly pulses per revolution), then the actual reference used would be 1025674 % 65536 or 42634, the remainder.

syntax

`pls` *`index state`*

parameters

`index`	The index of the PLS assigned the output to control. 1-5 index
`state`	**`on`** or **`off`**

This statement enables ("on") or disables ("off") a PLS for an output active for the given index.

- **on** - Enables the pls functionality initialized for a particular output with the PLS Define statement.

- **off** – Disables the pls functionality initialized for a particular output with the PLS Define statement.

If the output is on when a PLS is disabled, it will remain on – unless the user re-enables the PLS (to re-compute the PLS output).

pls *index* **using** *reference definitions*

parameters

index	The index of the PLS assigned the output to control, 1 to 5.
reference	the encoder count scaled reference variable to compare to:

fposc	Feedback position of axis msb
mposc1 - mposc5	Master position counters #1 through #5
mposc	Master position counter
smodc	Slave position (modulo)
smark	Slave marked position
tmc1 tmc2	Temporary master counters #1 & #2
tsc1 tsc2	Temporary slave counters #1 & #2
sdc	Slave decrement counter
fposc1	Feedback position of axis 1 (fposcA)
fposc2	Feedback position of axis 2 (fposcB)
tmodc	Temporary master counter mod mmc
sfposc	**Secondary feedback position of axis**
tposc	Target position of axis
ctr0	din1 mapped input counter
ctr1	din2 mapped input counter
ctr2	din3 mapped input counter
ctr3	din4 mapped input counter
ctr4	din5 mapped input counter
ctr5	Local quadrature encoder 1
ctr6	Local quadrature encoder 2
ctr7	Local quadrature encoder 3

definitions a comma-separated list of up to 16 PLS definitions:

on x **to** y	Turn output on when the reference is within the bounds specified by x through y (may be expressions)

This statement defines, or redefines, a PLS associated with a given output and its operation. When a PLS is defined/re-defined it will be disabled and will not compute the state for the output. To enable a PLS after it is defined/re-defined, a *pls <index> on* statement must be issued.

Example:

```
[beginTest]

// Assign the first remote output on a Wago IO Block to PLS index 1.
set pls 1 to output 1001;
```

```
// Output will be on when fposc is within 100000-250000 or 1750000-3000000
pls 1 using fposc on 100000 to 250000, on 1750000 to 3000000;
// enable the PLS for index #1
pls 1 on;

// Assign the first remote output on a Wago IO Block to PLS index 1.
set pls 2 to output 1005;
// Output will be on when fposc is within10-200000 or 250000-1750000
pls 2 using fposc on 0 to 200000, on 250000 to 1750000;
// enable the PLS for index #1
pls 2 on;

[run]
// Begin the move, .05 rev/second for 2 revolutions
move at 0.05 for 2;
wait for in position;
// Delay 1 second once in position
delay 1000 ms;
// Do a relative move back 2 revolutions at .05 rev/second
move at 0.05 for -2;
wait for in position;
// Delay 1 second once in position
delay 1000 ms;
// Do it again, forever...
goto run;
```

Blank

CHAPTER 7

[7] Error Handling

Each drive manufacturer lists its own error codes. The combination of EtherCAT and the many different servo drives available results in thousands of different error possibilities. The network is continually monitored for error conditions and emergency messages from the slave devices. In the event of an error, the MSB typically performs a Quick Stop operation and the EtherCAT master stops scanning and awaits program RESTART. Power to the amplifier is turned off when scanning stops, causing the servo to freewheel. Any braking needed during this state should be considered.

Both informational and error conditions are constantly logged to a universal message buffer that resides within the M3-41 module. The message buffer contents can be viewed in the QuickBuilder EtherCAT Explorer or by using telnet commands.

Retry Logic

EtherCAT networks are fairly robust but intermittently a packet can be lost due to noise affecting the cabling. The M3-41 has built in retry logic to attempt a recovery before considering a lost packet a non-recoverable error. With poll times set to the standard 1 mS a retry will occur up to two times if a packet is not returned within 300 uS of transmission. At a poll time of 500 uS this is shortened to 200 uS. If a retry occurs an information message will be logged to alert the user. With the EtherCAT Master dynamically adjusting its scan timer, the Master will quickly re-sync to the reference slave clock if a retry is successful. Reference Chapter 7, the EtherCAT Explorer 'User Options', if the default retry logic needs to be modified.

Drive Diagnostic Variables and Registers

Special QuickBuilder variables and controller registers are available to monitor EtherCAT operation and provide post analysis after faulting. The best method is by the use of the EtherCAT Explorer but in some cases a remote HMI or user may want to access the information that the Explorer does, except at the error code/register level.

<u>QuickBuilder MSB variables by axis:</u>

dwSlaveID – The EtherCAT slave ID as it appears in the EtherCAT Explorer.

faulted – 0, no fault, 1 fault on this axis.
faultOpcode – Type of fault, reference following section for definition.
wStatus – Current PDO status read from the drive, object 0x6041.
wControlWord – Current PDO control command to the drive, object 0x6040, reference manufacturer manual.
errorType – Last error type logged.
errorRegister – Object 0x1001 error register in drive if supported, see appendix for drive manufacturer.
errorCode – Last error from drive, typically object 0x603F if supported, see appendix for drive manufacturer.
last_ALStatusCode – Additional error information from drive, object 0x134:0x135.

By 5300 Controller Register:

Register 13700 – Axis Display Index Register. Set to 0, default, 14XX0 block same as defined in the 'Model 5300 Quick Reference Register Guide', set to 1 and maps as follows where XX is the axis number starting with 00 for axis 1 allowing for up to 100 drives:

Register	QuickBuilder MSB Variable
14XX0	dwSlaveID
14XX1	faulted
14XX2	faultOpcode
14XX3	wStatus
14XX4	wControlWord
14XX5	errorType
14XX6	errorRegister
14XX7	errorCode
14XX8	last_ALStatusCode
14XX9	Not Available

MSB 'wStatus' Variable Bit Definitions

'wStatus' references the drive's last Status Word, object 0x6041, read. References the specific drive Manufacture for AL additional details:

Bit	QuickBuilder MSB Variable
0	Ready to switch on
1	Switched on
2	Operation enabled

3	Fault
4	Voltage enabled
5	Quick stop
6	Switch on disabled
7	Warning
8	-
9	Remote
10	Target reached
11	Internal limit active
12	Operation mode specific
13	Operation mode specific
14	Torque limit active
15	-

MSB ‘errorType’ Variable Value Definitions

The ‘errorType’ variable defines the type of error that has occurred as identified by the M3-41 module. This typically allows CTC to locate where it was during software execution when the error occurred.

errorType	Description
0	*INFORMATIONAL_ONLY*
1	*INFORMATIONAL_ONLY_MALLOCED*
2	*ERROR_DEFAULT*
3	*ERROR_ECAT_PROFILE_POS_INIT*
4	*ERROR_ECAT_PROFILE_POS_STARTING1*
5	*ERROR_ECAT_PROFILE_POS_STARTING2*
6	*ERROR_ECAT_PROFILE_POS_RUNNING*
7	*ERROR_ECAT_HOMING_INIT*
8	*ERROR_ECAT_HOMING_STARTING1*
9	*ERROR_ECAT_HOMING_STARTING1A*
10	*ERROR_ECAT_HOMING_STARTING2*
11	*ERROR_ECAT_HOMING_RUNNING*
12	*ERROR_ECAT_PROFILE_VEL_WAIT*

errorType	Description
13	*ERROR_ECAT_PROFILE_WAIT_QSTOP*
14	*ERROR_RUNNING*
15	*ERROR_TRACKING*
16	*ERROR_LOST_CONNECTION*
17	*ERROR_Network_Interface*
18	*ERROR_No_Slaves_Found*
19	*ERROR_Not_All_Slaves_PreOp*
20	*ERROR_Not_All_Slaves_Operational*
21	*ERROR_PDO_Init_Failed*
22	*ERROR_ec_config_map_Failed*
23	*ERROR_Slave_Unknown*
24	*ERROR_Init_Send_Processdata*
25	*ERROR_Init_Receive_Processdata*
26	*ERROR_No_Slaves_For_DC*
27	*WARNING_No_Station_Alias*
28	*ERROR_DC_SYNC0_Failed*
29	*ERROR_ECATLoop_Execution_Exceed_Scantime*
30	*ERROR_MEMORY_BUFFER_EXCEEDED*
31	*ERROR_ECATLoop_Event_Timeout*
32	*ERROR_ECATLoop_RX_Timeout*
33	*ERROR_ECATLoop_TX_Timeout*
34	*ERROR_USER_SDO_READ*
35	*ERROR_USER_SDO_WRITE*
36	*ERROR_Profile_Thread*
37	*ERROR_Motion_Fault*
38	*ERROR_Dump*
39	*ERROR_USER_DCSYNC*
40	*ERROR_Not_All_Slaves_SafeOp*
41	*ERROR_Not_All_Slaves_Found*
42	*ERROR_Slaves_Not_Match_Expected*
43	*ERROR_Duplicate_Axis*
44	*WARNING_ECATLoop_Execution_Exceed_Scantime*
45	*ERROR_Slave_ALState_NotOperational*
46	*ERROR_ec_config_map_Too_Many_Segments*

errorType	Description
47	*ERROR_Slave_Not_Synced*
48	*ERROR_Drive_PowerUP_Fault*
49	WARNING_Too_Many_Servos

MSB 'last_ALStatusCode' Variable Value Definitions

References the specific drive Manufacture for AL Status Codes that are not listed below or where additional definition is required. Codes are shown in hexadecimal representation:

AL Status Code	Description
0x0000	No error
0x0001	Unspecified error
0x0002	No Memory
0x0011	Invalid requested EMS change
0x0012	Unknown requested state
0x0013	Bootstrap not supported
0x0014	No valid firmware
0x0015	Invalid mailbox configuration
0x0016	Invalid mailbox configuration
0x0017	Invalid sync manager configuration
0x0018	No valid inputs available
0x0019	No valid outputs
0x001A	Synchronization error
0x001B	Sync manager watchdog
0x001C	Invalid Sync Manager Types
0x001D	Invalid Output Configuration
0x001E	Invalid Input Configuration
0x001F	Invalid Watchdog configuration
0x0020	Slave needs cold start
0x0021	Slave needs INIT
0x0022	Slave needs PREOP
0x0023	Slave needs SAFEOP
0x0024	Invalid Input Mapping

AL Status Code	Description
0x0025	Invalid Output Mapping
0x0026	Unmatched setting
0x0027	Free-run mode unsupported
0x0028	SYNC mode unsupported
0x0029	Free-run mode, 3 Buffer mode not set
0x002A	Background watchdog
0x002B	No valid inputs and outputs
0x002C	Fatal sync error
0x002D	No sync error
0x0030	Invalid DC Sync configuration
0x0031	Invalid DC Latch configuration
0x0032	PLL error
0x0033	Invalid DC IO error
0x0034	Invalid DC timeout error
0x0035	DC invalid Sync cycle time
0x0036	DC SYNC0 cycle time
0x0037	DC SYNC1 cycle time
0x0042	MBX_EOE
0x0043	MBX_COE
0x0044	MBX_FOE
0x0045	MBX_SOE
0x004F	MBX_VOE
0x0050	EEProm No Access
0x0051	EEProm Error?
0x????	Unknown error, reference manuf.

CHAPTER

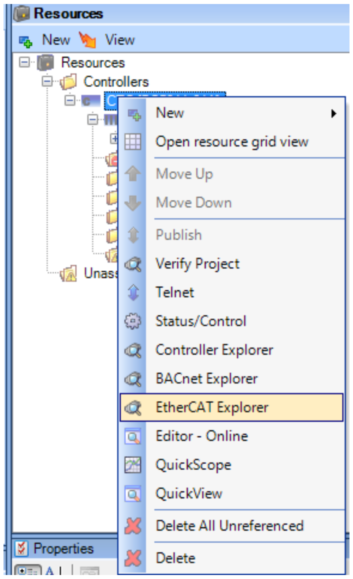
8

[8] QuickBuilder EtherCAT Explorer

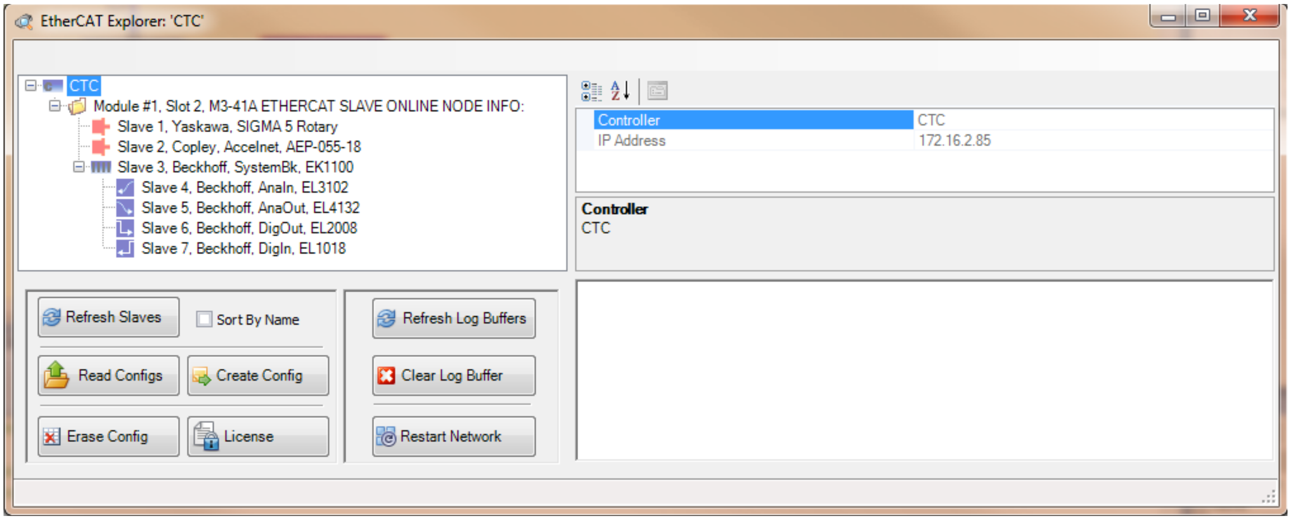
QuickBuilder provides a simple-to-use EtherCAT Explorer. The Explorer communicates directly with any model 5300 that has one or more M3-41 modules and graphically presents the network information. It also provides a high-level diagnostics capability.

QuickBuilder EtherCAT Explorer Status Window

The EtherCAT Explorer window is a feature of the QuickBuilder environment. To open the EtherCAT Explorer, right click on the controller available in Resources. A menu of options will appear; select 'EtherCAT Explorer' to connect to the defined controller.



Once invoked, a window similar to the one below will appear, enabling you to monitor the model 5300's EtherCAT module and its Master configuration.

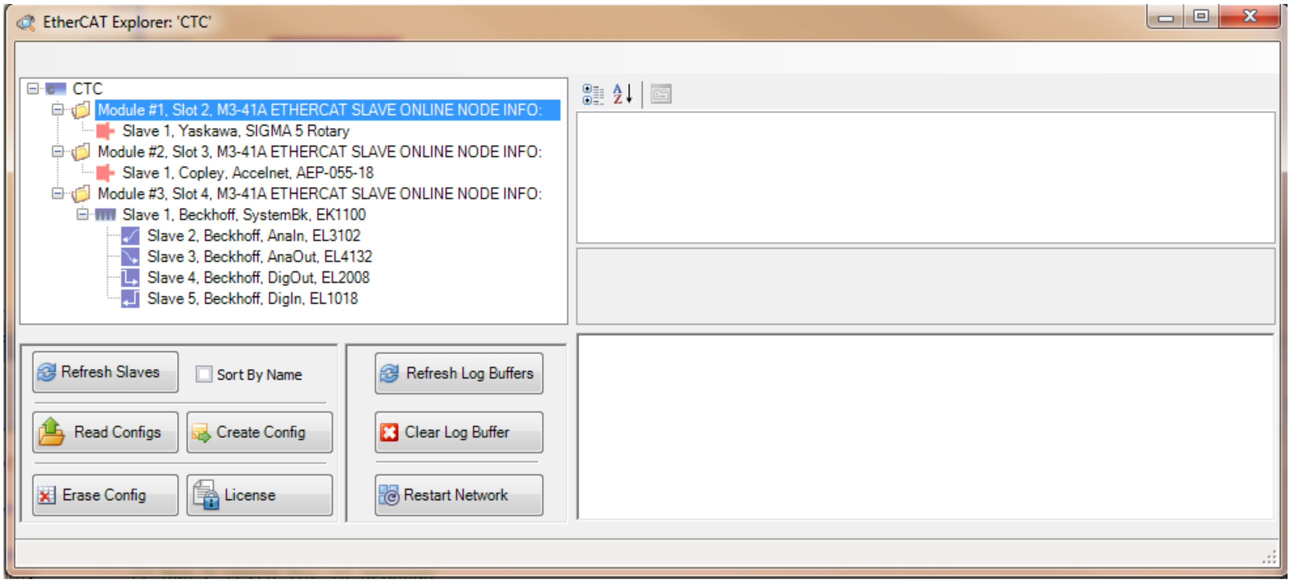


CTC
Module #1, Slot 2, M3-41A ETHERCAT SLAVE ONLINE NODE INFO:
Slave 1, Yaskawa, SIGMA 5 Rotary
Slave 2, Copley, Accelnet, AEP-055-18
Slave 3, Beckhoff, SystemBk, EK1100
Slave 4, Beckhoff, AnaIn, EL3102
Slave 5, Beckhoff, AnaOut, EL4132
Slave 6, Beckhoff, DigOut, EL2008
Slave 7, Beckhoff, DigIn, EL1018

Controller	CTC
IP Address	172.16.2.85

Controller
CTC

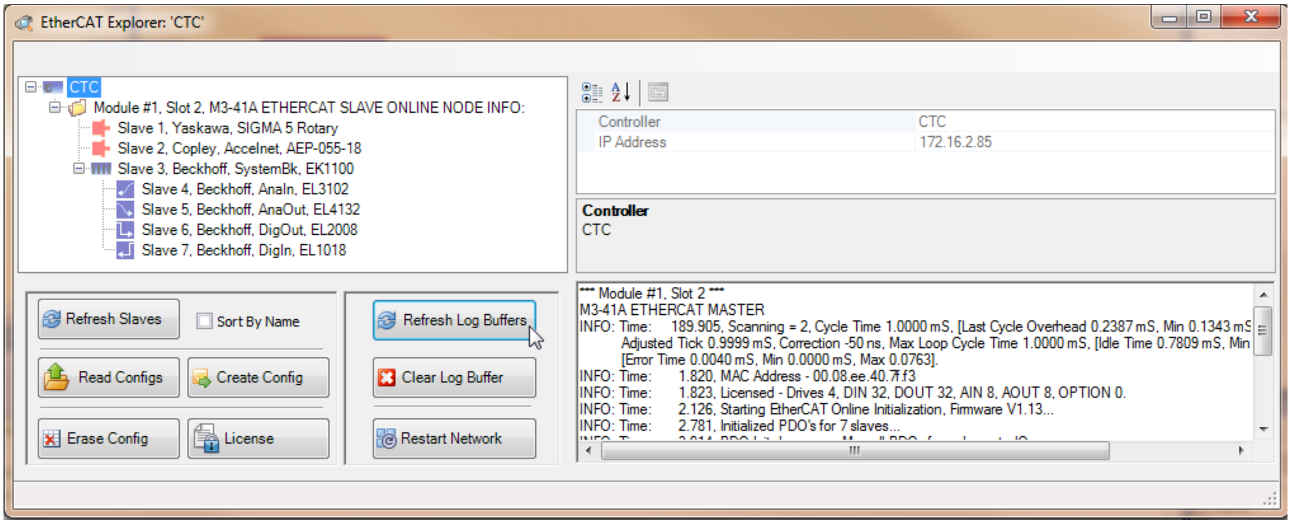
If multiple M3-41 modules are present in the 5300, a folder will appear for each module, with the slaves it controls listed below. Here is an example of three M3-41 EtherCAT Master networks in one model 5300 controller with the first M3-41 module selected (highlighted).



CTC
Module #1, Slot 2, M3-41A ETHERCAT SLAVE ONLINE NODE INFO:
Slave 1, Yaskawa, SIGMA 5 Rotary
Module #2, Slot 3, M3-41A ETHERCAT SLAVE ONLINE NODE INFO:
Slave 1, Copley, Accelnet, AEP-055-18
Module #3, Slot 4, M3-41A ETHERCAT SLAVE ONLINE NODE INFO:
Slave 1, Beckhoff, SystemBk, EK1100
Slave 2, Beckhoff, AnaIn, EL3102
Slave 3, Beckhoff, AnaOut, EL4132
Slave 4, Beckhoff, DigOut, EL2008
Slave 5, Beckhoff, DigIn, EL1018

The top left tree is known as the Slave Discovery Window. Both online slaves and the expected slave configurations appear here. Select a slave entry, and the available property information appears within the window to its right.

The window on the bottom right is known as the Message Window. As the EtherCAT Master executes, diagnostic log information is stored in the M3-41 module. By selecting 'Refresh Log Buffer' the most current contents of the log buffer will be displayed. Note that at power up the Licensing information appears in the list, "Licensed: Drives 4…". This is the total number of I/O and drives that your EtherCAT Master is authorized to control.


EtherCAT Explorer: 'CTC'
CTC
Module #1, Slot 2, M3-41A ETHERCAT SLAVE ONLINE NODE INFO:
Slave 1, Yaskawa, SIGMA 5 Rotary
Slave 2, Copley, Accelnet, AEP-055-18
Slave 3, Beckhoff, SystemBk, EK1100
Slave 4, Beckhoff, AnaIn, EL3102
Slave 5, Beckhoff, AnaOut, EL4132
Slave 6, Beckhoff, DigOut, EL2008
Slave 7, Beckhoff, DigIn, EL1018
Refresh Slaves
Sort By Name
Read Configs
Create Config
Erase Config
License
Refresh Log Buffers
Clear Log Buffer
Restart Network

Controller	CTC
IP Address	172.16.2.85

Controller
CTC

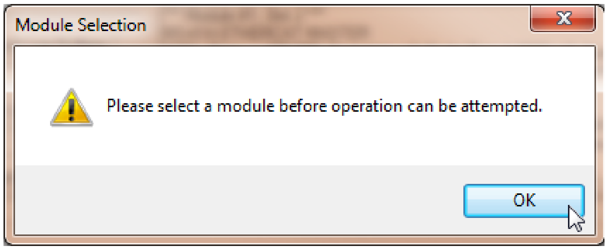
```
*** Module #1, Slot 2 ***
M3-41A ETHERCAT MASTER
INFO: Time:    189.905, Scanning = 2, Cycle Time 1.0000 mS, [Last Cycle Overhead 0.2387 mS, Min 0.1343 mS
          Adjusted Tick 0.9999 mS, Correction -50 ns, Max Loop Cycle Time 1.0000 mS, [Idle Time 0.7809 mS, Min
          [Error Time 0.0040 mS, Min 0.0000 mS, Max 0.0763].
INFO: Time:      1.820, MAC Address - 00.08.ee.40.7f.f3
INFO: Time:      1.823, Licensed - Drives 4, DIN 32, DOUT 32, AIN 8, AOUT 8, OPTION 0.
INFO: Time:      2.126, Starting EtherCAT Online Initialization, Firmware V1.13...
INFO: Time:      2.781, Initialized PDO's for 7 slaves...
```

The panel on the lower left contains a number of buttons. Some are for global access; others are for individual M3-41 modules. Those operating an individual EtherCAT Master Module (M3-41) require the Slave Discovery Tree entry with the folder icon (Module #, Slot) to be selected to identify which module is to be accessed. The following operations are available:

Refresh Slaves – Updates the Slave Discovery Window with all online slaves observed at the last restart for all installed M3-41 modules. Each slave's properties are also refreshed to the most current. These properties vary by drive. Drive information will contain present PDO contents, position and state information, etc.

Read Configs – Updates the Slave Discovery Window with any saved configuration file whose content resides in the module's non-volatile storage for all installed M3-41 modules. The information displayed is what is required to be online for the network to become active.

Create Config – This button erases the non-volatile memory stored in the selected module and writes XML information matching the current online slave's to the EtherCAT Master module. This is an alternate approach to using an EtherCAT Configurator, such as Beckhoff's, allowing the configuration to be dynamically created from within QuickBuilder. The actual creation and storage is performed by the EtherCAT Master module, thereby requiring no file transfer. Note that this operation takes about 20 seconds to complete, because of the length of time required to erase non-volatile memory. The appropriate Module # must be selected from within the tree list prior to pressing the **Create Config** button, or an error message will be displayed:



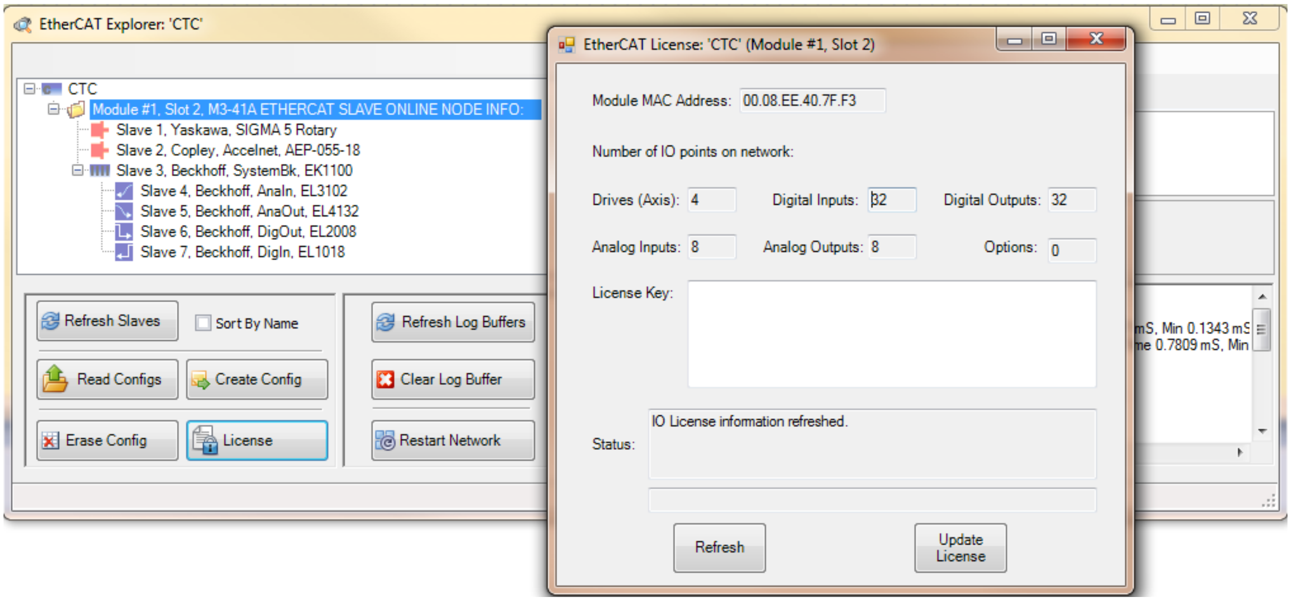
Please select a module before operation can be attempted.

Note that the PC runtime stores this information in a file called _slaveConfig_[MAC Address].txt located in the _system\Programs directory.

Configuration files must be used in a production environment to ensure all the required devices are online prior to executing their controller MSBs. Differing devices power up at different times and may not initially respond to the EtherCAT Master online broadcast. Having a configuration file to compare against informs the master that it must wait for devices to come online prior to proceeding with the boot operation.

Erase Config – This button erases the current configuration file stored in the selected EtherCAT Master Module's non-volatile memory. By default, when no file is stored, and the network is restarted, no verification of online slaves occurs, and the controller begins operation with whatever devices and I/O are found on the network. This is known as Slave Discovery Mode and is useful when initially setting up a network. It can take up to 20 seconds for this command to complete. The appropriate Module # must be selected from within the tree list prior to pressing the **Create Config** button or an error message will be displayed.

License – This button displays the EtherCAT Master License form for the selected module. The MAC Address of the module appears along with the type and number of devices authorized for control by the master. New license keys can be purchased from CTC technical support, and entered within this form to change the current authorization.



EtherCAT Explorer: 'CTC'

CTC
Module #1, Slot 2, M3-41A ETHERCAT SLAVE ONLINE NODE INFO:
Slave 1, Yaskawa, SIGMA 5 Rotary
Slave 2, Copley, Accelnet, AEP-055-18
Slave 3, Beckhoff, SystemBk, EK1100
Slave 4, Beckhoff, AnaIn, EL3102
Slave 5, Beckhoff, AnaOut, EL4132
Slave 6, Beckhoff, DigOut, EL2008
Slave 7, Beckhoff, DigIn, EL1018

mS, Min 0.1343 mS
me 0.7809 mS, Min

EtherCAT License: 'CTC' (Module #1, Slot 2)

Module MAC Address: 00.08.EE.40.7F.F3

Number of IO points on network:

Drives (Axis): 4 Digital Inputs: 32 Digital Outputs: 32

Analog Inputs: 8 Analog Outputs: 8 Options: 0

License Key:

Status: IO License information refreshed.

Copy the license key you receive by email and paste it into the 'License Key" text box. Click the **Update License** button to update the number of I/O authorized. Click the **Refresh** button to verify the changes have been made. You must reboot the controller for the changes to take effect at the network level. Note that the PC runtime stores this information in a file called _ioOptions_[MAC Address].txt located in the _system\Programs directory.

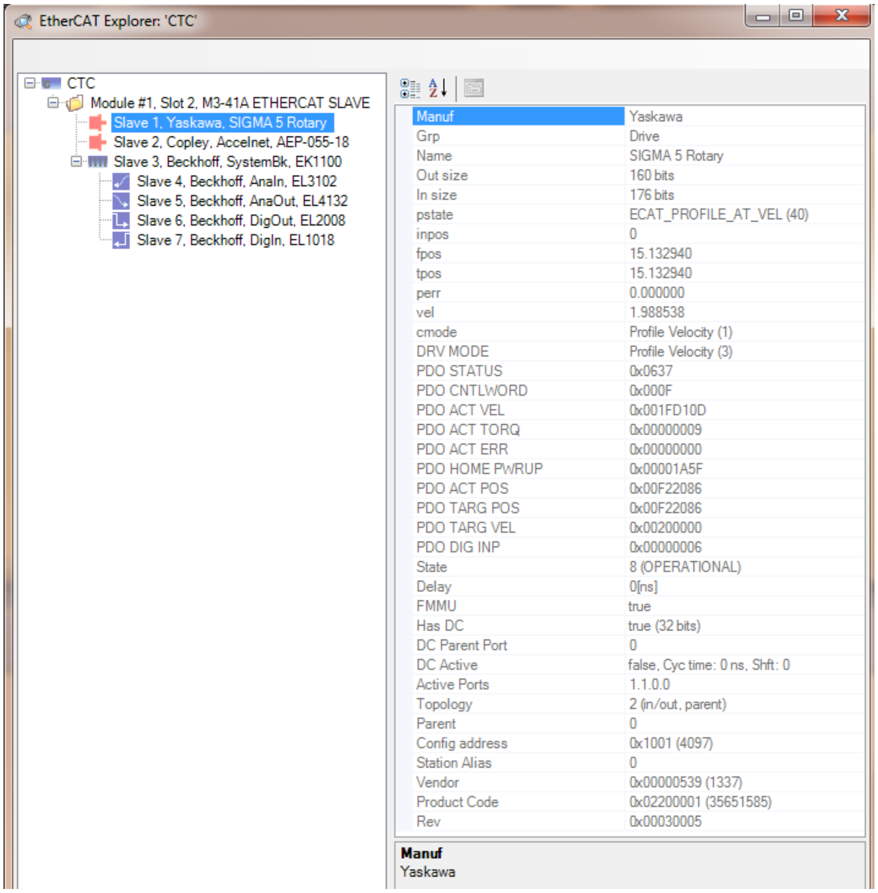
Refresh Log Buffers – This button displays log messages residing in all EtherCAT Master Modules in the lower right window. It is useful for diagnostic purposes.
Clear Log Buffer – This button clears the log messages for the selected EtherCAT Master module. Only new messages that occur after the clear operation will appear after the **Refresh Log Buffers** button is selected.

Restart Network – This button causes the selected EtherCAT Master module to re-scan the network and display whatever slaves are found. Prior scan results are overwritten. Available I/O in the controller will not be updated, and a reboot is required if the configuration changes. The network will not be available until after the restart is completed. Restarting the network is useful when connecting new slaves to the network or after power cycling a slave to verify that it is seen on the network. Note that MSBs will also restart.

Reboot Controller – This button causes the controller to be rebooted remotely. This is a hard reset and can take up to 30 seconds before the controller will be back online.

User Options – This button allows customization of the EtherCAT Master parameters, such as PDO cycle time, number of virtual drives, timeouts, and retries. Note that the PC runtime stores this information in a file called _options_[MAC Address].txt located in the _system\Programs directory.

EtherCAT Explorer Properties



EtherCAT Explorer: 'CTC'

- CTC
  - Module #1, Slot 2, M3-41A ETHERCAT SLAVE
    - Slave 1, Yaskawa, SIGMA 5 Rotary
    - Slave 2, Copley, Accelnet, AEP-055-18
    - Slave 3, Beckhoff, SystemBk, EK1100
      - Slave 4, Beckhoff, AnaIn, EL3102
      - Slave 5, Beckhoff, AnaOut, EL4132
      - Slave 6, Beckhoff, DigOut, EL2008
      - Slave 7, Beckhoff, DigIn, EL1018

Manuf	Yaskawa
Grp	Drive
Name	SIGMA 5 Rotary
Out size	160 bits
In size	176 bits
pstate	ECAT_PROFILE_AT_VEL (40)
inpos	0
fpos	15.132940
tpos	15.132940
perr	0.000000
vel	1.988538
cmode	Profile Velocity (1)
DRV MODE	Profile Velocity (3)
PDO STATUS	0x0637
PDO CNTLWORD	0x000F
PDO ACT VEL	0x001FD10D
PDO ACT TORQ	0x00000009
PDO ACT ERR	0x00000000
PDO HOME PWRUP	0x00001A5F
PDO ACT POS	0x00F22086
PDO TARG POS	0x00F22086
PDO TARG VEL	0x00200000
PDO DIG INP	0x00000006
State	8 (OPERATIONAL)
Delay	0[ns]
FMMU	true
Has DC	true (32 bits)
DC Parent Port	0
DC Active	false, Cyc time: 0 ns, Shft: 0
Active Ports	1.1.0.0
Topology	2 (in/out, parent)
Parent	0
Config address	0x1001 (4097)
Station Alias	0
Vendor	0x00000539 (1337)
Product Code	0x02200001 (35651585)
Rev	0x00030005

Manuf
Yaskawa

Manuf - Manufacturer description

Grp – Group description

Name – Device Name

Out size – Number of bits the device consumes in the output Ethernet packet

In size – Number of bits the device consumes in the input Ethernet packet

Program Variables (MSB variables, if a drive):
pstate – present MSB program operational state
inpos – 1 if motor is in position, 0 if it is not
fpos – Current motor position in revolutions
tpos – Present target position in revolutions
perr – Present error (tpos – fpos)
vel – Present motor velocity in revolutions/second
cmode – MSB program commanded mode for the motor

DRV MODE – CANOpen DS402 mode the drive is in for motion control

PDO STATUS – Object 0x6041 representing the device state

PDO CNTLWORD – Object 0x6040 representing the currently written Control Word

PDO ACT VEL – Object 0x606C representing the current velocity

PDO ACT TORQ – Object 0x6077 representing the current torque. On some drives this is the Actual Current when torque is not available.

PDO ACT ERR – Object 0x60F4 representing the current servo position error

PDO HOME PWRUP – Power up position first seen by the EtherCAT Master

PDO ACT POS – Object 0x6064 representing the actual current position in increments

PDO TARG POS – Object 0x607A representing the target position in increments. This is relevant in Cyclic Sync Position and Profile Position modes. Interpolated motion mode uses 0x60C1 subindex 1.

PDO TARG VEL - Object 0x60FF representing the target velocity in increments/sec. This is only relevant in Profile Velocity mode.

PDO DIG INP – Drive Inputs as reported by the cyclic PDO scan

State – Last seen EtherCAT state of this device

Delay – Propagation delay, in ns, of this device as cabled on the network.

FMMU – True if Fieldbus Memory Management unit is bit oriented, false if it is not.

Has DC – Set to 1 if the drive can support being the source of the distributed clock. The EtherCAT Master selects the first device that 'Has DC' as the source of the clock and then periodically reads the time from that device and writes it to the rest of the slaves. The Master attempts to sync its internal clock to that device as well.

DC Parent Port – The slave named as the Distributed Clock master

DC Active – Information relative to DC Sync, if enabled

Active Ports – Each device can typically have up to 4 ports. This represents the ports being used on this device.

Topology – Each EtherCAT slave has up to 4 internal ports, each represented by a bit.

Parent – Set to the parent node. 0 means it is the parent.

Config address – EtherCAT assigned device address representing its place with regards to being cabled on the network. First device would be 0x1001.

Station Alias – Programmable station alias used to define axis numbers for MSB assignment

Vendor – Device vendor code

Product Code – Device product code

Rev – Firmware revision of the device

Log Buffer Timings

When the Log Buffer is viewed the first line after the module identification information contains internal timing information. This information can be critical in troubleshooting problems or possibly preventing them. The timing information contains the state of the EtherCAT network scanning, control loop overhead, idle, and slave sync timings. Below was observed in a six axis system do simple back and forth motion on all drives:

```
*** Module #1, Slot 4 ***
M3-41A ETHERCAT MASTER
INFO: Time:     275.943, Scanning = 2, Cycle 1.0000 mS, [Overhead 0.3180 mS, Min 0.1580 mS, Max 0.3995 mS, Avg 0.2951 mS],
        Adjusted Tick 1.0000 mS, Correction -6 ns, Max Tick 1.0000 mS, [Idle 0.6640 mS, Min 0.5234 mS, Max 0.8462].
        [Sync Error 0.0002 mS, Min 0.0000 mS, Max 0.0730, Avg 0.0022].
```

The following is how to interpret these timings:

[Scanning = 2] – The EtherCAT network scanner has three possible states.

- Initializing, 0.
- Scanning but for initial sync, 1.
- Online and executing, 2.

[Cycle 1.0000 mS] – Network control loop scan time, typically 1 mS or 500 µS.

[Overhead] – The time needed to process the PDO packet from the slaves, calculate the new trajectories, update IO, and prepare the PDO packet for transmission. The Cycle time minus the Overhead time is how much time the rest of the system has to execute. The Overhead may never be greater than the Cycle time. If it occurs once in a while and is less than 50 µS a warning will result and recovery attempted, otherwise a fault error. The first time listed is the time for the last completed cycle.

[Adjusted Tick] – The time that the FPGA timer was last set to, on the last cycle. This will shift slightly to sync with the reference slave.

[Correction] – The amount of correction added to the last time cycle in order to more closely sync to the reference slave.

[Max Tick] – The maximum time that the FPGA timer was set to for its periodic interrupt.

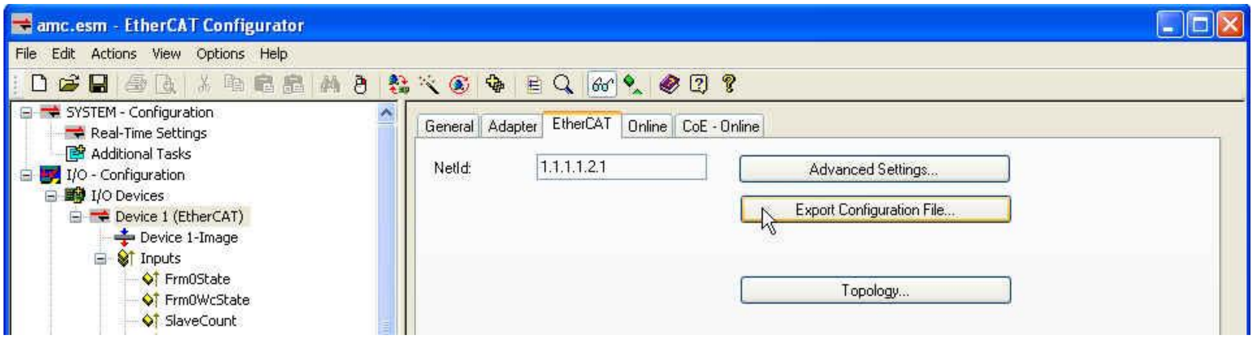
[Idle] – The amount of idle time available for the rest of the system to run, with the current time for the last completed cycle listed first.

[Sync Error] – This is the amount of error or jitter that the master has experienced while attempting to sync to the slave reference. The first listed time is from the last completed control loop cycle. Note that the average jitter is only 2.2 µS with a maximum of 73 µS. The 73 µS only lasts for a single cycle as corrections are applied.

EtherCAT Master ENI Configuration Files

Standard EtherCAT configuration files can be generated either automatically, via the **Create Config** button of the QuickBuilder EtherCAT Explorer, or through third-party tools such as Beckhoff’s TwinCAT or EtherCAT Configurator. The file format stored in the M3-41 is standard XML.

To generate a configuration file using Beckhoff’s EtherCAT Configurator, first create a network of the desired configuration and then select (as shown below) to export to an XML file (I/O-Configuration->I/O Devices->Device 1 (EtherCAT) followed by the EtherCAT tab and the **Export Configuration File** option button):



NetId: 1.1.1.1.2.1

The saved file must be renamed to M341ACFGV0100.xml and placed in the model 5300 controller's _system->Firmware directory. You may set the V#### part of the file name to anything you desire, but the first part must be M341ACFG. Use the standard firmware update commands to load the file into a module. 'update M341ACFGV0100.xml' for all modules in the rack, or 'force update slot # M341ACFGV0100.xml' for a specific slot. Note that the 'force update' command is also needed if the M3-41 module I/O/Drives are not online. The 'update' command only works when the card is fully operational and the I/O/Drives have been added by the controller.

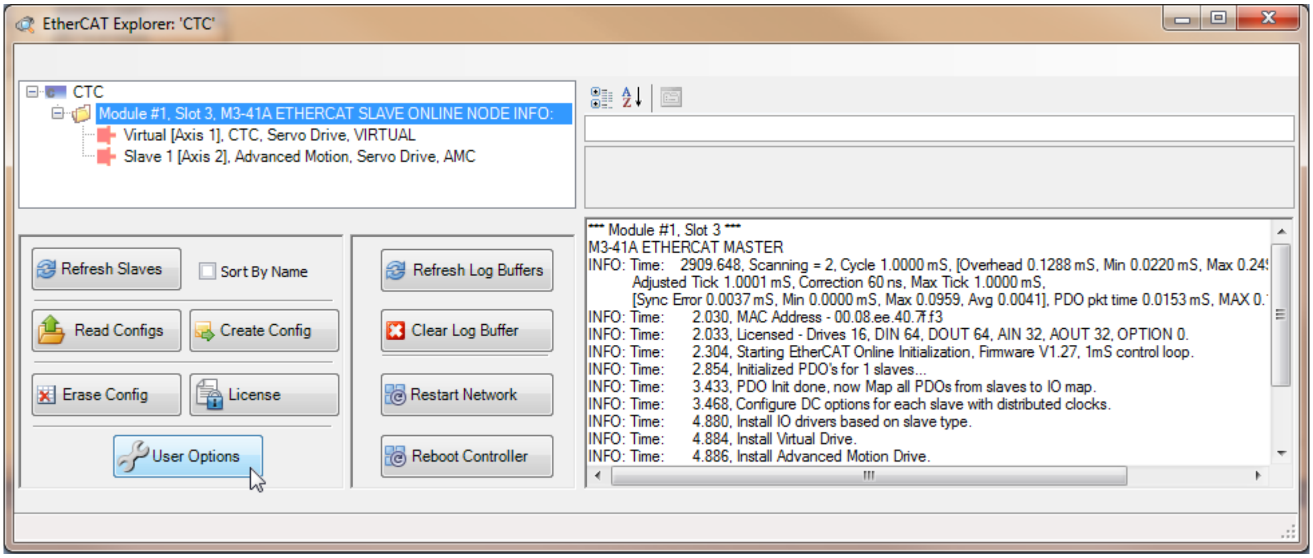
The configuration file size limit is 1,572,860 bytes. For larger configuration files, you must use QuickBuilder to create the configuration file. This is not supported on the PC Runtime version and is not recommended for general use. It is better to use the QuickBuilder 'Save Config' feature when the desired network is present.

User Options

The EtherCAT Explorer allows the user the ability to customize the EtherCAT Master's operation. The customization currently supported consists of:

- Master PDO cycle loop times of 500 µS, 1 mS, 2 mS, or 4 mS.
- Automatic virtual axis creation.
- Capability to add the virtual axis to the end or beginning of those drives online.
- Retry forever option.
- PDO Timeout & retries.
- Initialization retries.

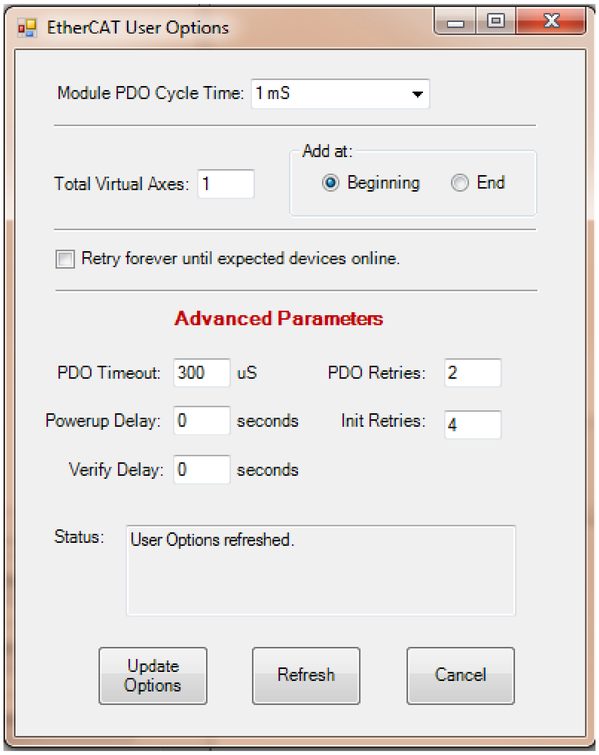
Invoking the 'User Option' form is done by clicking that button within the EtherCAT Explorer:



CTC
Module #1, Slot 3, M3-41A ETHERCAT SLAVE ONLINE NODE INFO:
Virtual [Axis 1], CTC, Servo Drive, VIRTUAL
Slave 1 [Axis 2], Advanced Motion, Servo Drive, AMC

```
*** Module #1, Slot 3 ***
M3-41A ETHERCAT MASTER
INFO: Time:   2909.648, Scanning = 2, Cycle 1.0000 mS, [Overhead 0.1288 mS, Min 0.0220 mS, Max 0.24
        Adjusted Tick 1.0001 mS, Correction 60 ns, Max Tick 1.0000 mS,
        [Sync Error 0.0037 mS, Min 0.0000 mS, Max 0.0959, Avg 0.0041], PDO pkt time 0.0153 mS, MAX 0.
INFO: Time:      2.030, MAC Address - 00.08.ee.40.7f.f3
INFO: Time:      2.033, Licensed - Drives 16, DIN 64, DOUT 64, AIN 32, AOUT 32, OPTION 0.
INFO: Time:      2.304, Starting EtherCAT Online Initialization, Firmware V1.27, 1mS control loop.
INFO: Time:      2.854, Initialized PDO's for 1 slaves...
INFO: Time:      3.433, PDO Init done, now Map all PDOs from slaves to IO map.
INFO: Time:      3.468, Configure DC options for each slave with distributed clocks.
INFO: Time:      4.880, Install IO drivers based on slave type.
INFO: Time:      4.884, Install Virtual Drive.
INFO: Time:      4.886, Install Advanced Motion Drive.
```

Once invoked the currently programmed options will appear:



EtherCAT User Options

Module PDO Cycle Time: 1 mS

Total Virtual Axes: 1

Add at: ● Beginning ○ End

□ Retry forever until expected devices online.

Advanced Parameters

PDO Timeout: 300 uS PDO Retries: 2

Powerup Delay: 0 seconds Init Retries: 4

Verify Delay: 0 seconds

Status: User Options refreshed.

Module PDO Cycle Time – This option sets the EtherCAT master control loop cycle time. The default of 1 mS is typically fine but in some situations the user may wish to speed up or slow down the loop. For example if the system is heavily loaded a 2 or 4 mS control loop will work well with most drives. Selections of 500 µS, 1 mS, 2 mS, and 4 mS are available.

Total Virtual Axes – This option sets the number of virtual axis to 'Add at' the 'Beginning' or 'End' of the online drive list. A virtual axis runs an MSB just like an online axis except that its feedback position (fpos) is updated automatically to its incremental tpos on each control loop, thereby simulating motion. The Virtual Axis is reported to the QuickBuilder as a normal axis.

Retry forever – This option, when selected, will cause the controller and EtherCAT network not to boot until the stored online configuration is observed. If the option is not selected then the 'Init Retries' parameter within the Advanced Parameters will be referenced and that many retries attempted prior to reporting the fault state to the controller and aborting operation.

PDO Timeout – This option should not be set unless instructed by CTC technical support. It will automatically be optimized to the proper setting based upon the PDO Cycle time selected. The option is the amount of time that the EtherCAT Master will wait for the response to the cyclical PDO packet transmission. PDO Timeout X PDO Retries should be less than the cycle time to ensure no DC Sync errors.

PDO Retries – This option should not be set unless instructed by CTC technical support. It will automatically be optimized to the proper setting based upon the PDO Cycle time selected. The option is the number of PDO Timeouts that are allowed before aborting operation and faulting. PDO Timeout X PDO Retries should be less than the cycle time to ensure no DC Sync errors.

Powerup Delay – This option defines how long, in seconds, the M3-41 module should wait, after power up, prior to beginning its identification of network slaves and initializing the EtherCAT network. It is useful when attempting to prevent timeouts on equipment that may take a long time to power up and come online.

Init Retries – This option is the number of times the EtherCAT Master will attempt to activate the network and initialize devices. If a Network Configuration is saved and those devices observed online do not match the network will be re-initialized and scanned again, with this count decremented by 1. Once a count of 0 is reached the module will abort, fault, and report an error.

Verify Delay – This option sets the amount of time, in seconds, the M3-41 module should delay after it identifies all required slave devices online (INIT state) and initializes their PDO's (PRE_OP state). After the delay occurs one more INIT cycle will be done and delay prior to updating the PDO mappings within the slave and marking the devices as online. This is required in some networks where slaves can report that they are online and ready but in fact other equipment is powering up or the slave still needs a small amount of time to continue initialization.

Available buttons:

Update Options – Clicking this button will cause the module to be set to the settings currently displayed. The status window will display the results of the operation.

Refresh – Clicking this button will cause the form to be updated with the options currently programmed within the module.

Cancel – Clicking this button will close the form, without changes and return to the EtherCAT Explorer form.

Blank

CHAPTER

9

[9] Telnet Commands

The model 5300 has an administrative mode that can be accessed via standard telnet. The use of telnet is beyond the scope of this manual and discussion here is limited to the specific commands available that affect the M3-41 EtherCAT module. Many of the commands are the same as those used by the QuickBuilder EtherCAT Explorer.

Status Commands

***get ethercat info all* –** Displays all available information for online slave devices, including mailbox information

***get ethercat info summary* –** Displays all available information for online slave devices, less the mailbox information. This is what is used by the QuickBuilder EtherCAT Explorer.

***get ethercat slave coe <slave #>* -** Displays the current PDO mapping and mailbox information for a specific slave

Network Commands

***restart ethercat networks* –** Restarts all EtherCAT networks. I/O and drives should remain the same. If changes are made to any network devices, cycle power rather than using this command. Otherwise the new modules will not be usable by the controller or a configuration error could occur.

***restart ethercat network slot <slot #>* –** This command will restart only the EtherCAT network for the M3-41 module in the specified slot, I/O and drives should remain the same. If changes are made to any network devices cycle power rather than using this command. Otherwise the new modules will not be usable by the controller or a configuration error could occur.

***get ethercat mac address* slot <slot # or -1 for first> -** Retrieves the MAC Address of the EtherCAT Master module

Message Log Commands

***clear ethercat messages* –** Erases the message log buffer on all M3-41 EtherCAT modules

***clear ethercat messages slot <slot # >* –** Erases the message log buffer for only the M3-41 EtherCAT module in the specified slot, with 1 being the first

***get ethercat messages* –** Displays all the message log buffers of all M3-41 EtherCAT modules installed.

Configuration File Commands

***erase ethercat config files* –** Erases the configuration files stored in the M3-41 modules. If the controller is rebooted, the network will be used without verification.

***erase ethercat config file slot <slot #>* –** Erases the configuration file only in the M3-41 module at the specified slot, with 1 being the first. If the controller is rebooted, the network will be used without verification.

***generate ethercat config files* –** Generates an XML file of the existing online slaves and stores it to serial flash memory of all M3-41 modules. Upon re-start, this file will be read and used to verify the slave devices found on the network. Operation will not begin until a full match is found. The use of this command greatly simplifies using a third-party configurator: connect all the slaves to the EtherCAT Master; confirm they are present with the 'get ethercat info summary' command; and use this command to save the configuration.

***generate ethercat config file slot <slot #>* –** Generates an XML file of the existing online slaves and stores it to serial flash memory for only the M3-41 module in the specified slot. Upon re-start, this file will be read and used to verify the slave devices found on the network. Operation will not begin until a full match is found. The use of this command greatly simplifies using a third-party configurator: connect all the slaves to the EtherCAT Master; confirm they are present with the 'get ethercat info summary' command; and use this command to save the configuration.

get ethercat expected info – Displays any saved configuration on the M3-41 module. It will be displayed in the 'get ethercat info summary' format containing information such as the VendorID, Product Code, etc.

Firmware Update Commands

***update <Filename>* -** The 'update' command is the standard command supported by the model 5300 to update online firmware modules. The file should reside in the /_system/Firmware directory and that should be made current using the change directory command, 'cd'. The M3-41 module supports two different files: an .xml used as a master configuration file, as exported from a program such as TwinCat, and a .bin file to update the module firmware. The file naming convention is fixed and must use the following format:

> M341ACFGV0100.xml – used as the configuration file. The V0100 represents the version number and can be anything desired.
>
> M341ASOMV0100.bin – used as the module firmware file. The V0100 represents the version number and can be anything desired.

***fupdate slot <#> <Filename>* -** The 'fupdate' command is the standard command supported by the model 5300 to update firmware modules that are offline or online. It is considered a forced update to a specific slot, regardless of module type. The <#> represents the slot number, starting at 1, with the <Filename> convention the same as for the 'update' command.

'update' will only work if the module is fully operational and devices are online. If not online and the controller is faulted, use the 'fupdate' command.

The equivalent of 'update' will also occur if a file is dragged and dropped onto the root 5300 directory via Internet Explorer.

License Commands

***get ethercat IO enabled slot <slot # or -1 for first>* -** Used to display the current licensed I/O totals authorized for use on the EtherCAT Master module.

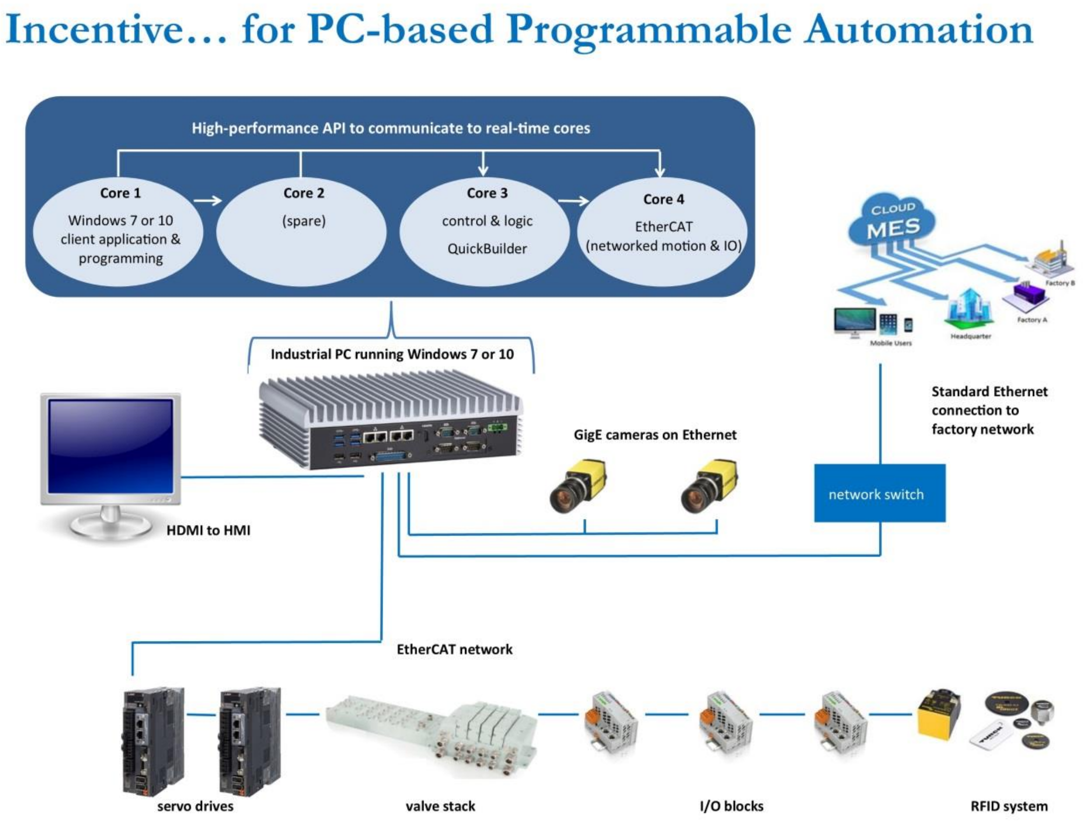
***set ethercat IO enabled slot <slot # or -1 for first> <Encrypted Key>* -** Used to set a new authorization license total within the EtherCAT Master module.

Blank

CHAPTER 10

[10] Incentive PC Runtime

The 5300 Controller QuickBuilder run-time has been ported to the Windows® PC where all your EtherCAT programs can be used on both environments with just the selection of a translation combo box within QuickBuilder. A virtual, soft 5300 PLC, can now execute on multiple platforms allowing for versatility in your automation decisions. In addition to QuickBuilder a complete Incentive .Net Managed API is available allowing for programming of all functions directly from languages such as C#, VB.Net, and C++ using Visual Studio®. The Incentive API works both on the local computer and transparently over a network. Complex, highly integrated solutions can now be integrated into a single platform, programmed in your language of choice.


Incentive… for PC-based Programmable Automation
High-performance API to communicate to real-time cores
Core 1
Windows 7 or 10 client application & programming
Core 2
(spare)
Core 3
control & logic
QuickBuilder
Core 4
EtherCAT (networked motion & IO)
Cloud MES
Factory B
Factory A
Headquarter
Mobile Users
Industrial PC running Windows 7 or 10
GigE cameras on Ethernet
Standard Ethernet connection to factory network
network switch
HDMI to HMI
EtherCAT network
servo drives
valve stack
I/O blocks
RFID system

Incentive Runtime

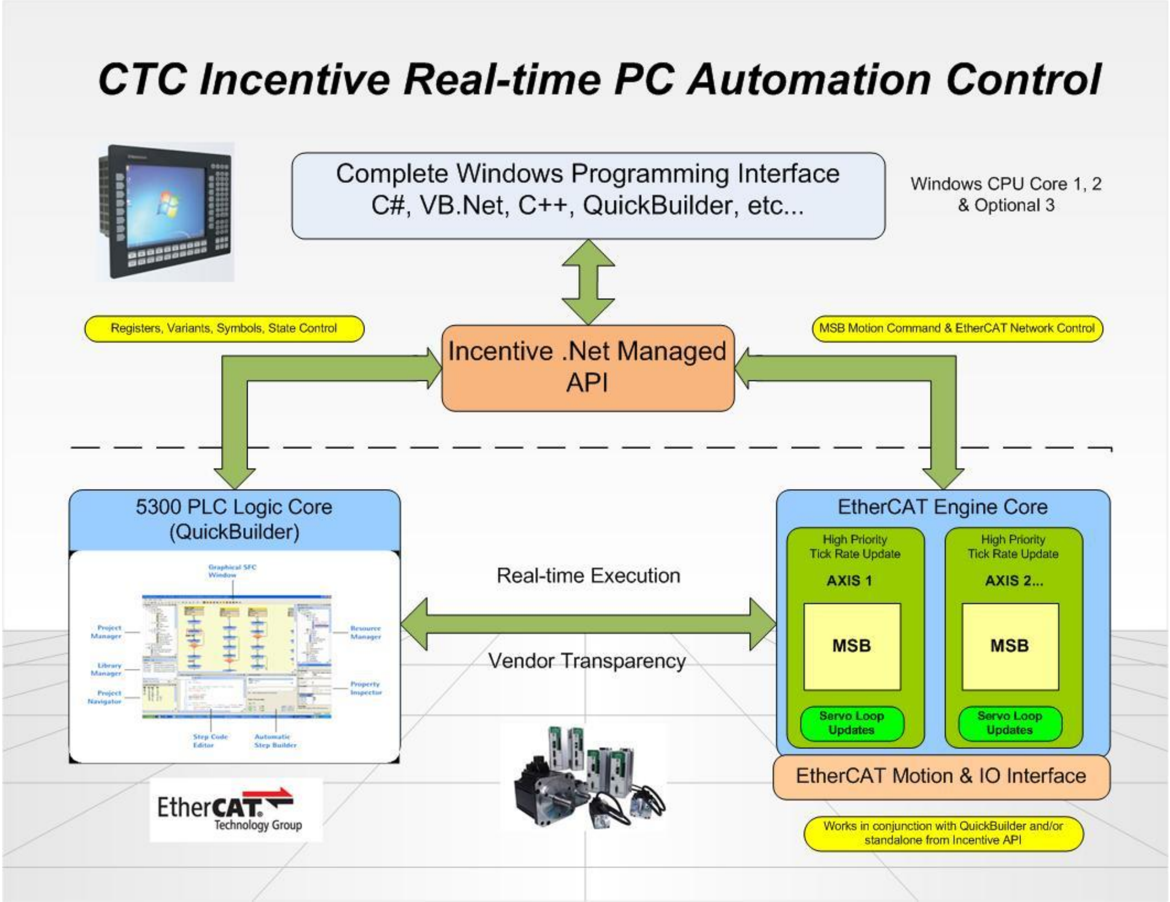
CTC Incentive offers system flexibility not only in its open vendor support of numerous EtherCAT drives and IO selections but its tightly integrated programming offerings, fully supporting Microsoft Visual Studio®, and transforming Windows® into a real-time automation environment. Standard Windows® based programs such as HMI's, vision systems, and custom applications run transparently, in parallel to the Incentive real-time EtherCAT Master.

The Incentive Runtime can run on systems with:

- Windows® 7 64 bit or greater, 32 bit is not supported. Windows 10 Professional preferred.
- Dual or Quad core processors such as J1900, E3845, i5, i7. Quad core is recommended although dual core processors may be used in smaller, cost sensitive environments with reduced performance. 4 G memory minimum, 8G-16G recommended for efficient Windows® operation.
- Some processors may have jitter issues and need to be evaluated by CTC. Currently Intel processor families such as Bay Trail, Haswell, and Broadwell work fine. Skylake does not work on some systems due to multi-milliseconds cache pollution. The exceptions are those with the Q170 chipset. Kaby Lake currently has a 600uS intermittent jitter issue that Intel is aware of and may be resolved but currently should not be used in systems with more than 4 axis, if at all. CTC is continually evaluating systems and has partnered with Axiomtek to ensure stability.
- Solid state drives must be used to ensure data integrity of registers during shutdown as well as support the fast access time required by the real time environment. The Samsung EVO and PRO ssd drives are recommended.
- BIOS with limited System Management Interrupt usage. SMI's can cause excessive jitter when not using an I210 Ethernet Adapter and reduce performance. They can prevent the real-time operating system from gaining control for several hundred microseconds. Refer to the section within this chapter entitled "System Management Interrupt Detection" for additional information.
- Intel I210 Ethernet Adapter preferred (Realtek 8111E PCIe Gbe, i211 and i219 family adapters with reduced performance). Realtek 8168E did not pass EtherCAT performance testing. Other adapters are possible but subject to testing.
- Onboard Graphics controllers such as Intel HD typically offer better timing stability than Nvidia or AMD.
- The BIOS of the PC must have controls to disable such things as SpeedStep, Hyper-Threading and C-States. Failure to do so can cause excessive jitter and possible EtherCAT errors. In many cases it is less expensive, with similar performance, to use an i5 rather than an i7 since i5 is essentially an i7 with no Hyper-Threading and a smaller cache. Secure boot must also be disabled.
- A small UPS with Windows UPS shutdown utilities is suggested for register preservation upon power loss. Both CyberPower and APC have been tested to work well.

To optimize performance the Windows® processor can run different operating systems on each of its cores. With CTC Incentive, in a quad core system, one core runs the QuickBuilder PLC Logic; a second core runs a virtual EtherCAT Master, all communicating through high speed shared memory. Both of these cores run in real-time using an operating system provided by TenAsys, called INtime. This leaves the remaining two cores to run any normal Windows® application. A second EtherCAT Master can also be run for larger systems, where two networks are desired, using 3 cores in real-time and leaving one core for Windows®. For even greater flexibility the EtherCAT Master can be run standalone programmed solely by a Windows Application (C#, VB.Net, C++., etc…), using a single core, leaving 3 cores for greater Windows performance.

In a dual core environment, both the QuickBuilder application and EtherCAT Master are loaded as separate processes and execute on the same core. This should only be used in smaller systems where performance is not as important given the QuickBuilder application will affect the EtherCAT performance, and vice versa. Quad core allows each to run at maximum speed, independently, neither affecting the other. Where Windows® performance is more important the dual core environment can also be used on a quad core processor. In this case Windows® would run 3 cores for itself and one for QuickBuilder/EtherCAT.



A significant improvement with the Incentive PC Runtime is that of performance. For raw QuickBuilder execution the PC can be tailored to improve performance, especially in EtherCAT IO update rates. A simple itineration test provided some comparison numbers showing a quad core J1900 Celeron 2GHZ 8GB ram, at 2.3 X that of a 5300 and an i7-3770 3.40GHZ 16GB ram, 5.3 X. Much of the IO speed improvement is from the tightly coupled independent cores, each running their own environments, communicating through shared memory. One dedicated to QuickBuilder execution and the other to EtherCAT IO and motion control.

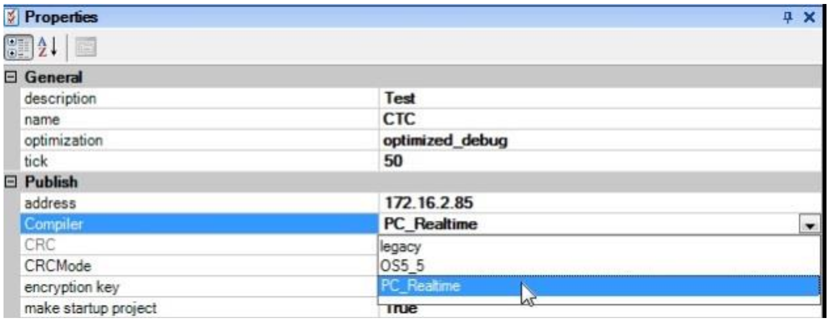
With the increased performance of the Incentive Runtime it also now supports up to 64 axes per network when the I210 network controller is used at a 1 ms scan rate as compared to that of 16 per M3-41 hardware version. For large networks segmentation is also supported, where data can be chained in multiple packets to allow for a greater number of IO and axes. With this also comes the importance of good cabling practices as, unlike the M3-41, the PC cannot do timely packet re-transmission and still maintain DC SYNC. A final improvement is the support of twice the number of RFID channels, 32, versus 16 of the M3-41. In summary:

Feature	5300 M3-41 Hardware	Incentive Soft M3-41 Dual Core	Incentive Soft M3-41 Quad Core
Max Coordinated Axes/Network	16	8 (Atom) 16 (i7)	64 (I210) 32 (other)
Max Networks	4	1	2
QuickBuilder Performance	1 X	1X to 4 X	2.3 to 8 X
EtherCAT Segmentation	No	Yes	Yes
Fast Packet Retry on Loss	Yes	No	No
Turck RFID Channels	16	8 (Atom) 16 (i7)	32
Scan Rates	500us-4ms	1ms-4ms	500us-4ms

Currently 500uS, 1ms and 2ms scan rates work on all devices. 1ms is the recommended default. 4ms does not work on Mitsubishi. Remember you must set the DC SYNC SYNC0 timing to match the scan rate or an error from the drive may occur. When using a non-I210 Ethernet Adapter it may best to offset the SYNC0 by up to 500uS (dcsync -1, 1000000, 0, 500000, 100000000).

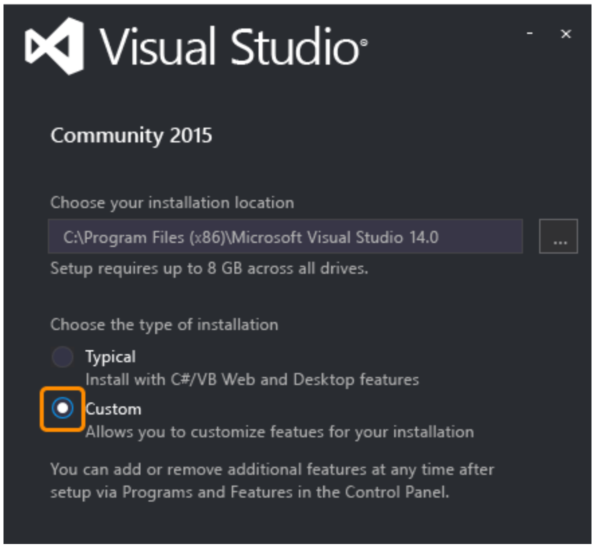
QuickBuilder Programming for Windows®

QuickBuilder generates 'C' code which can execute on multiple environments. Programming for the 5300 Controller is identical to that of the PC, you simply have to select 'PC_Runtime' from the Controller Compiler property and click Translate.



General	
description	Test
name	CTC
optimization	optimized_debug
tick	50
Publish	
address	172.16.2.85
Compiler	PC_Realtime
CRC	legacy
CRCMode	OS5_5
encryption key	PC_Realtime
make startup project	True

This assumes that you have previously installed a copy of Microsoft's Visual Studio® Express for Windows Desktop, or Community Edition, and the latest QuickBuilder Support libraries. Any Visual Studio® version greater than Visual Studio® 2012 can be used, as long as it has Desktop support and C++, this includes the free 2015 Community version (https://www.visualstudio.com/products/visual-studio-community-vs). Just make sure to do the custom install if installing 2015 or greater since Microsoft stopped installing C++ automatically beginning with that revision. The C++ compiler is required for QuickBuilder.



Community 2015

Choose your installation location

C:\Program Files (x86)\Microsoft Visual Studio 14.0

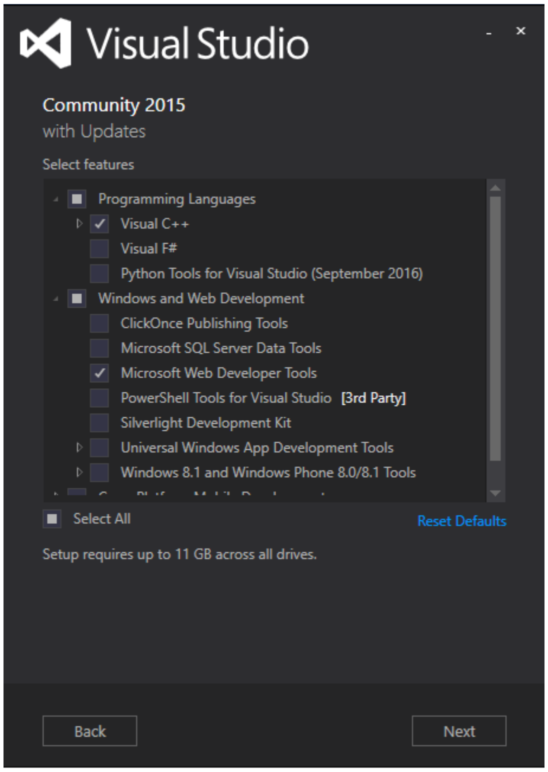
Setup requires up to 8 GB across all drives.

Choose the type of installation

○ Typical
Install with C#/VB Web and Desktop features

● Custom
Allows you to customize featues for your installation

You can add or remove additional features at any time after setup via Programs and Features in the Control Panel.



Community 2015
with Updates

Select features

- ■ Programming Languages
  - ☑ Visual C++
  - □ Visual F#
  - □ Python Tools for Visual Studio (September 2016)
- ■ Windows and Web Development
  - □ ClickOnce Publishing Tools
  - □ Microsoft SQL Server Data Tools
  - ☑ Microsoft Web Developer Tools
  - □ PowerShell Tools for Visual Studio [3rd Party]
  - □ Silverlight Development Kit
  - □ Universal Windows App Development Tools
  - □ Windows 8.1 and Windows Phone 8.0/8.1 Tools

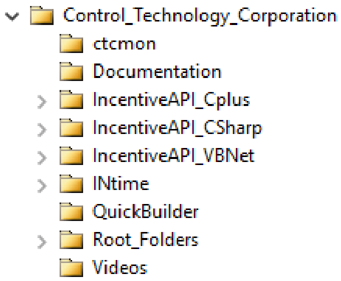
■ Select All

Setup requires up to 11 GB across all drives.

When ‘Select features’ appears expand ‘Programming Languages’ and make sure to select C++ as well as your desired languages if using the Incentive managed API.

Incentive Installation Overview

A single installation file, Incentive_Setup.msi, is available which provides the Incentive runtime, API, documentation, QuickBuilder, CTCMon, test projects, and installation video. Incentive_Setup.msi installs and registers the CTC_Incentive.dll, managed .Net dll, as well as provides additional folders within which are other programs to be installed. Upon installation the directory tree will look something like:



- Control_Technology_Corporation
  - ctcmon
  - Documentation
  - IncentiveAPI_Cplus
  - IncentiveAPI_CSharp
  - IncentiveAPI_VBNet
  - INtime
  - QuickBuilder
  - Root_Folders
  - Videos

ctcmon – Contains an installation program called *mon##setup.exe* (where ## is the version). This program will install a communications dll called ctccom32.dll as well as CTCmon. The dll may be used by any Windows program to interact with a controller on a network or serial port. The utility is used to monitor registers, program status, and IO via TCP, UDP, and serial ports.

Documentation – This manual in PDF format as well as the Incentive API chm help file.

IncentiveAPI_Cplus – A Visual Studio 2013 sample project using C++ with the Incentive DLL.

IncentiveAPI_CSharp - A Visual Studio 2013 sample project using C# with the Incentive DLL.

IncentiveAPI_VBNet - A Visual Studio 2013 sample project using VB.Net with the Incentive DLL.

INtime – Contains the installation programs for both a host only (remote PC with no resident runtime), *host61-17004_installer.exe* (where the numerics are the revision level), as well as the realtime runtime environment required by the EtherCAT Master, *runtime61-17004_installer.exe*. By default the runtime will install as a demo for 60 days. No license is required for the host environment. Patch files are also included and are updated as problems are identified and resolved. An older installer, *runtime61-16250_installer.exe* may also be used and requires no patching. Reference the readme in the INtime_Patches sub-directory for current patch requirements.

QuickBuilder –

- *QuickBuilder_Support.msi* installation program which contains the gcc compiler tools required for translating application programs for the 5300 embedded PLC. Note that Visual Studio 2015 Community or later is required to translate programs for execution on the PC realtime environment (available directly from Microsoft).
- *QuickBuilder_Setup.msi* installs the Quickbuilder development system which by default will install as a demo for 30 days.
- *ECAT_SimpleTurn_DCSYNC1.qbp* is a simple EtherCAT application program to control a single servo motor.
- *QuickBuilder_Windows_8_10_Installation.pdf* contains special instruction required for installation of QuickBuilder on that platform.

Root_Folders – Contain three subfolders, *_system*, *5300PC*, and *ramdisk*, all of which should be copied to the root directory of the C: drive. The *_system* and *ramdisk* folder contents replicate that of the 5300 embedded PLC while the *5300PC* subdirectory contains the executable files for both the PLC Logic and EtherCAT Master realtime Windows processor cores.

Videos – This folder contains the video ***Incentive_Runtime_Installation.mp4***. This video provides information on how to install and setup the Incentive environment.

The recommended sequence for installation is as follows:

Application Development Only or Remote PC with no EtherCAT Master:

1. Install CTCmon – mon##setup.exe, optional for debug.
2. Install QuickBuilder_Support.msi, required for QuickBuilder programming.
3. Install QuickBuilder_Setup.msi, required for QuickBuilder programming.
4. Download and install Visual Studio 2015 or greater Community edition from Microsoft. (https://www.visualstudio.com/downloads/)
5. Install host61-17004_installer.exe (where the numerics are the revision level).

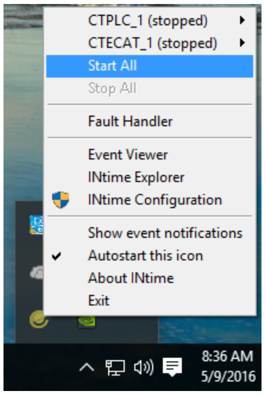
Realtime Incentive EtherCAT Master PC:

1. Perform all Windows updates prior to installation.
2. Copy each folder found in 'Root_Folders' to the root of the C: drive.
3. Install runtime61-17004_installer.exe (where the numerics are the revision level).
4. Watch Incentive_Runtime_Installation.mp4 video and configure accordingly.
5. If development will be done on the same PC as the Incentive EtherCAT Master then do installation steps 1 to 4 listed for Application Development. Installation of the host INtime environment is not needed since it is included with the runtime. Steps 1 to 4 for Application Development may be done before or after installation of the runtime environment.

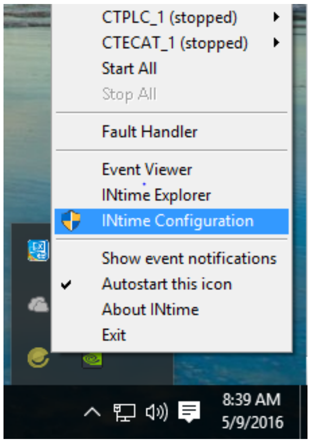
Reference the prior sections for additional configuration and licensing issues. By default the system will run in demo mode and licenses need to be obtained for QuickBuilder, Incentive API, and the EtherCAT_Master. The Incentive API is an option available under the EtherCAT Master license.

Startup and Network Configuration

Upon installation the Windows® PC appears and operates identical to a normal PC. The EtherCAT environment can be setup to run automatically when the PC boots or started/stopped manually by using a tray icon. The environment should only be started when the EtherCAT devices are ready for operation otherwise it will timeout and require a restart, just like the hardware based 5300 controller.

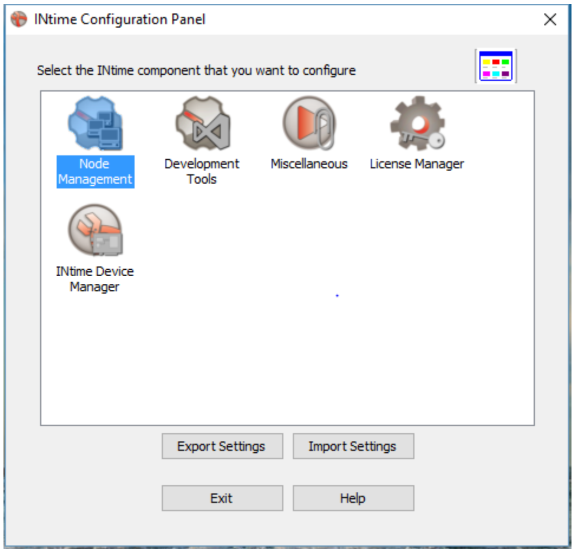


To modify the properties of the 5300 real-time environment you may invoke the INtime Configuration Manager.

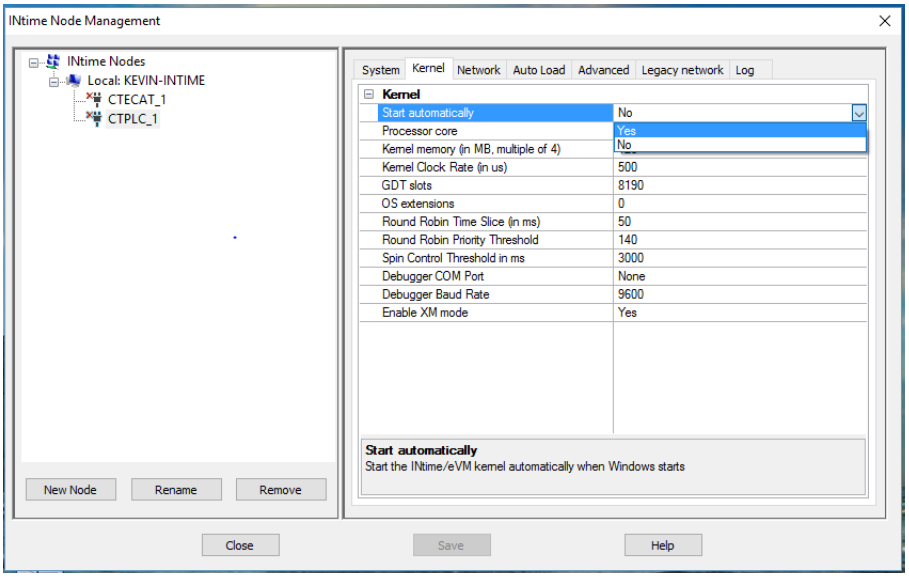


If for some reason the icon for INtime does not appear in the system tray, or INtime was installed by another user, simply run 'intimestatus' from a cmd window or located at "C:\Program Files (x86)\INtime\bin". Nothing will appear to happen but the icon will be in the system tray now. Click the "Autostart this icon" for it to appear upon each login of this user. Note that it can take several seconds after logging in for the icon to appear, depending upon the speed of your processor and other programs run by your startup menu.

Select 'Node Management':



If you wish to have the Incentive EtherCAT environment start automatically with Windows® you can select 'Yes' to the 'Start Automatically' prompt within the INtime Node Management screen. This must be done for both CTPLC_1 and CTECAT_1 in a quad core system, CTECAT_1 only with dual core.

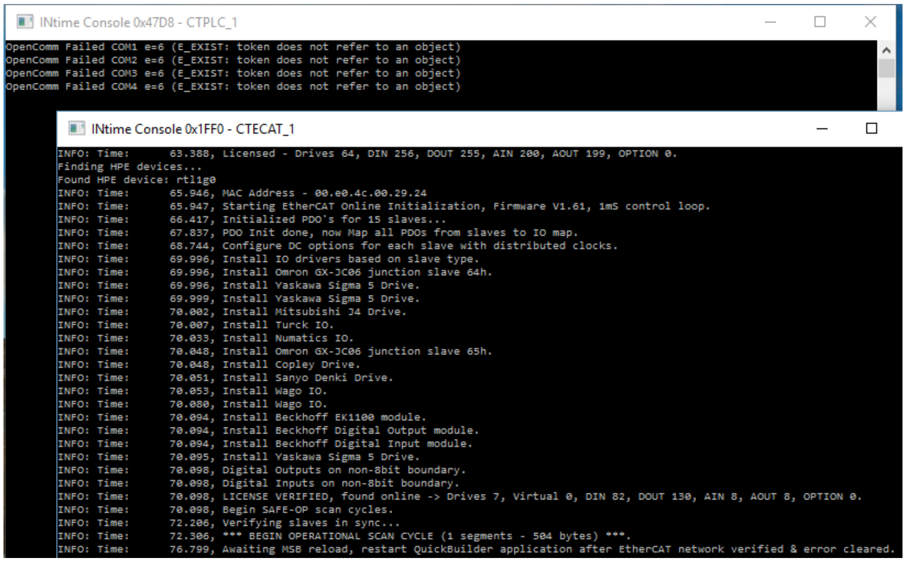


Kernel

Start automatically	No
Processor core	
Kernel memory (in MB, multiple of 4)	
Kernel Clock Rate (in us)	500
GDT slots	8190
OS extensions	0
Round Robin Time Slice (in ms)	50
Round Robin Priority Threshold	140
Spin Control Threshold in ms	3000
Debugger COM Port	None
Debugger Baud Rate	9600
Enable XM mode	Yes

Start automatically
Start the INtime/eVM kernel automatically when Windows starts

When started, Incentive will open up to two console screens, one for the EtherCAT core and one for the QuickBuilder PLC Logic core (non-automatic mode); expect several seconds of delay, especially if the optional LCD display is used. Diagnostic messages will be presented to these screens identical to what is typically viewed via the EtherCAT Explorer log, remotely. Note the "OpenComm Failed" errors, that is normal for each COM port not found and is meant to notify you that if you intend to use a serial port, none was found.


INtime Console 0x47D8 - CTPLC_1

```
OpenComm Failed COM1 e=6 (E_EXIST: token does not refer to an object)
OpenComm Failed COM2 e=6 (E_EXIST: token does not refer to an object)
OpenComm Failed COM3 e=6 (E_EXIST: token does not refer to an object)
OpenComm Failed COM4 e=6 (E_EXIST: token does not refer to an object)
```

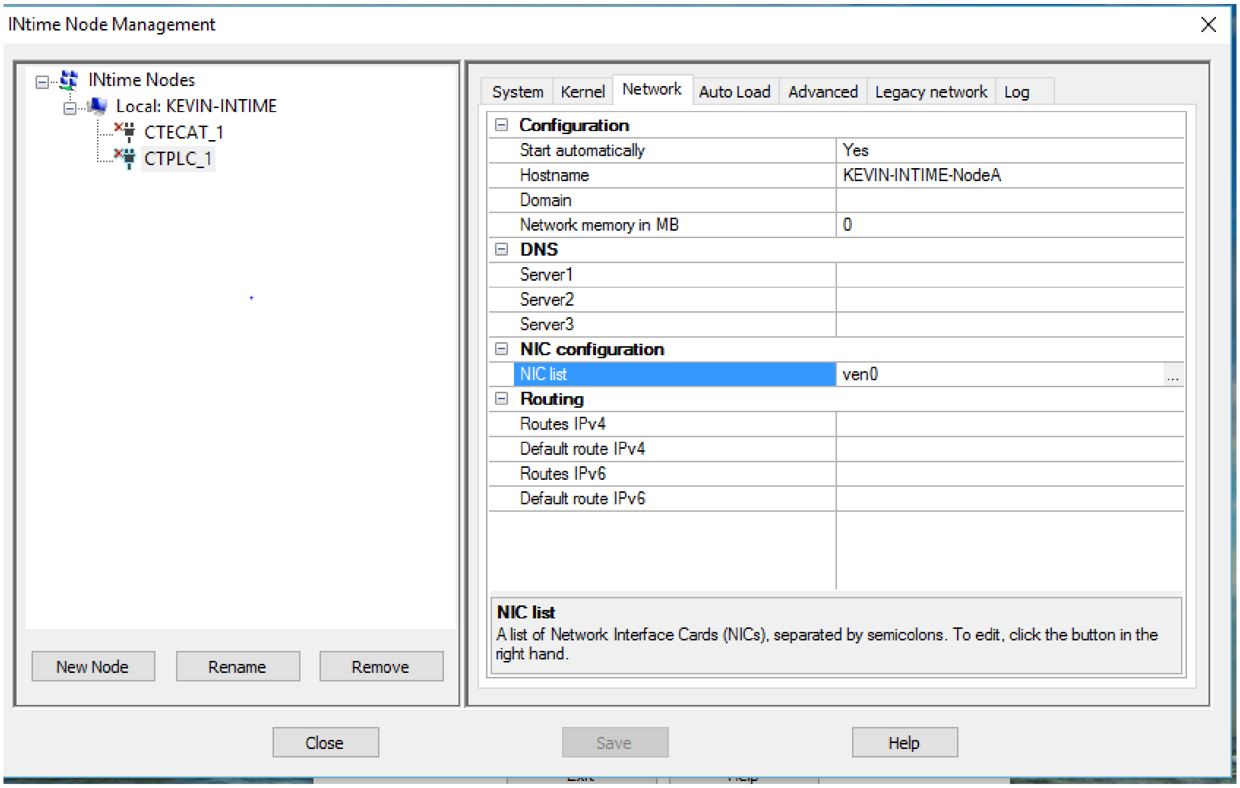
INtime Console 0x1FF0 - CTECAT_1

```
INFO: Time:      63.388, Licensed - Drives 64, DIN 256, DOUT 255, AIN 200, AOUT 199, OPTION 0.
Finding HPE devices...
Found HPE device: rtl1g0
INFO: Time:      65.946, MAC Address - 00.e0.4c.00.29.24
INFO: Time:      65.947, Starting EtherCAT Online Initialization, Firmware V1.61, 1mS control loop.
INFO: Time:      66.417, Initialized PDO's for 15 slaves...
INFO: Time:      67.837, PDO Init done, now Map all PDOs from slaves to IO map.
INFO: Time:      68.744, Configure DC options for each slave with distributed clocks.
INFO: Time:      69.996, Install IO drivers based on slave type.
INFO: Time:      69.996, Install Omron GX-JC06 junction slave 64h.
INFO: Time:      69.996, Install Yaskawa Sigma 5 Drive.
INFO: Time:      69.999, Install Yaskawa Sigma 5 Drive.
INFO: Time:      70.002, Install Mitsubishi J4 Drive.
INFO: Time:      70.007, Install Turck IO.
INFO: Time:      70.033, Install Numatics IO.
INFO: Time:      70.048, Install Omron GX-JC06 junction slave 65h.
INFO: Time:      70.048, Install Copley Drive.
INFO: Time:      70.051, Install Sanyo Denki Drive.
INFO: Time:      70.053, Install Wago IO.
INFO: Time:      70.080, Install Wago IO.
INFO: Time:      70.094, Install Beckhoff EK1100 module.
INFO: Time:      70.094, Install Beckhoff Digital Output module.
INFO: Time:      70.094, Install Beckhoff Digital Input module.
INFO: Time:      70.095, Install Yaskawa Sigma 5 Drive.
INFO: Time:      70.098, Digital Outputs on non-8bit boundary.
INFO: Time:      70.098, Digital Inputs on non-8bit boundary.
INFO: Time:      70.098, LICENSE VERIFIED, found online -> Drives 7, Virtual 0, DIN 82, DOUT 130, AIN 8, AOUT 8, OPTION 0.
INFO: Time:      70.098, Begin SAFE-OP scan cycles.
INFO: Time:      72.206, Verifying slaves in sync...
INFO: Time:      72.306, *** BEGIN OPERATIONAL SCAN CYCLE (1 segments - 504 bytes) ***.
INFO: Time:      76.799, Awaiting MSB reload, restart QuickBuilder application after EtherCAT network verified & error cleared.
```

The QuickBuilder PLC Logic core shares the main PC Ethernet adapter (bridged) and requires its own IP address. This is the IP address which all the communication protocols will use and it is different than the main Windows® PC even though using the same network adapter. You may assign this dynamically with

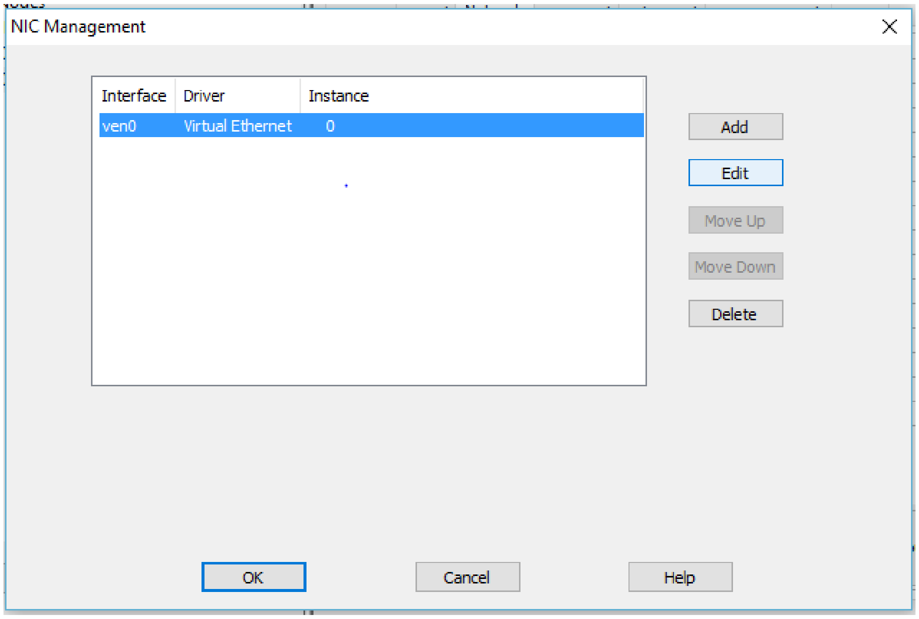
DHCP or use a fixed static IP address, depending upon your network. The network can start automatically, as shown below set to 'Yes', or let Incentive do it, set to 'No'. 'No' is the recommended selection with the benefit of having Incentive do it being a warning message will appear on the IO console that it can take up to 30 seconds to resolve DHCP, giving the usual visual feedback that things are running. During this time there is no activity since Incentive does not have control therefore if set to 'Yes' the user can be left with a blank Windows screen, or none at all, wondering if things have started.

The 'Hostname' is that which is registered with the DHCP server when requesting an IP address and/or used by a remote Incentive API to address the computer on a network. The host path for the API would use 'KEVIN-INTIME-NodeA' to open a connection to the node defined below.



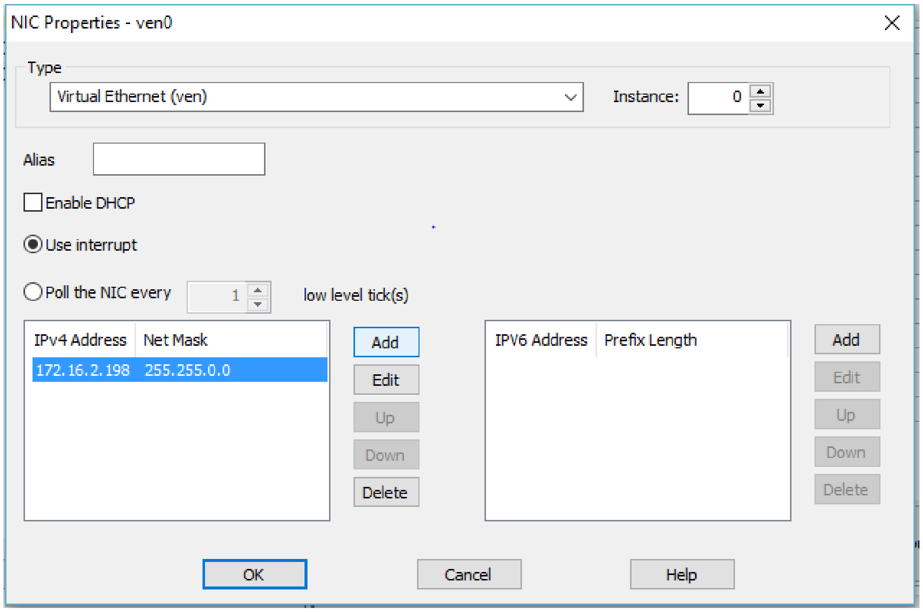
Configuration	
Start automatically	Yes
Hostname	KEVIN-INTIME-NodeA
Domain	
Network memory in MB	0
DNS	
Server1	
Server2	
Server3	
NIC configuration	
NIC list	ven0
Routing	
Routes IPv4	
Default route IPv4	
Routes IPv6	
Default route IPv6	

NIC list
A list of Network Interface Cards (NICs), separated by semicolons. To edit, click the button in the right hand.



NIC Management

Interface	Driver	Instance
ven0	Virtual Ethernet	0



NIC Properties - ven0

Type: Virtual Ethernet (ven)   Instance: 0

Alias

☐ Enable DHCP

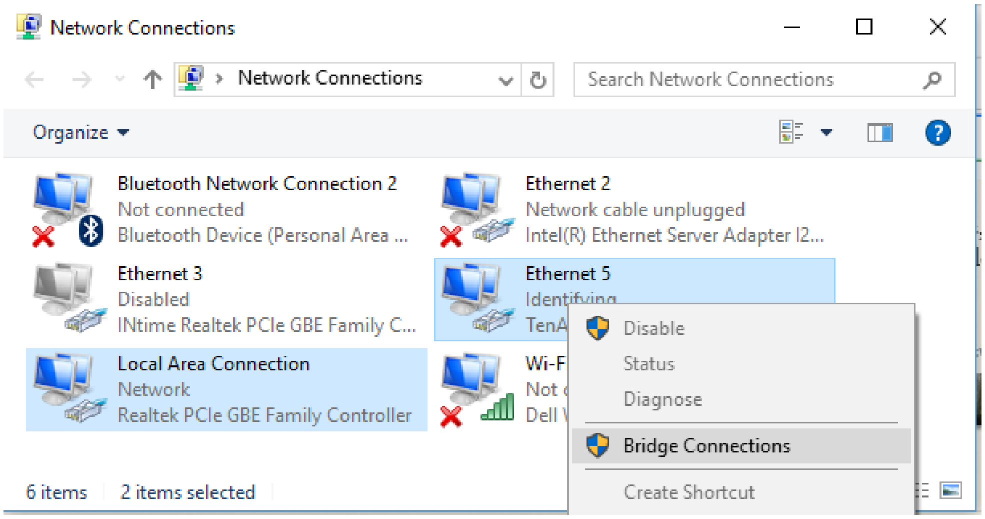
● Use interrupt

○ Poll the NIC every 1 low level tick(s)

IPv4 Address	Net Mask
172.16.2.198	255.255.0.0

IPV6 Address	Prefix Length

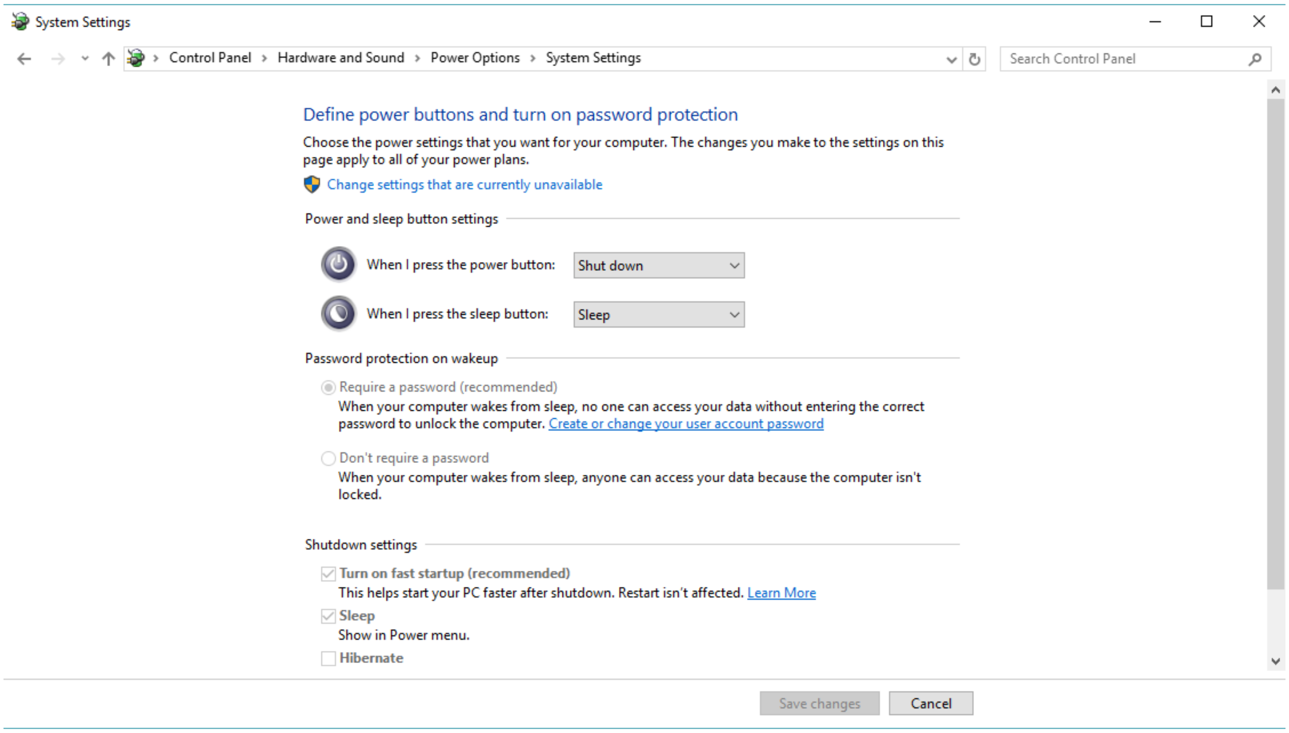
In order to share the main PC network adapter a bridge must be created with the virtual TenAsys INtime adapter. This is done by going to your Network Connections within Windows®, hold the control key and click on the Network adapter desired as well as the TenAsys Virtual Adapter, right click and click “Bridge Connections”. In a quad core system CTPLC_1 is used, CTECAT_1 in dual core.



To remove the Bridge, simple right click on the adapter and click “Remove from Bridge”. The QuickBuilder PLC Logic core will not be able to communicate via UDP or TCP without the Bridge in place. Reboot after creation.

In order to find out the current IP Address assigned to the 5300 PLC Logic environment reference the CTPLC_1 IO Console. A message will appear at startup displaying that assigned either by DHCP or static IP. CTECAT_1 IO Console will display the MAC Address of the assigned Ethernet port, typically referenced for licensing.

Once the Bridge is created we have discovered a problem within Windows where doing a shutdown on some systems causes the Bridge to not function anymore. This has something to do with a shutdown being a memory image dump on Windows 10 and not really a full shutdown, more a hibernate. The resolution to this problem is to go into ‘Power Options’ within the Control Panel, on the left side of the panel there is an option “Choose what the power buttons do”, select that and the screen below will appear. Note that the option “Turn on fast startup (recommended)” is grayed out. Select “Change settings that are unavailable” and you will be able to modify it. Deselect it and you may then press the “Save Changes” button. This will allow a full shutdown and resolve any bridge issues caused by rebooting. Note that “Restart” does not have the issue, only “Shutdown”, due to hibernation mode being entered for fast startup.



System Settings
Control Panel › Hardware and Sound › Power Options › System Settings
Search Control Panel

Define power buttons and turn on password protection

Choose the power settings that you want for your computer. The changes you make to the settings on this page apply to all of your power plans.

Change settings that are currently unavailable

Power and sleep button settings

When I press the power button: Shut down

When I press the sleep button: Sleep

Password protection on wakeup

● Require a password (recommended)
When your computer wakes from sleep, no one can access your data without entering the correct password to unlock the computer. Create or change your user account password

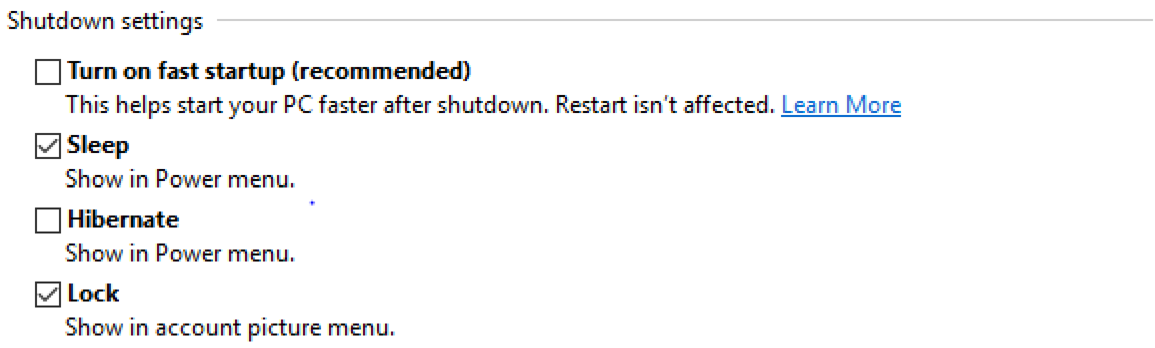
○ Don't require a password
When your computer wakes from sleep, anyone can access your data because the computer isn't locked.

Shutdown settings

☑ Turn on fast startup (recommended)
This helps start your PC faster after shutdown. Restart isn't affected. Learn More

☑ Sleep
Show in Power menu.

□ Hibernate



Shutdown settings

□ Turn on fast startup (recommended)
This helps start your PC faster after shutdown. Restart isn't affected. Learn More

☑ Sleep
Show in Power menu.

□ Hibernate
Show in Power menu.

☑ Lock
Show in account picture menu.

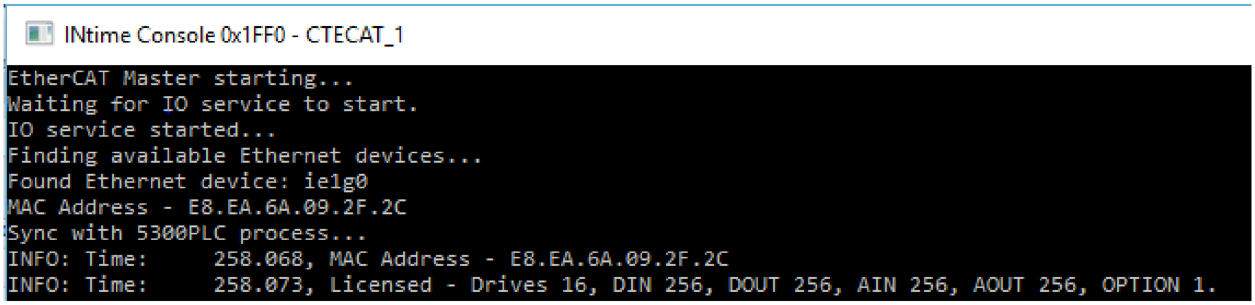
MAC Address for Large Systems

The MAC Address used by the INtime Bridged virtual adapter is dynamically created. In large systems there is the potential for duplicates to occur. If this should happen or to prevent it from happening a unique MAC Address should be used. It is recommended to use the MAC Address that appears on The CTECAT_1 screen for your EtherCAT network. This network will be private and ensure a unique MAC Address. Thus edit the file loader.cfg found in: C:\ProgramData\TenAsys\INtime\CTPLC_1\etc

At the end of the file is something like:

```
dev.ven0.macaddr=1A:EF:C8:5B:BC:00
```

Change the macaddr to that found on the EtherCAT IO Console screen:



INtime Console 0x1FF0 - CTECAT_1

```
EtherCAT Master starting...
Waiting for IO service to start.
IO service started...
Finding available Ethernet devices...
Found Ethernet device: ie1g0
MAC Address - E8.EA.6A.09.2F.2C
Sync with 5300PLC process...
INFO: Time:     258.068, MAC Address - E8.EA.6A.09.2F.2C
INFO: Time:     258.073, Licensed - Drives 16, DIN 256, DOUT 256, AIN 256, AOUT 256, OPTION 1.
```

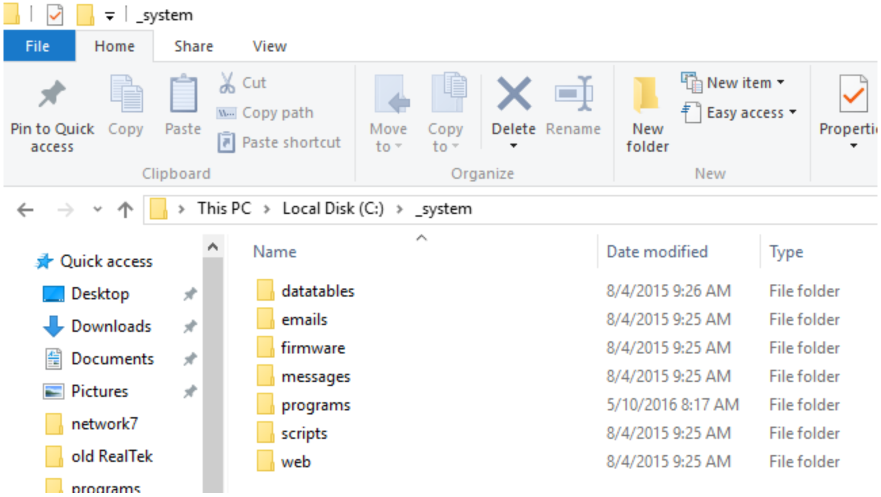
In order to use that show above the line would be changed to:

```
dev.ven0.macaddr=E8:EA:6A:09:2F:2C
```

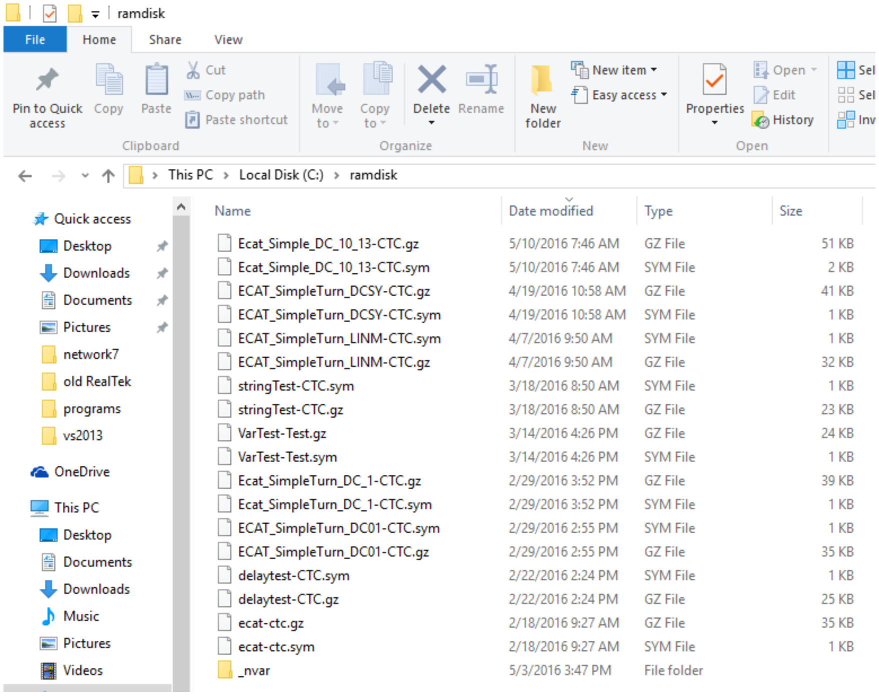
Reboot the controller for it to take effect.

File System

The PC file system mimics that of the standard embedded 5300 controller, using two subdirectories off the C: drive, '_system' and 'ramdisk'. The files within these directories are the same as described in the 5300 manual. The main difference is the local SATA solid state drive is used instead of the flash file system of the 5300 for the '_system' directory, as well as 'ramdisk'. As with the embedded 5300, the \ramdisk\nvar subdirectory contains any variant storage as well as some scratch files called 'nv#501.reg' and 'nv#32001.reg' (where # is 0 or 1). These files contain the non-volatile registers 501 to 1000 and 32001 to 36000. If any of these 5300 registers are modified, these files are updated every 5 seconds and saved during a normal shutdown for reloading at the next power-up.

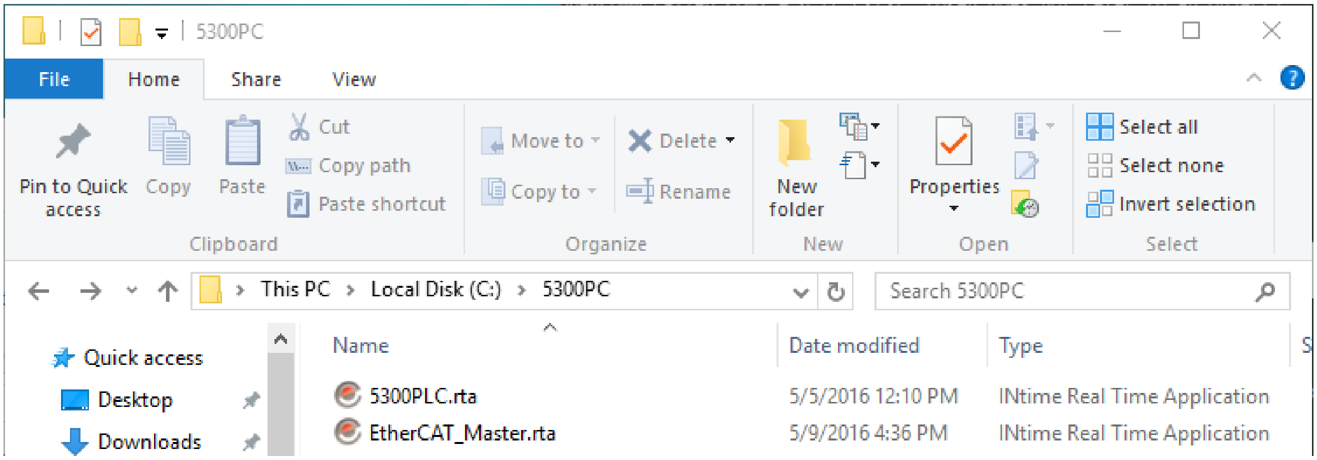


Name	Date modified	Type
datatables	8/4/2015 9:26 AM	File folder
emails	8/4/2015 9:25 AM	File folder
firmware	8/4/2015 9:25 AM	File folder
messages	8/4/2015 9:25 AM	File folder
programs	5/10/2016 8:17 AM	File folder
scripts	8/4/2015 9:25 AM	File folder
web	8/4/2015 9:25 AM	File folder



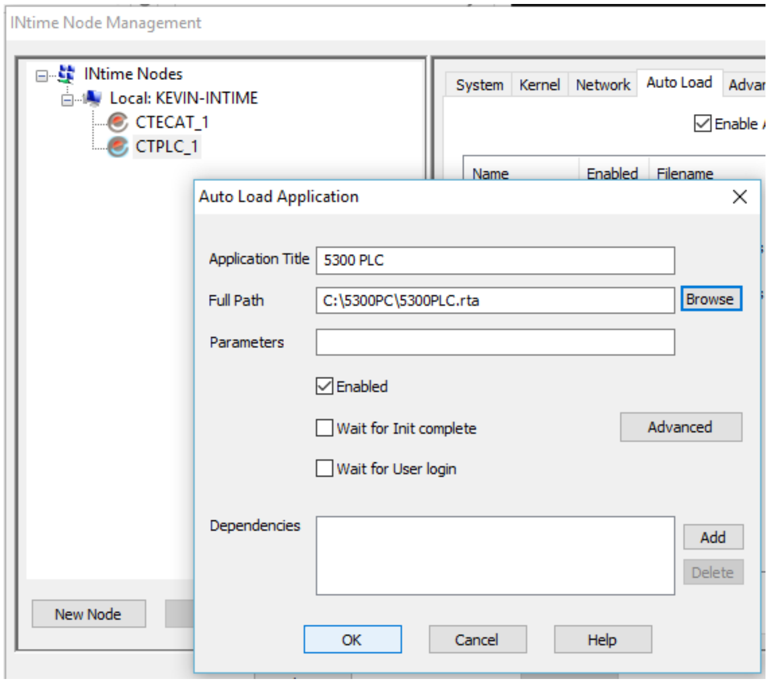
Name	Date modified	Type	Size
Ecat_Simple_DC_10_13-CTC.gz	5/10/2016 7:46 AM	GZ File	51 KB
Ecat_Simple_DC_10_13-CTC.sym	5/10/2016 7:46 AM	SYM File	2 KB
ECAT_SimpleTurn_DCSY-CTC.gz	4/19/2016 10:58 AM	GZ File	41 KB
ECAT_SimpleTurn_DCSY-CTC.sym	4/19/2016 10:58 AM	SYM File	1 KB
ECAT_SimpleTurn_LINM-CTC.sym	4/7/2016 9:50 AM	SYM File	1 KB
ECAT_SimpleTurn_LINM-CTC.gz	4/7/2016 9:50 AM	GZ File	32 KB
stringTest-CTC.sym	3/18/2016 8:50 AM	SYM File	1 KB
stringTest-CTC.gz	3/18/2016 8:50 AM	GZ File	23 KB
VarTest-Test.gz	3/14/2016 4:26 PM	GZ File	24 KB
VarTest-Test.sym	3/14/2016 4:26 PM	SYM File	1 KB
Ecat_SimpleTurn_DC_1-CTC.gz	2/29/2016 3:52 PM	GZ File	39 KB
Ecat_SimpleTurn_DC_1-CTC.sym	2/29/2016 3:52 PM	SYM File	1 KB
ECAT_SimpleTurn_DC01-CTC.sym	2/29/2016 2:55 PM	SYM File	1 KB
ECAT_SimpleTurn_DC01-CTC.gz	2/29/2016 2:55 PM	GZ File	35 KB
delaytest-CTC.sym	2/22/2016 2:24 PM	SYM File	1 KB
delaytest-CTC.gz	2/22/2016 2:24 PM	GZ File	25 KB
ecat-ctc.gz	2/18/2016 9:27 AM	GZ File	35 KB
ecat-ctc.sym	2/18/2016 9:27 AM	SYM File	1 KB
_nvar	5/3/2016 3:47 PM	File folder	

The real-time applications which run on each CPU core are maintained in the \5300PC directory, 5300PLC.rta and EtherCAT_Master.rta and are referenced from the INtime Node Management Auto Load tab.



Name	Date modified	Type
5300PLC.rta	5/5/2016 12:10 PM	INtime Real Time Application
EtherCAT_Master.rta	5/9/2016 4:36 PM	INtime Real Time Application

Auto Load configuration, starts EtherCAT from tray icon, the default for quad core:



Auto Load Application

Application Title: 5300 PLC

Full Path: C:\5300PC\5300PLC.rta

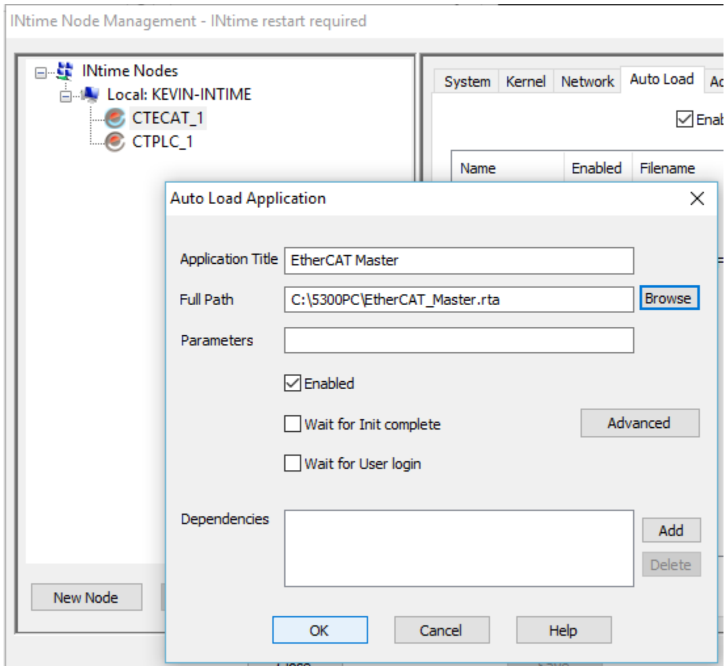
Parameters:

☑ Enabled

☐ Wait for Init complete

☐ Wait for User login

Dependencies:



Auto Load Application

Application Title: EtherCAT Master

Full Path: C:\5300PC\EtherCAT_Master.rta

Parameters:

☑ Enabled

☐ Wait for Init complete

☐ Wait for User login

Dependencies:

In a dual core system (single core used by Incentive) both EtherCAT_Master.rta and 5300PLC.rta would be listed under CTECAT_1, CTPLC_1 would not exist (freeing up a core for Windows®). This is also true for standalone mode, when driven solely by the Windows API.

The _system\Programs directory contains various configuration files. These files are set by using the EtherCAT Explorer ‘User Options’, ‘Save Config, and ‘License’ forms. Since more than one EtherCAT network can run at a time there are some files that contain the MAC Address of the EtherCAT Ethernet adapter as part of its file name to make the name unique. Files in this directory are:

Used by the 5300PLC.rta process –

_lcddisplay.txt : Some embedded PC devices have an LCD display for status messages to be displayed. If present this file contains the IP address in text format, for example: 172.16.2.190
_runprogram.txt : Contains the path\filename of any QuickBuilder project to automatically startup at boot. It is set by QuickBuilder during a download of a project file if the option is enabled in the project properties. For example: \RAMDISK\ECAT_SimpleTurn_DCSY-CTC.gz
_Init.bin: Saves configuration information like custom serial port setting, baud rates, etc. It is automatically created with the system defaults if it does not exist.

Used by the EtherCAT_Master.rta process –

options[MAC ADDRESS].txt : This file contains the settings from the 'User Options' EtherCAT Explorer configuration form. If no file exists a file will be created with default settings of 1 mS cycle time, 300uS pdo timeout, 5 initialization retries, and no virtual axis. Example filename: _options_E8EA6A092F2C.txt where E8EA6A092F2C is the 6 byte mac address of the EtherCAT Ethernet network adapter being used.

ioOptions[MAC ADDRESS].txt : This file contains the license settings from the EtherCAT Explorer license configuration form. It is an encrypted binary file. When the software is first installed a file called _license_[MAC ADDRESS].txt will be emailed to you. That file should be placed in the same directory as this file. When the _ioOptions file is not present the system will look for a valid _license file. If found that file will be imported and the _ioOptions file generated. The _ioOptions file can also be generated from the QuickBuilder EtherCAT Explorer configuration form, as it is with the 5300 M3-41A. In a PC system it is typically easier to have the license file emailed to you and then placed in the _system\Programs directory for automatic import.

license[MAC ADDRESS].txt : This file is generated by CTC Technical Support to enable the number of drives and IO your EtherCAT runtime is licensed for. If this file and the _ioOptions file are both missing the system will run in demo mode. In this mode EtherCAT will control up to 16 drives and 256 digital and analog IO for a period of about 3 hours. After that the network will reset and power will have to be cycled to restart another 3 hour period. The generation of this file requires the MAC Address of the EtherCAT network adapter. You can get the MAC Address by starting the EtherCAT_Master.rta process and it will appear on the console screen. Either email Technical Support or have it available when calling so that a proper license file can be generated. Systems pre-configured by CTC will already have the license installed. Example filename: _license_E8EA6A092F2C.txt.

log[MAC ADDRESS].txt : This file is written to with the same information as the IO diagnostic console and provides a hard copy of each session. The file is automatically appended to. Example filename: _log_E8EA6A092F2C.txt where E8EA6A092F2C is the 6 byte mac address of the EtherCAT Ethernet network adapter being used.

It is recommended that an inexpensive UPS be connected to a serial port for automatic shutdown should extended or abrupt power failures occur. The will ensure file integrity as well as preservation of non-volatile registers.

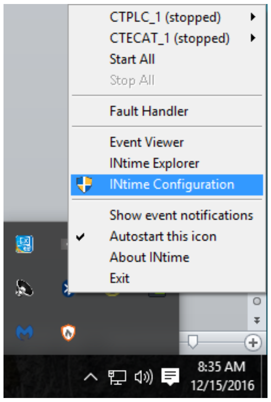
Demo Mode and Licensing

Both the TenAsys INtime runtime, Incentive API and EtherCAT Master can run in demo mode. Demo mode for INtime is valid for 60 days after which the QuickBuilder EtherCAT environment must be purchased. It cannot be extended due to limitations imposed by TenAsys. The QuickBuilder EtherCAT network will run for up to about 3 hours continuously, before resetting. In this mode EtherCAT will control up to 16 drives and 256 digital and analog IO. In order to purchase a full EtherCAT license please contact Control Technology with your motion control and IO needs as well as the MAC Address of the Ethernet adapter that will be used. From Windows the MAC Address can be found by opening a 'cmd' window and typing 'ipconfig /all'. The Physical Address of your adapter is the MAC Address.

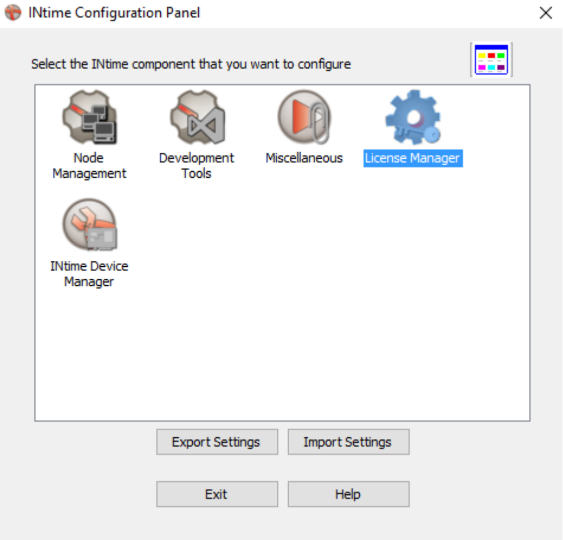
TenAsys INtime Licensing

If your INtime license has not already been installed you must send a system fingerprint to CTC (along with purchase order) so that a license string can be generated for your system. This license is keyed to your hard drive so should that be replaced, you will need a new license.

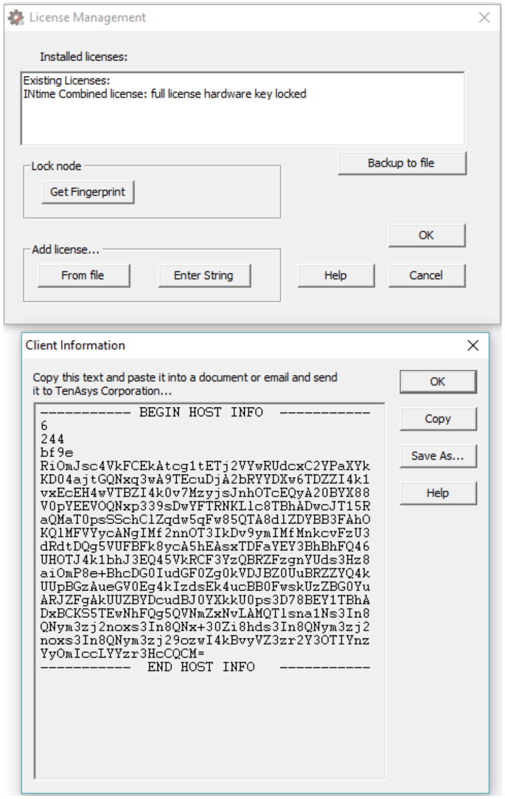
In order to retrieve the fingerprint go to the system tray and select 'INtime Conguration', right clicking on the TenAsys INtime icon:



The INtime Configuration Panel will open. Select 'License Manager':



Select ‘Get Fingerprint’ followed by ‘Copy’ or ‘Save As…’ on the Client Information form that opens:



Existing Licenses:
INtime Combined license: full license hardware key locked

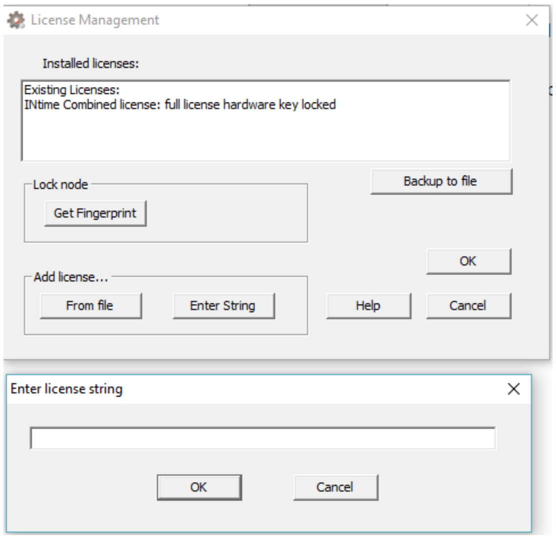
Copy this text and paste it into a document or email and send it to TenAsys Corporation…

```
----------- BEGIN HOST INFO  -----------
6
244
bf9e
RiOmJsc4VkFCEkAtcg1tETj2VYwRUdcxC2YPaXYk
KD04ajtGQNxq3wA9TEcuDjA2bRYYDXw6TDZZI4k1
vxEcEH4wVTBZI4k0v7MzyjsJnhOTcEQyA20BYX88
V0pYEEVOQNxp339sDwYFTRNKLlc8TBhADwcJT15R
aQMaT0psSSchClZqdw5qFw85QTA8dlZDYBB3FAhO
KQlMFVYycANgIMf2nnOT3IkDv9ymIMfMnkcvFzU3
dRdtDQg5VUFBFk8ycA5hEAsxTDFaYEY3BhBhFQ46
UHOTJ4k1bhJ3EQ45VkRCF3YzQBRZFzgnYUds3Hz8
aiOmP8e+BhcDG0IudGF0Zg0kVDJBZ0UuBRZZYQ4k
UUpBGzAueGV0Eg4kIzdsEk4ucBB0FwskUzZBG0Yu
ARJZFgAkUUZBYDcudBJ0YXkkU0ps3D78BEY1TBhA
DxBCKS5TEwNhFQg5QVNmZxNvLAMQTlsna1Ns3In8
QNym3zj2noxs3In8QNx+30Zi8hds3In8QNym3zj2
noxs3In8QNym3zj29ozwI4kBvyVZ3zr2Y3OTIYnz
YyOmIccLYYzr3HcCQCM=
-----------   END HOST INFO   -----------
```

Copy (or attach) the information into an email and send it to CTC for validation. Once validate a license string will be generated and returned. An example of this would look something like:

*AA2KZ4OIAJ75NDSYKM6GN3ETSEVKHRA44DRS3KBJZIB8IEEWSVYY2# "16" version "", no expiration date, exclusive

Select ‘Enter String’ from the License Management screen, copy and paste the license you sent them followed by clicking ‘OK’:



Existing Licenses:
INtime Combined license: full license hardware key locked

Incentive Licensing

Your Incentive license is keyed to the MAC Address of the Ethernet port used for EtherCAT. While in demo mode it will run for 3 hours and then need restarting. To fully unlock a license file (_license_[MAC ID].txt must be requested from CTC which defines the number and type of IO licensed as well as to whether the API is fully enabled. This file would be placed in the C:_system\programs directory. All _ioOptions* files must be removed if only a EtherCAT Master. If multiple then only remove the existing _ioOptions* file that matches the MAC Address of the network adapter used.

> ***_license_[MAC ADDRESS].txt*** : This file is generated by CTC Technical Support to enable the number of drives and IO your EtherCAT runtime is licensed for. If this file and the _ioOptions file are both missing the system will run in demo mode. In this mode EtherCAT will control up to 16 drives and 256 digital and analog IO for a period of about 3 hours. After that the network will reset and power will have to be cycled to restart another 3 hour period. The generation of this file requires the MAC Address of the EtherCAT network adapter. You can get the MAC Address by starting the EtherCAT_Master.rta process and it will appear on the console screen. Either email Technical Support or have it available when calling so that a proper license file can be generated. Systems pre-configured by CTC will already have the license installed. Example filename: _license_E8EA6A092F2C.txt.

Windows® Updates

Windows® updates are pushed by Microsoft periodically and are required for proper operation as well as security. With the release of Windows® 10 Micosoft forces updates rather than allowing you to be prompted. Doing an update while you are controlling an automation environment is generally not a good thing since in many cases Microsoft reboots the PC. Thus it is strongly suggested that updates be turned off or at least request a prompt. On Windows® 10 you can set a connection, network or WiFi, as metered to prevent updates. Wifi can be set within the Control Panel whereas an Ethernet network must be set via a registry entry. There are numerous articles available about how to do this with that below being a particularly good reference:

Windows 10 Pro (schedule updates):

http://www.windowscentral.com/how-schedule-windows-updates-windows-10
You can also go to "Settings->Advanced Options->Defer feature updates" should be selected.

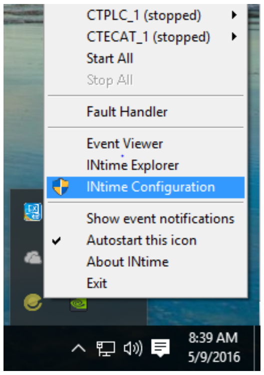
Windows 10 Home & Pro (metered connection):

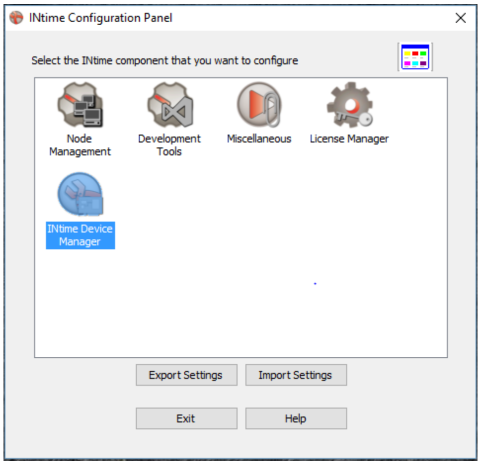
http://www.windowscentral.com/how-set-ethernet-connection-metered-windows-10

Failure to defer your updates until you are ready for them will cause your PC to reboot automatically at unpredictable times due to Microsoft pushing out Windows Updates. This will cause your real time control to stop operation as well.

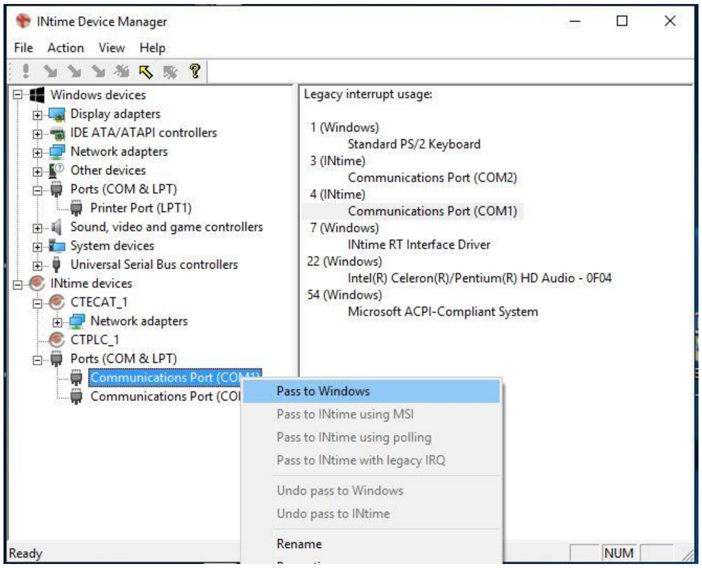
Serial Ports

At present time the QuickBuilder PLC Logic core will attempt to open COM1 through COM4 for its own use. Whichever ports are found will be used by the standard CTC Binary Protocol and available for QuickBuilder programming, those not found will cause a diagnostic warning on the Incentive PLC console screen. Serial ports may be mapped back and forth between the INtime real-time environment or for Windows® programming use. Mapping is required in order for ports to be disabled from Windows® and enabled for INtime. To map serial ports between the INtime real-time environment and Windows® the Intime Device Manager must be used:



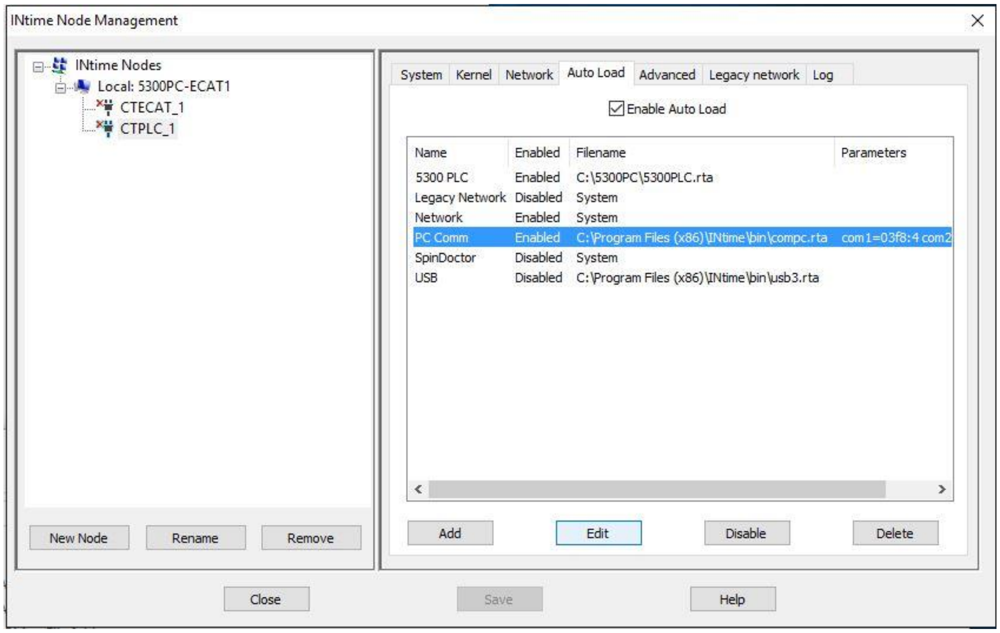


Note to return a serial port to Windows control, right click on the COM port under INtime devices and select 'Pass to Windows'. To map it back select the COM port under Windows devices and select 'Pass to INtime with legacy IRQ'. Rebooting is required for it to map properly.



```
Legacy interrupt usage:

1 (Windows)
      Standard PS/2 Keyboard
3 (INtime)
      Communications Port (COM2)
4 (INtime)
      Communications Port (COM1)
7 (Windows)
      INtime RT Interface Driver
22 (Windows)
      Intel(R) Celeron(R)/Pentium(R) HD Audio - 0F04
54 (Windows)
      Microsoft ACPI-Compliant System
```

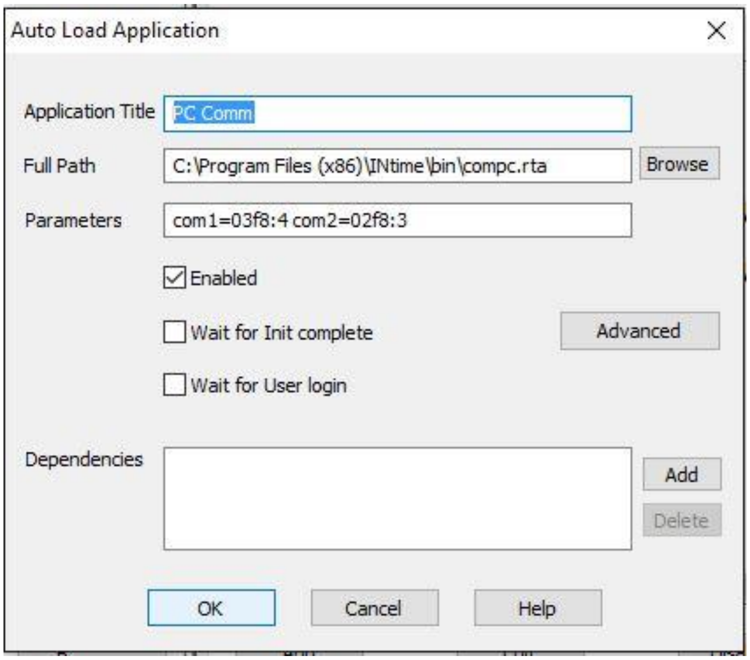
In addition, when mapping serial ports a special driver (compc.rta) must be passed the IO address of the port and interrupt used. This is configured within INtime Node Management, CTPLC_1 for quad core, CTECAT_1 if dual:



☑ Enable Auto Load

Name	Enabled	Filename	Parameters
5300 PLC	Enabled	C:\5300PC\5300PLC.rta	
Legacy Network	Disabled	System	
Network	Enabled	System	
PC Comm	Enabled	C:\Program Files (x86)\INtime\bin\compc.rta	com1=03f8:4 com2
SpinDoctor	Disabled	System	
USB	Disabled	C:\Program Files (x86)\INtime\bin\usb3.rta	

Modify the 'Parameters' section as required. You may refer to the Windows® Device Manager->Ports (COM & LPT) section properties to determine what is currently available and IO address:interrupts used.



Application Title: PC Comm

Full Path: C:\Program Files (x86)\INtime\bin\compc.rta

Parameters: com1=03f8:4 com2=02f8:3

☑ Enabled

□ Wait for Init complete

□ Wait for User login

Dependencies:

For example the parameters for COM1 with a hex address of 0x03f8, using IRQ 4, and COM2 with a hex address of 0x02f8, IRQ 3 are shown above.

Serial ports are referenced by Incentive starting with COM1 and added sequentially. This means if COM2 is mapped by INtime but not COM1 then COM2 will become COM1 within the Incentive environment. If COM2 and COM3 were mapped then Incentive COM1 is Windows COM2 and Incentive COM2 is Windows COM3. Incentive has no idea of how Windows referenced the COM ports, just each assigned will become sequential, from 1 to 4.

Another thing to keep in mind is that you cannot change the serial port parameters while the port is open (Windows must have the port closed), thus the registers for changing baud rate, parity, data and stop bits, will have no effect. The proper way to set serial port parameters is via the _Init.bin file, where settings are stored and read during initialization. The 'set COMM#' command, available via telnet, can be used to retrieve and save serial port setups.

Set COMM[Port] [Baud Rate], [Data Bits], [Parity], [Stop Bits], [Protocol], [Flow Control], [Address]

Port

1, 2, 3, or 4 are valid entries as long as the port is available. Example: 'set COMM1', 'set COMM2', …

Baud Rate

Baud Rate may be one of the following (19200 is the default):

- 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200

Data Bits

"7" or "8" data bits with 8 being the default.

Parity

"None", "Odd", or "Even" parity, with "None" being the default.

Stop Bits

"1" or "2" stop bits with "1" being the default.

Protocol

Protocol may be one of below, case sensitive:

- CTC Binary (Default, compatible with CTCMON and ctccom32.dll)
- Modbus Master/RTU – controller polls the device, binary mode.
- Modbus Master/ASCII – controller polls the device, ASCII mode.
- Modbus Slave/RTU – controller polled by external device, binary mode
- Modbus Slave/ASCII – controller polled by external device, ASCII mode
- Diag. Terminal – Diagnostic Terminal
- Philips ISP – Programmable chip mode
- Scale – Custom scale protocol

Flow Control (not supported on 5300, only Incentive PC environment)

"None", "Xon", or "Hardware" are available options.

None – Typically 2 wire, TX/RX half duplex protocol like CTC Binary.

Xon – Xoff (0x13) is sent when receive buffer is full, Xon (0x11) when OK to send again.

Hardware – DTR is enabled when the port is open. RTS is active when OK to receive characters, CTS controls transmitter, when not active transmission will not occur.

Address

This is the address to be used when Modbus protocols are selected. When in Master mode only a single device may be polled. To poll multiple devices the Address register must be changed by the Quickstep program, dynamically. An address from 1 to 255 is valid with 1 as the default.

Example:

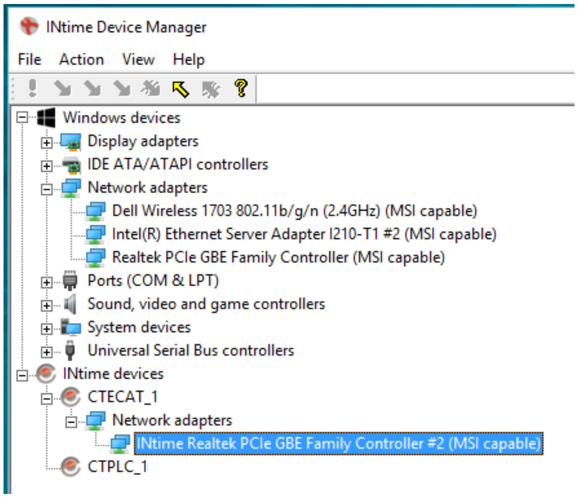
Set COMM1 19200, 8, None, 1, CTC Binary, None, 1

You can also use 'get COMM[#]' to retrieve the current settings.

EtherCAT Ethernet Adapter

Within the INtime configuration screens you will notice CTECAT_1 for a single network configuration, CTECAT_2 would be added for dual network. These are the real-time processor cores responsible for the

EtherCAT Master network. Reference the 'INtime Device Manager' for those available and/or presently being used.



- Windows devices
  - Display adapters
  - IDE ATA/ATAPI controllers
  - Network adapters
    - Dell Wireless 1703 802.11b/g/n (2.4GHz) (MSI capable)
    - Intel(R) Ethernet Server Adapter I210-T1 #2 (MSI capable)
    - Realtek PCIe GBE Family Controller (MSI capable)
  - Ports (COM & LPT)
  - Sound, video and game controllers
  - System devices
  - Universal Serial Bus controllers
- INtime devices
  - CTECAT_1
    - Network adapters
      - INtime Realtek PCIe GBE Family Controller #2 (MSI capable)
  - CTPLC_1

The above configuration shows a Realtek PCIe GBE Controller being used under 'INtime devices.' Referencing 'Windows devices', there are three additional adapters currently used by Windows® which are available. The ideal adapter to use would be the I210-T1 adapter as that offers special high speed queuing and an internal precision timer that is optimum for EtherCAT packet timing. For example the Realtek adapter will typically be about 30 uS average jitter with a max of 200 uS (400 uS dual core) whereas the I210 is typically has sub-microsecond average jitter and significantly offloads the processor allowing for greater number of EtherCAT devices to be supported on a single network.

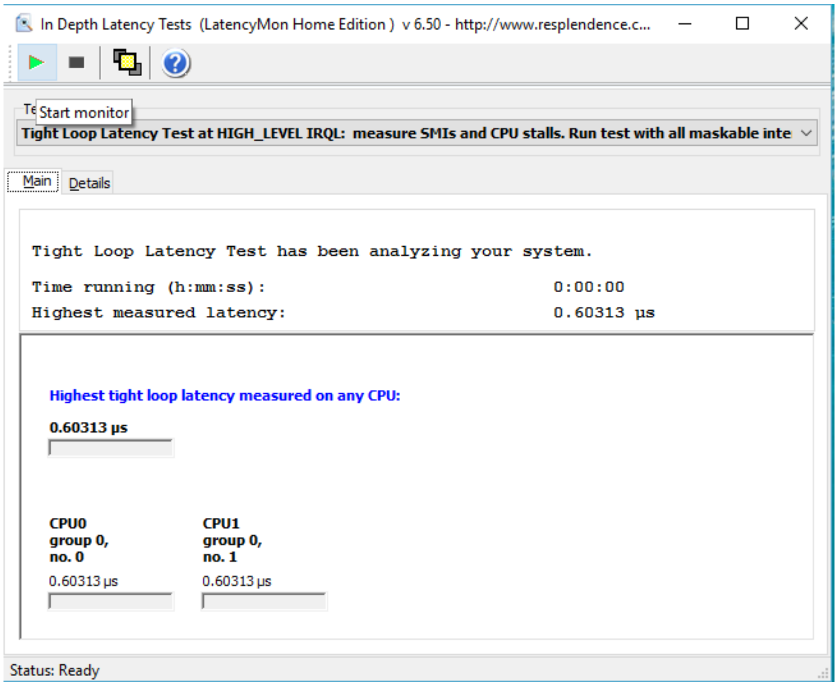
The CTECAT core will automatically detect which adapter is selected and attempt to use it. When mapping an adapter to the INtime environment be sure to select the 'Pass to INtime using MSI' property. Only a single Ethernet adapter is supported by each CTECAT core, where _1 is incremented to __2 when additional networks are used for EtherCAT. As with serial ports, reboot after any assignment changes.

System Management Interrupt Detection

System Management Interrupts are calls made to the BIOS for special functions. This can be something as simple as USB support for a mouse and/or keyboard versus using a PS/2 version or thermal control of the CPU. Some SMI's are necessary but with some BIOS's bad practices are encounters where several hundred microseconds of disabling control from higher levels can occur, thus ruining any real-time performance and causing jitter.

Luckily there is software available to test a system. One free version in particular is the home version of LatencyMon (http://www.resplendence.com/latencymon). When you download and install this two programs are available, 'LatencyMon' and 'In Depth Latency Tests.' The 'In Depth Latency Tests' program can be used to detect the effect of SMI's with you BIOS. Something around 50 uS or less is desirable, more can be tolerated with the I210 network adapter.

Once the program is started you may select the "Start Monitor" green icon button and let it run for about 30 seconds as you do your normal operations on the computer, access a file, web browsing, etc. Then click the red "Stop Monitor" button:


In Depth Latency Tests (LatencyMon Home Edition) v 6.50 - http://www.resplendence.c...
Te Start monitor
Main
Details

Tight Loop Latency Test at HIGH_LEVEL IRQL: measure SMIs and CPU stalls. Run test with all maskable inte

```
Tight Loop Latency Test has been analyzing your system.

Time running (h:mm:ss):                              0:00:00
Highest measured latency:                            0.60313 µs
```

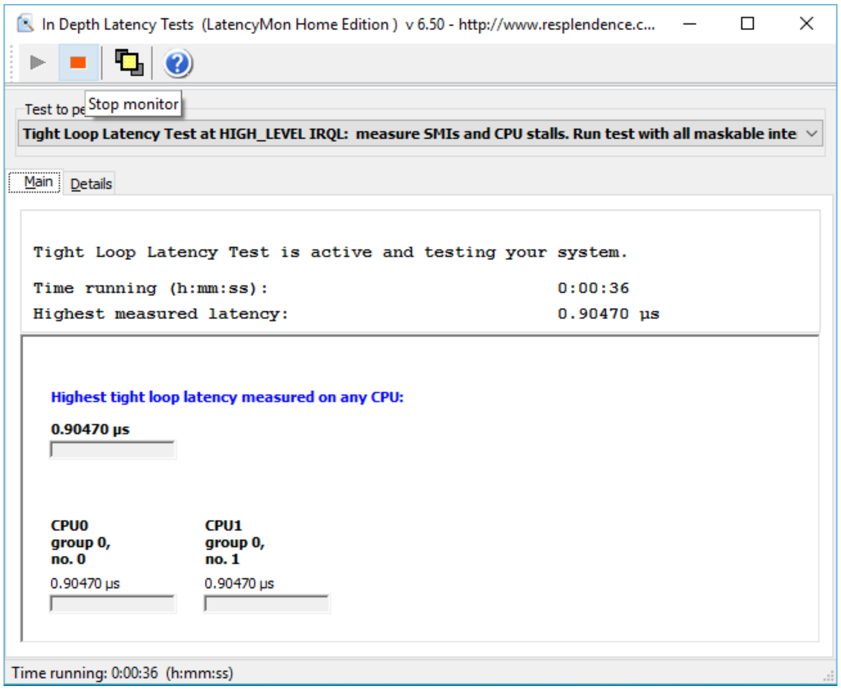
Highest tight loop latency measured on any CPU:

0.60313 µs

CPU0 group 0, no. 0: 0.60313 µs

CPU1 group 0, no. 1: 0.60313 µs

Status: Ready


In Depth Latency Tests (LatencyMon Home Edition) v 6.50 - http://www.resplendence.c...
Test to pe Stop monitor
Main
Details

Tight Loop Latency Test at HIGH_LEVEL IRQL: measure SMIs and CPU stalls. Run test with all maskable inte

```
Tight Loop Latency Test is active and testing your system.

Time running (h:mm:ss):                              0:00:36
Highest measured latency:                            0.90470 µs
```

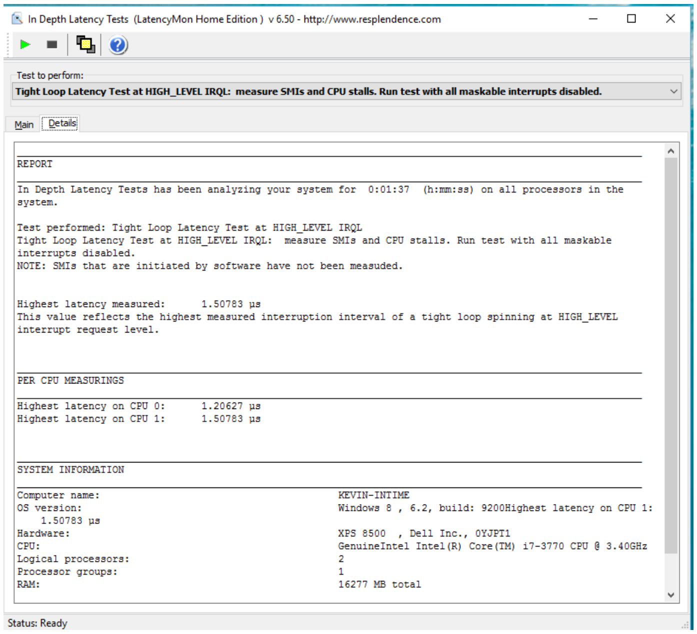
Highest tight loop latency measured on any CPU:

0.90470 µs

CPU0 group 0, no. 0: 0.90470 µs

CPU1 group 0, no. 1: 0.90470 µs

Time running: 0:00:36 (h:mm:ss)


In Depth Latency Tests (LatencyMon Home Edition) v 6.50 - http://www.resplendence.com
Main
Details
Status: Ready

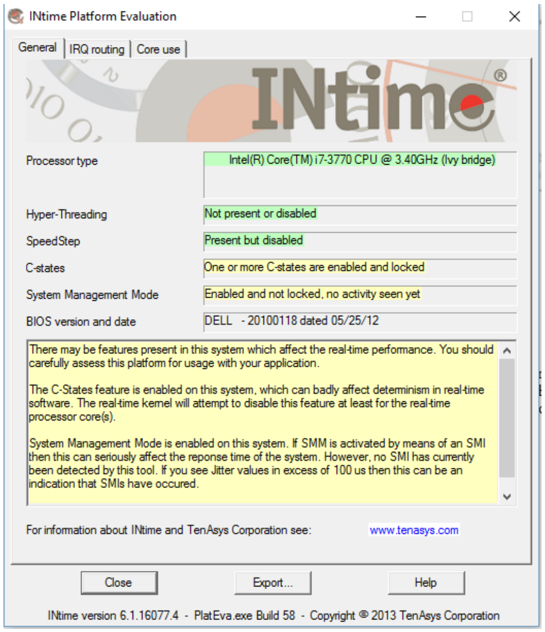
Test to perform:
Tight Loop Latency Test at HIGH_LEVEL IRQL: measure SMIs and CPU stalls. Run test with all maskable interrupts disabled.

```
REPORT

In Depth Latency Tests has been analyzing your system for  0:01:37  (h:mm:ss) on all processors in the
system.

Test performed: Tight Loop Latency Test at HIGH_LEVEL IRQL
Tight Loop Latency Test at HIGH_LEVEL IRQL:  measure SMIs and CPU stalls. Run test with all maskable
interrupts disabled.
NOTE: SMIs that are initiated by software have not been measuded.


Highest latency measured:      1.50783 µs
This value reflects the highest measured interruption interval of a tight loop spinning at HIGH_LEVEL
interrupt request level.



PER CPU MEASURINGS

Highest latency on CPU 0:      1.20627 µs
Highest latency on CPU 1:      1.50783 µs



SYSTEM INFORMATION

Computer name:                                        KEVIN-INTIME
OS version:                                           Windows 8 , 6.2, build: 9200Highest latency on CPU 1:
    1.50783 µs
Hardware:                                             XPS 8500  , Dell Inc., 0YJPT1
CPU:                                                  GenuineIntel Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
Logical processors:                                   2
Processor groups:                                     1
RAM:                                                  16277 MB total
```

The Details tab will yield a report. In this test there were 2 CPU cores found (INtime has control of the other 2 cores during this test but typically 4 cores would be seen) with a maximum latency of 1.2 uS on one core and 1.5 on the second core. This is excellent real-time performance. A J1900 Celeron would probably be around 50 uS. Some BIOS vendors are as high as 200 uS which is unacceptable.

Platform Evaluation

Once the real-time environment is installed, even in demo mode, an actual evaluation of the suitability of that PC can be made by running two programs. The first is the INtime Platform Evaluation utility. This utility will check various BIOS settings and give warnings of things that may need changing. Additionally IRQ routine and how the CPU cores are presently assigned are available via that tabs:



Processor type	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz (Ivy bridge)
Hyper-Threading	Not present or disabled
SpeedStep	Present but disabled
C-states	One or more C-states are enabled and locked
System Management Mode	Enabled and not locked, no activity seen yet
BIOS version and date	DELL - 20100118 dated 05/25/12

There may be features present in this system which affect the real-time performance. You should carefully assess this platform for usage with your application.

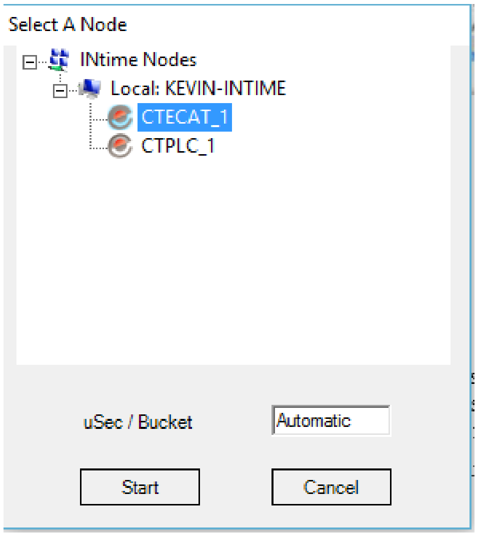
The C-States feature is enabled on this system, which can badly affect determinism in real-time software. The real-time kernel will attempt to disable this feature at least for the real-time processor core(s).

System Management Mode is enabled on this system. If SMM is activated by means of an SMI then this can seriously affect the reponse time of the system. However, no SMI has currently been detected by this tool. If you see Jitter values in excess of 100 us then this can be an indication that SMIs have occured.

For information about INtime and TenAsys Corporation see: www.tenasys.com

INtime version 6.1.16077.4 - PlatEva.exe Build 58 - Copyright © 2013 TenAsys Corporation

The actual final test is the “Graphical Jitter Display”, this test actually checks the ability of the real-time environment to run in real-time, ideally with no jitter caused by SMI’s. Upon startup you are presented with the available cores to test (assuming they have been started). Select and test each independently:

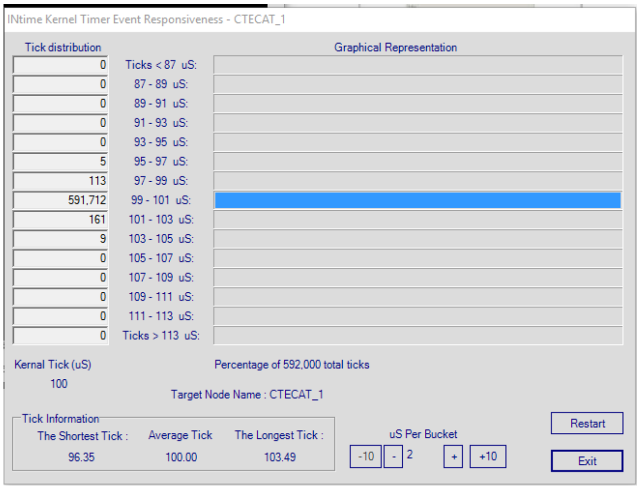


Select A Node

- INtime Nodes
  - Local: KEVIN-INTIME
    - CTECAT_1
    - CTPLC_1

uSec / Bucket: Automatic

The screen should look something like this, with no red indicators showing excessive jitter after you perform the necessary operations on your PC:



INtime Kernel Timer Event Responsiveness - CTECAT_1

Tick distribution		Graphical Representation
0	Ticks < 87 uS:	
0	87 - 89 uS:	
0	89 - 91 uS:	
0	91 - 93 uS:	
0	93 - 95 uS:	
5	95 - 97 uS:	
113	97 - 99 uS:	
591,712	99 - 101 uS:	█
161	101 - 103 uS:	
9	103 - 105 uS:	
0	105 - 107 uS:	
0	107 - 109 uS:	
0	109 - 111 uS:	
0	111 - 113 uS:	
0	Ticks > 113 uS:	

Kernal Tick (uS): 100

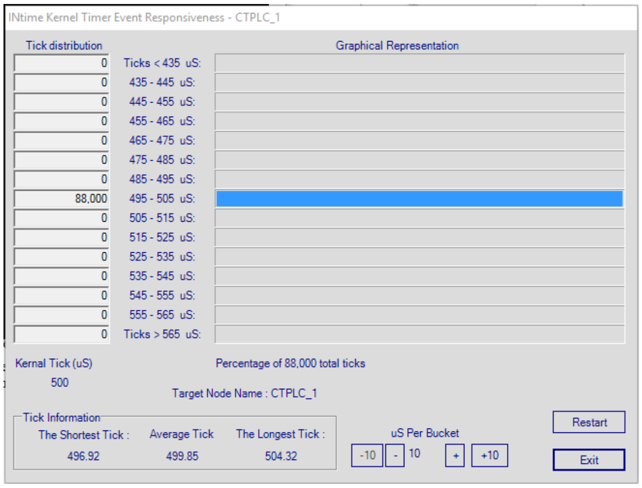
Percentage of 592,000 total ticks

Target Node Name : CTECAT_1

Tick Information

The Shortest Tick :	Average Tick	The Longest Tick :
96.35	100.00	103.49

uS Per Bucket: 2



INtime Kernel Timer Event Responsiveness - CTPLC_1

Tick distribution		Graphical Representation
0	Ticks < 435 uS:	
0	435 - 445 uS:	
0	445 - 455 uS:	
0	455 - 465 uS:	
0	465 - 475 uS:	
0	475 - 485 uS:	
0	485 - 495 uS:	
88,000	495 - 505 uS:	
0	505 - 515 uS:	
0	515 - 525 uS:	
0	525 - 535 uS:	
0	535 - 545 uS:	
0	545 - 555 uS:	
0	555 - 565 uS:	
0	Ticks > 565 uS:	

Kernal Tick (uS): 500

Percentage of 88,000 total ticks

Target Node Name : CTPLC_1

Tick Information

The Shortest Tick :	Average Tick	The Longest Tick :
496.92	499.85	504.32

uS Per Bucket: 10

CTC Incentive® .Net API

CTC has created a very powerful tool for .Net programmers, a tightly integrated dynamic link library (DLL) interface which allows users to program the PC based controller using C#, managed C++, and VB.Net. This DLL includes the complete MSB motion control language, axis properties, registers, variant storage, as well as the ability to access variable registers and symbolic names, both locally and over a network. It runs standalone or in conjunction with the QuickBuilder application programs where programs can execute in mixed mode with .Net programs controlling one aspect of the automation and QuickBuilder, another.

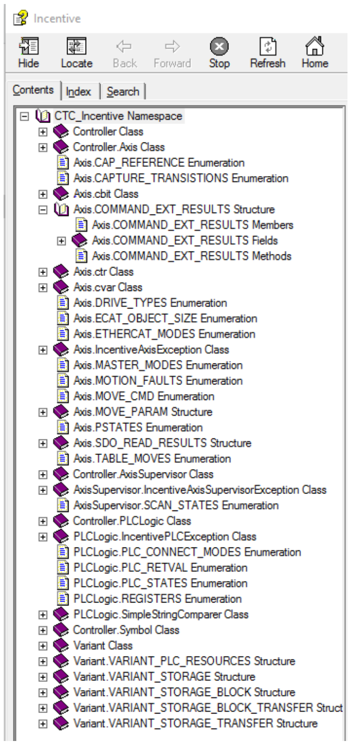
Typically QuickBuilder would be used where absolute real-time is required whereas .Net can be leveraged to coordinate larger systems, perform database queries, provide vision control interfaces, complex calculations, HMI updates, and/or real-time motion control of some of the axis, limited only by the programmer’s imagination. The only difference between QuickBuilder and the .Net interface is that the .Net program will execute under the Windows operating system (non-realtime) whereas QuickBuilder runs in real-time on a dedicated processor core.

Some of the features of the API are as follows:

- Portable .Net DLL, written in C#, abstracting the user from unmanaged code.
- Sample Applications included for C#, VB.Net, and managed C++.
- Connection based high speed parallel threaded interface to both the PLC logic and EtherCAT Master processes.
- Same exact interface whether communicating locally or over a network, simply include the defined remote hostname for CTPLC_1 process when using the API ‘openConnection’ function in a networked environment. Multiple connections supported, both local and networked.
- Full support and mapping of the QuickBuilder Motion language to .Net. .Net Methods mimic those of the QuickBuilder MSB language helping to shorten learning curves when moving between programming environments.
- Full register interface, including Variant support. Exception based error processing.

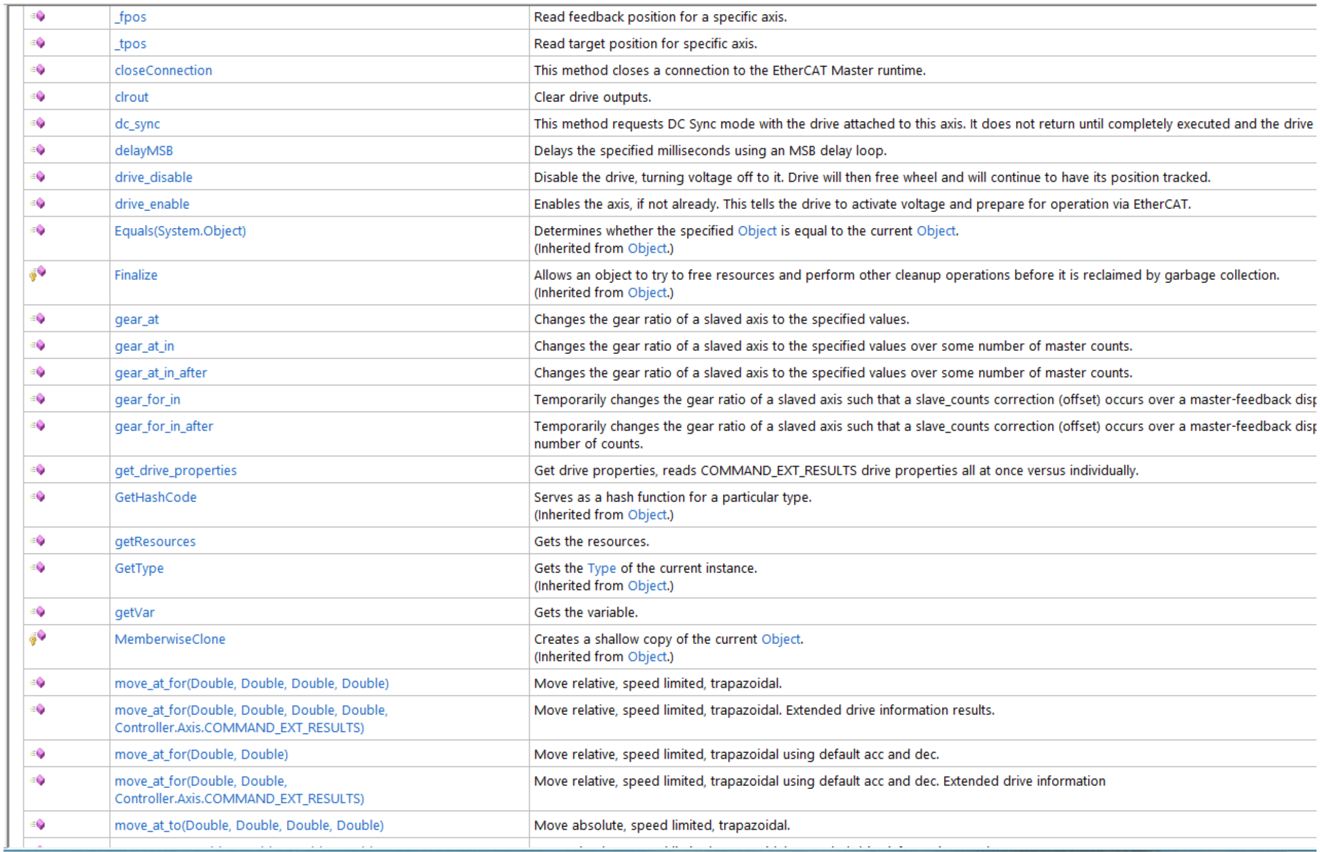
- Requests are passed to the real-time processes where they execute using existing MSB instruction objects, thus executes exactly as an MSB would.
- Full access to a QuickBuilder programs' symbol tables for variable reference, both at the QuickBuilder and MSB level. All axis properties are directly supported as well as access to user variables. This allows QuickBuilder programs to be re-compiled, variable registers re-assigned, and still work with .Net since the symbol name is the reference, not the low level register number.
- System state control to restart programs, restart EtherCAT, monitor current execution state, etc. Including managing the starting and stopping of the real time operating system INtime, as well as monitor the status of programs running on its dedicated processor cores.
- Optional standalone EtherCAT Master operation for Windows only programs. Requires only a single core for real-time versus the preferred two when using QuickBuilder applications.

Some of the class features of the CTC_Incentive Namespace (from CTC_Incentive.chm):

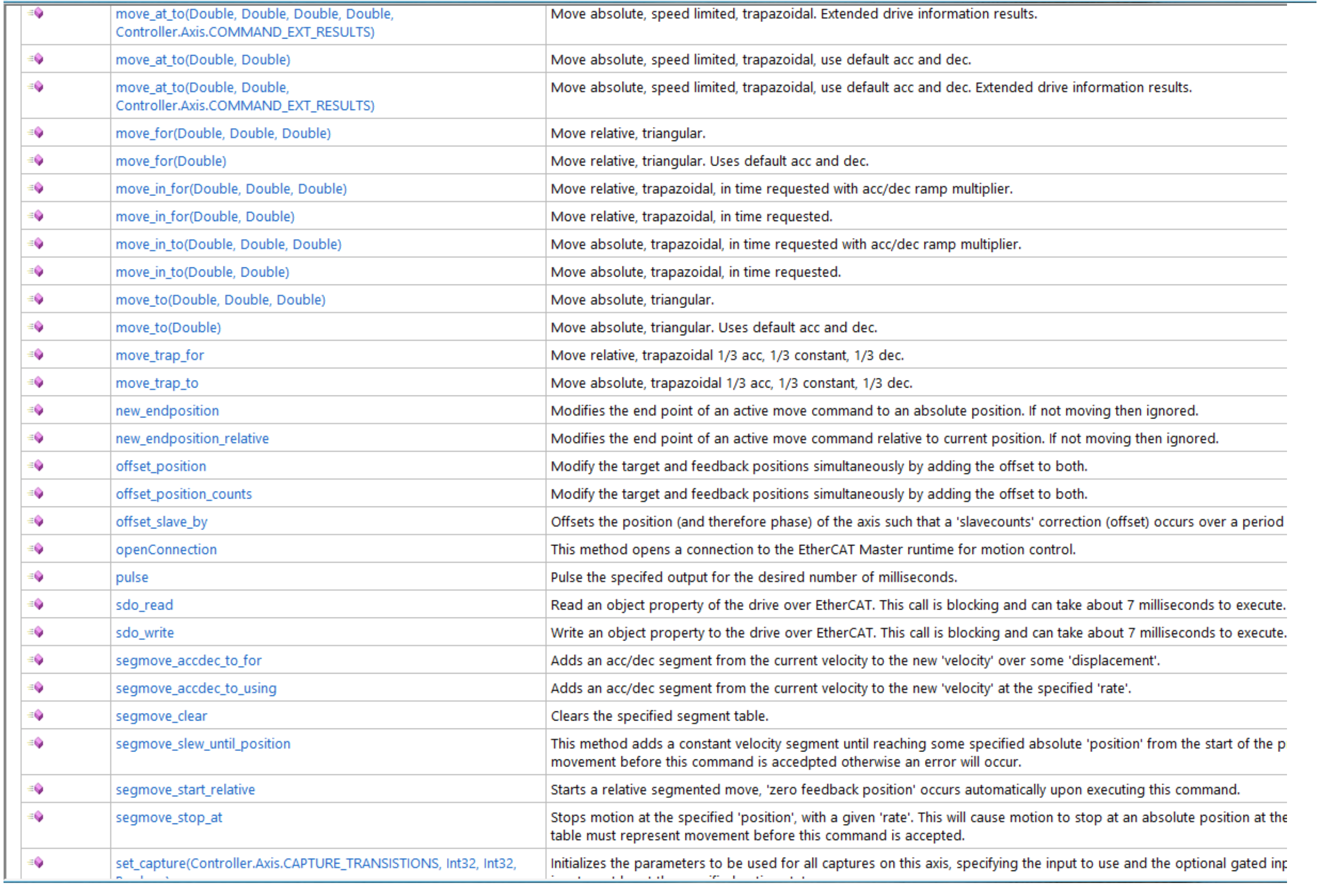


- CTC_Incentive Namespace
  - Controller Class
  - Controller.Axis Class
  - Axis.CAP_REFERENCE Enumeration
  - Axis.CAPTURE_TRANSISTIONS Enumeration
  - Axis.cbit Class
  - Axis.COMMAND_EXT_RESULTS Structure
    - Axis.COMMAND_EXT_RESULTS Members
    - Axis.COMMAND_EXT_RESULTS Fields
    - Axis.COMMAND_EXT_RESULTS Methods
  - Axis.ctr Class
  - Axis.cvar Class
  - Axis.DRIVE_TYPES Enumeration
  - Axis.ECAT_OBJECT_SIZE Enumeration
  - Axis.ETHERCAT_MODES Enumeration
  - Axis.IncentiveAxisException Class
  - Axis.MASTER_MODES Enumeration
  - Axis.MOTION_FAULTS Enumeration
  - Axis.MOVE_CMD Enumeration
  - Axis.MOVE_PARAM Structure
  - Axis.PSTATES Enumeration
  - Axis.SDO_READ_RESULTS Structure
  - Axis.TABLE_MOVES Enumeration
  - Controller.AxisSupervisor Class
  - AxisSupervisor.IncentiveAxisSupervisorException Class
  - AxisSupervisor.SCAN_STATES Enumeration
  - Controller.PLCLogic Class
  - PLCLogic.IncentivePLCException Class
  - PLCLogic.PLC_CONNECT_MODES Enumeration
  - PLCLogic.PLC_RETVAL Enumeration
  - PLCLogic.PLC_STATES Enumeration
  - PLCLogic.REGISTERS Enumeration
  - PLCLogic.SimpleStringComparer Class
  - Controller.Symbol Class
  - Variant Class
  - Variant.VARIANT_PLC_RESOURCES Structure
  - Variant.VARIANT_STORAGE Structure
  - Variant.VARIANT_STORAGE_BLOCK Structure
  - Variant.VARIANT_STORAGE_BLOCK_TRANSFER Struct
  - Variant.VARIANT_STORAGE_TRANSFER Structure

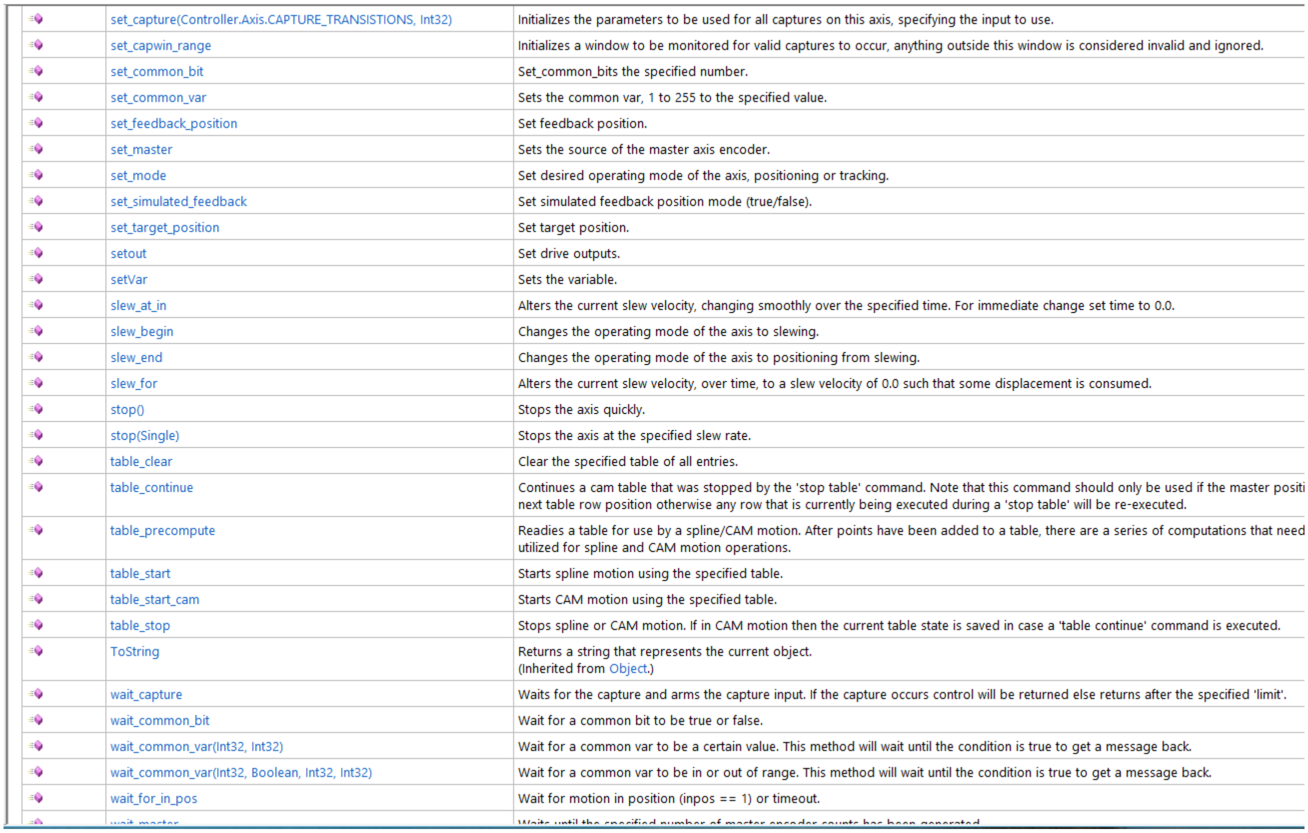
Some examples of supported motion commands:



_fpos	Read feedback position for a specific axis.
_tpos	Read target position for specific axis.
closeConnection	This method closes a connection to the EtherCAT Master runtime.
clrout	Clear drive outputs.
dc_sync	This method requests DC Sync mode with the drive attached to this axis. It does not return until completely executed and the drive
delayMSB	Delays the specified milliseconds using an MSB delay loop.
drive_disable	Disable the drive, turning voltage off to it. Drive will then free wheel and will continue to have its position tracked.
drive_enable	Enables the axis, if not already. This tells the drive to activate voltage and prepare for operation via EtherCAT.
Equals(System.Object)	Determines whether the specified Object is equal to the current Object. (Inherited from Object.)
Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)
gear_at	Changes the gear ratio of a slaved axis to the specified values.
gear_at_in	Changes the gear ratio of a slaved axis to the specified values over some number of master counts.
gear_at_in_after	Changes the gear ratio of a slaved axis to the specified values over some number of master counts.
gear_for_in	Temporarily changes the gear ratio of a slaved axis such that a slave_counts correction (offset) occurs over a master-feedback disp
gear_for_in_after	Temporarily changes the gear ratio of a slaved axis such that a slave_counts correction (offset) occurs over a master-feedback disp number of counts.
get_drive_properties	Get drive properties, reads COMMAND_EXT_RESULTS drive properties all at once versus individually.
GetHashCode	Serves as a hash function for a particular type. (Inherited from Object.)
getResources	Gets the resources.
GetType	Gets the Type of the current instance. (Inherited from Object.)
getVar	Gets the variable.
MemberwiseClone	Creates a shallow copy of the current Object. (Inherited from Object.)
move_at_for(Double, Double, Double, Double)	Move relative, speed limited, trapazoidal.
move_at_for(Double, Double, Double, Double, Controller.Axis.COMMAND_EXT_RESULTS)	Move relative, speed limited, trapazoidal. Extended drive information results.
move_at_for(Double, Double)	Move relative, speed limited, trapazoidal using default acc and dec.
move_at_for(Double, Double, Controller.Axis.COMMAND_EXT_RESULTS)	Move relative, speed limited, trapazoidal using default acc and dec. Extended drive information
move_at_to(Double, Double, Double, Double)	Move absolute, speed limited, trapazoidal.

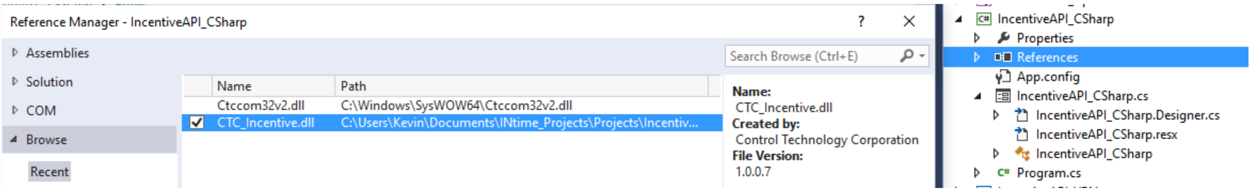


move_at_to(Double, Double, Double, Double, Controller.Axis.COMMAND_EXT_RESULTS)	Move absolute, speed limited, trapazoidal. Extended drive information results.
move_at_to(Double, Double)	Move absolute, speed limited, trapazoidal, use default acc and dec.
move_at_to(Double, Double, Controller.Axis.COMMAND_EXT_RESULTS)	Move absolute, speed limited, trapazoidal, use default acc and dec. Extended drive information results.
move_for(Double, Double, Double)	Move relative, triangular.
move_for(Double)	Move relative, triangular. Uses default acc and dec.
move_in_for(Double, Double, Double)	Move relative, trapazoidal, in time requested with acc/dec ramp multiplier.
move_in_for(Double, Double)	Move relative, trapazoidal, in time requested.
move_in_to(Double, Double, Double)	Move absolute, trapazoidal, in time requested with acc/dec ramp multiplier.
move_in_to(Double, Double)	Move absolute, trapazoidal, in time requested.
move_to(Double, Double, Double)	Move absolute, triangular.
move_to(Double)	Move absolute, triangular. Uses default acc and dec.
move_trap_for	Move relative, trapazoidal 1/3 acc, 1/3 constant, 1/3 dec.
move_trap_to	Move absolute, trapazoidal 1/3 acc, 1/3 constant, 1/3 dec.
new_endposition	Modifies the end point of an active move command to an absolute position. If not moving then ignored.
new_endposition_relative	Modifies the end point of an active move command relative to current position. If not moving then ignored.
offset_position	Modify the target and feedback positions simultaneously by adding the offset to both.
offset_position_counts	Modify the target and feedback positions simultaneously by adding the offset to both.
offset_slave_by	Offsets the position (and therefore phase) of the axis such that a 'slavecounts' correction (offset) occurs over a period
openConnection	This method opens a connection to the EtherCAT Master runtime for motion control.
pulse	Pulse the specifed output for the desired number of milliseconds.
sdo_read	Read an object property of the drive over EtherCAT. This call is blocking and can take about 7 milliseconds to execute.
sdo_write	Write an object property to the drive over EtherCAT. This call is blocking and can take about 7 milliseconds to execute.
segmove_accdec_to_for	Adds an acc/dec segment from the current velocity to the new 'velocity' over some 'displacement'.
segmove_accdec_to_using	Adds an acc/dec segment from the current velocity to the new 'velocity' at the specified 'rate'.
segmove_clear	Clears the specified segment table.
segmove_slew_until_position	This method adds a constant velocity segment until reaching some specified absolute 'position' from the start of the p movement before this command is accedpted otherwise an error will occur.
segmove_start_relative	Starts a relative segmented move, 'zero feedback position' occurs automatically upon executing this command.
segmove_stop_at	Stops motion at the specified 'position', with a given 'rate'. This will cause motion to stop at an absolute position at the table must represent movement before this command is accepted.
set_capture(Controller.Axis.CAPTURE_TRANSISTIONS, Int32, Int32,	Initializes the parameters to be used for all captures on this axis, specifying the input to use and the optional gated inp



◆	set_capture(Controller.Axis.CAPTURE_TRANSISTIONS, Int32)	Initializes the parameters to be used for all captures on this axis, specifying the input to use.
◆	set_capwin_range	Initializes a window to be monitored for valid captures to occur, anything outside this window is considered invalid and ignored.
◆	set_common_bit	Set_common_bits the specified number.
◆	set_common_var	Sets the common var, 1 to 255 to the specified value.
◆	set_feedback_position	Set feedback position.
◆	set_master	Sets the source of the master axis encoder.
◆	set_mode	Set desired operating mode of the axis, positioning or tracking.
◆	set_simulated_feedback	Set simulated feedback position mode (true/false).
◆	set_target_position	Set target position.
◆	setout	Set drive outputs.
◆	setVar	Sets the variable.
◆	slew_at_in	Alters the current slew velocity, changing smoothly over the specified time. For immediate change set time to 0.0.
◆	slew_begin	Changes the operating mode of the axis to slewing.
◆	slew_end	Changes the operating mode of the axis to positioning from slewing.
◆	slew_for	Alters the current slew velocity, over time, to a slew velocity of 0.0 such that some displacement is consumed.
◆	stop()	Stops the axis quickly.
◆	stop(Single)	Stops the axis at the specified slew rate.
◆	table_clear	Clear the specified table of all entries.
◆	table_continue	Continues a cam table that was stopped by the 'stop table' command. Note that this command should only be used if the master positi next table row position otherwise any row that is currently being executed during a 'stop table' will be re-executed.
◆	table_precompute	Readies a table for use by a spline/CAM motion. After points have been added to a table, there are a series of computations that need utilized for spline and CAM motion operations.
◆	table_start	Starts spline motion using the specified table.
◆	table_start_cam	Starts CAM motion using the specified table.
◆	table_stop	Stops spline or CAM motion. If in CAM motion then the current table state is saved in case a 'table continue' command is executed.
◆	ToString	Returns a string that represents the current object. (Inherited from Object.)
◆	wait_capture	Waits for the capture and arms the capture input. If the capture occurs control will be returned else returns after the specified 'limit'.
◆	wait_common_bit	Wait for a common bit to be true or false.
◆	wait_common_var(Int32, Int32)	Wait for a common var to be a certain value. This method will wait until the condition is true to get a message back.
◆	wait_common_var(Int32, Boolean, Int32, Int32)	Wait for a common var to be in or out of range. This method will wait until the condition is true to get a message back.
◆	wait_for_in_pos	Wait for motion in position (inpos == 1) or timeout.
◆	wait_master	Waits until the specified number of master encoder counts has been generated.

In order to access the features of the CTC Incentive API simply add the CTC_Incentive.dll resource to your Visual Studio project (Visual Studio 2013 or greater). This is done by right clicking on your project in the Solution Explorer, followed by selection of Add->Reference. Once the Reference Manager appears select Browse and browse to where you installed the dll, typically "C:\Program Files\Control Technology Corporation\CTC_IncentiveAPI\CTC_Incentive.dll", make the selection and it will be added as a Reference to your .Net project. Example projects are installed with the dll installation for your review. Reference the "CTC Incentive® .Net API User Manual" for additional details (chm help file format).



	Name	Path
	Ctccom32v2.dll	C:\Windows\SysWOW64\Ctccom32v2.dll
☑	CTC_Incentive.dll	C:\Users\Kevin\Documents\INtime_Projects\Projects\Incentiv...

Name:
CTC_Incentive.dll
Created by:
Control Technology Corporation
File Version:
1.0.0.7

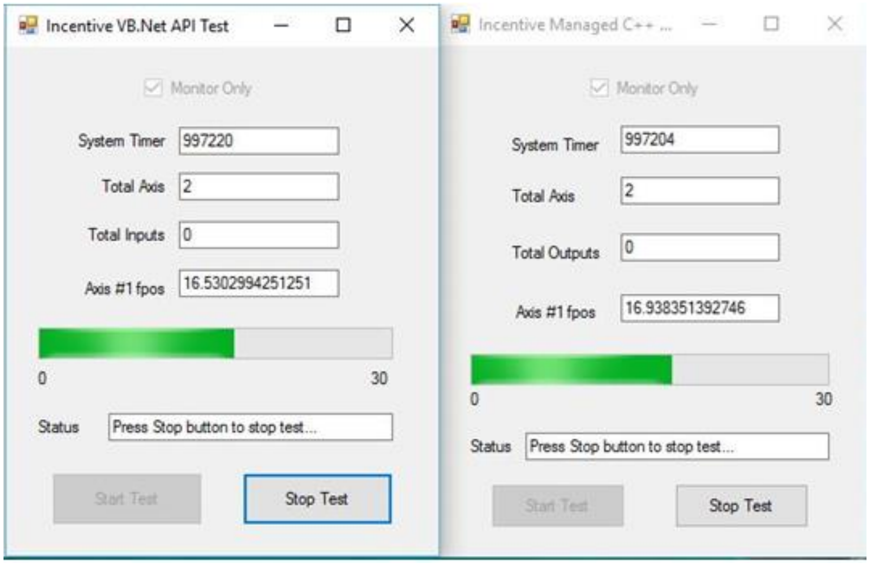
.Net API Sample Program Overview

The sample programs are available in three different languages; C#, VB.Net, and managed C++. The Monitor checkbox, when checked, allows the program to attached to the 5300 PLC environment in a passive, monitor only mode. When not checked, any QuickBuilder program will be shutdown, EtherCAT network reset and the sample program will take control of the axis motion. Only one instance should be run with the checkbox unchecked.

Upon selection of the 'Start Test' button the program will first attempt to connect to the 5300 PLCLogic process, CTPLC_1. Once connected it will check to see if 'Monitor Only' is selected. If not selected any QuickBuilder program will be stopped and the EtherCAT network reset. Next the available resources will be checked and a connection as well as a thread spawned for each axis found. A PLCLogic monitor thread will also be spawned to periodically display register 13002, system tick, as well as the network status register.

If not monitoring, each thread will initialize the servo drive for operation and begin a back and forth move, displaying the axis #1 feedback (fpos) position in a textbox. Either the 'Stop Test' button or the 'X' at the upper right of the GUI form will cause the threads to stop and connections to be released.

The sample programs themselves all perform similar operations, with a few extra in the C# example, but in different programming languages. Only one test program should control the axis at a time, the others can execute in Monitor Only mode. If you wish more than one to run the axis then changes must be made to ensure the EtherCAT network is not reset while the other application is running, else an error will result. Below shows three application test programs, one for each language monitoring the PLC system timer and axis 1.



Incentive VB.Net API Test

☑ Monitor Only

System Timer: 997220

Total Axis: 2

Total Inputs: 0

Axis #1 fpos: 16.5302994251251

Status: Press Stop button to stop test...

Incentive Managed C++ ...

☑ Monitor Only

System Timer: 997204

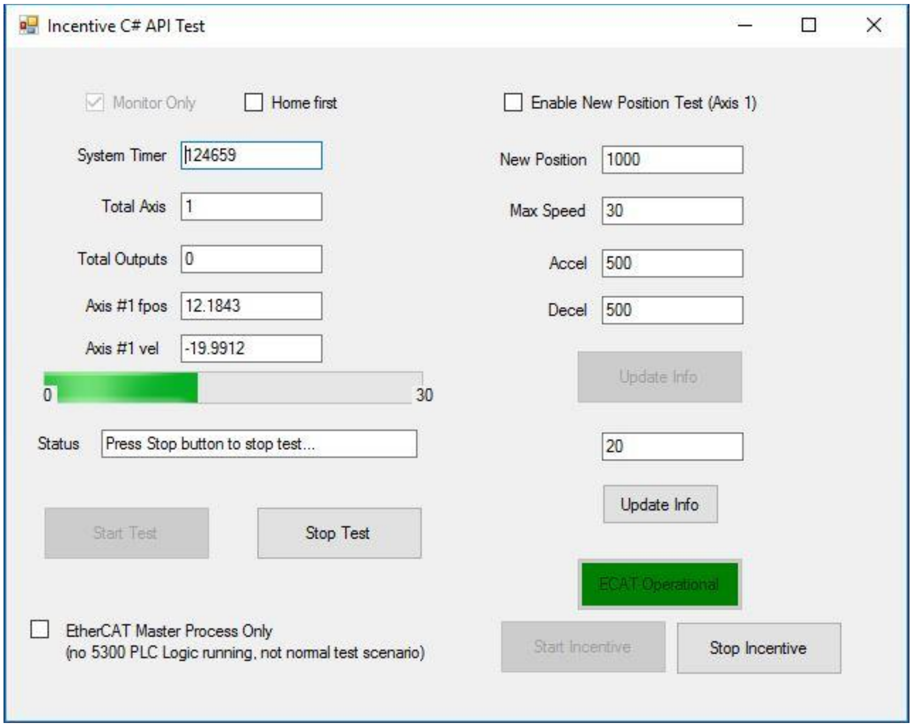
Total Axis: 2

Total Outputs: 0

Axis #1 fpos: 16.938351392746

Status: Press Stop button to stop test...

C# has been enhanced beyond that of VB.Net and C++ to demonstrate such features as running in Standalone mode (EtherCAT Master only), homing, new position commands, and starting/stopping the Incentive environment:



☑ Monitor Only □ Home first □ Enable New Position Test (Axis 1)

System Timer: 124659
Total Axis: 1
Total Outputs: 0
Axis #1 fpos: 12.1843
Axis #1 vel: -19.9912
0 – 30
Status: Press Stop button to stop test...

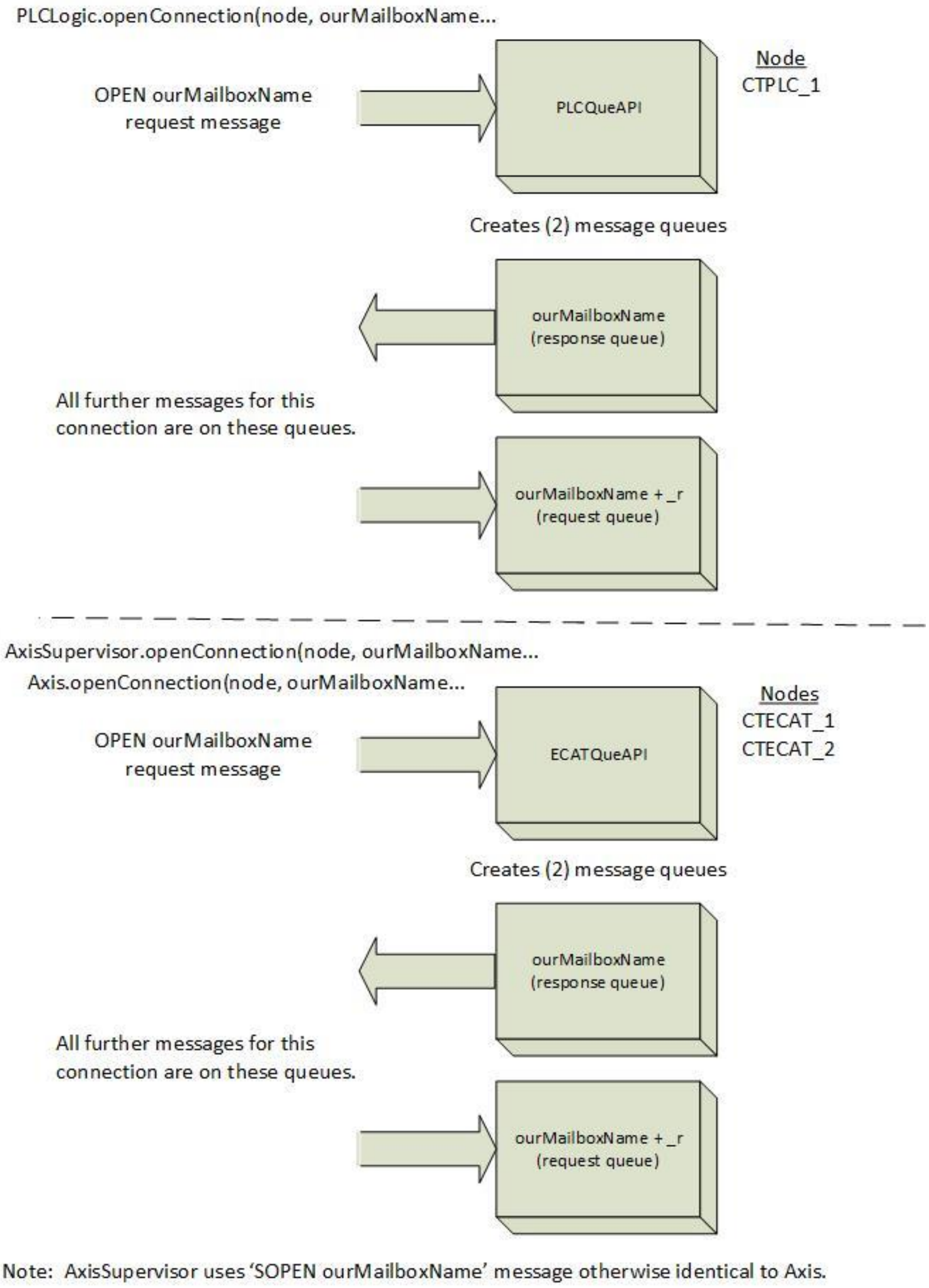
New Position: 1000
Max Speed: 30
Accel: 500
Decel: 500
20

□ EtherCAT Master Process Only
(no 5300 PLC Logic running, not normal test scenario)

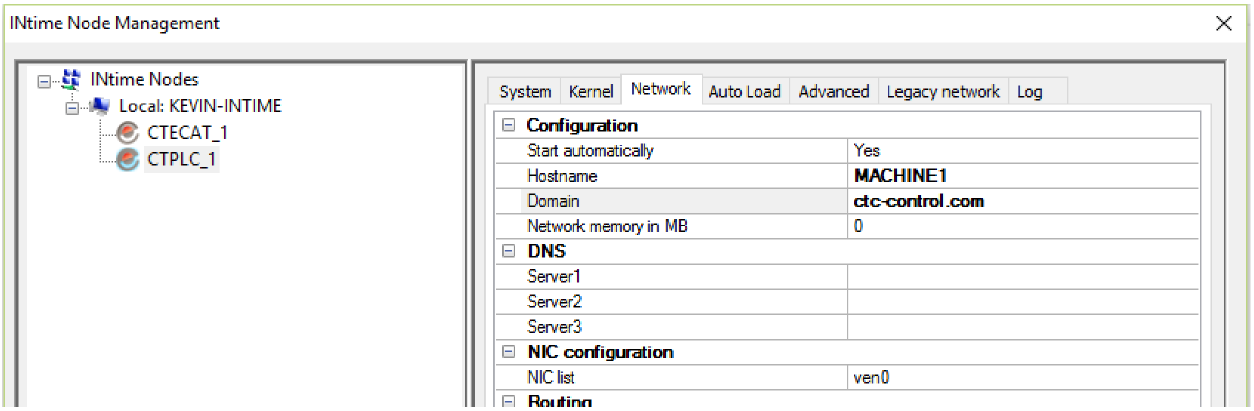
C# has the same basic tests of the other programming languages with the additions of some of the enhancements discussed. Start/Stop of Incentive realtime environment is controlled by the buttons on the lower right of the form, with the status now displaying in green, “ECAT Operational”. In an EtherCAT only environment (usually controlled directly by a high level Windows application), the checkbox on the lower right of the form should be checked. This allows the logic to then connect to an AxisSupervisor class to start and stop EtherCAT.

.Net API Opening a Connection Locally and Remote

In order to establish a connection with a runtime process the ‘openConnection’ function is invoked. This function creates two private high speed mailboxes within the Incentive runtime with which to communicate, one for sending messages and one for receiving. The parameters passed to the ‘openConnection’ differ slightly for the PLC Logic runtime, CTPLC_1, and that of the EtherCAT Master, CTECAT_1. The first parameter is the node name and the second mailbox name, common for all ‘openConnection’ functions. The mailbox name must be unique on every node for every connection since the name is published globally for each node. Different nodes can have the same mailbox name since the mapping includes the node name when addressing a mailbox.

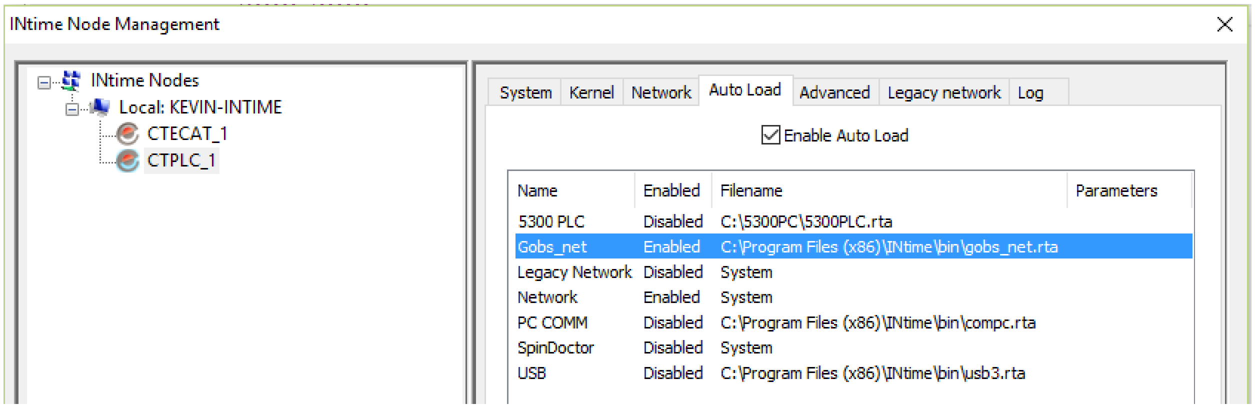

PLCLogic.openConnection(node, ourMailboxName...
OPEN ourMailboxName request message
PLCQueAPI
Node
CTPLC_1
Creates (2) message queues
ourMailboxName (response queue)
All further messages for this connection are on these queues.
ourMailboxName + _r (request queue)
AxisSupervisor.openConnection(node, ourMailboxName...
Axis.openConnection(node, ourMailboxName...
OPEN ourMailboxName request message
ECATQueAPI
Nodes
CTECAT_1
CTECAT_2
Creates (2) message queues
ourMailboxName (response queue)
All further messages for this connection are on these queues.
ourMailboxName + _r (request queue)
Note: AxisSupervisor uses 'SOPEN ourMailboxName' message otherwise identical to Axis.

When running the API locally only the node name of CTPLC_1 or CTECAT_1 is used. In order to communicate over a network the node name is expanded to include the unique Hostname assigned to CTPLC_1 within the INtime Node Management configuration form. Thus for a local connection to the PLC logic process, QuickBuilder program, the parameter would be CTPLC_1. For a remote host it would be Hostname plus any domain name entered in the configuration form followed by /CTPLC_1. For example if the Hostname was MACHINE1 with no domain information the parameter passed to 'openConnection' would be 'MACHINE1/CTPLC_1'. With a domain of ctc-control.com it would become 'MACHINE1.ctc-control.com'/CTPLC_1. Note that a Hostname must be passed to 'openConnection' not a raw IP address.



Configuration	
Start automatically	Yes
Hostname	**MACHINE1**
Domain	**ctc-control.com**
Network memory in MB	0
DNS	
Server1	
Server2	
Server3	
NIC configuration	
NIC list	ven0
Routing	

In order for automatic remote access to work a driver must also be loaded called 'gobs_net.rta'. This driver is responsible for all connections to both CTPLC_1 and CTECAT_1 and will only be enabled and loaded on the node CTPLC_1. The driver will automatically create a shared memory path to CTECAT_1.



☑ Enable Auto Load

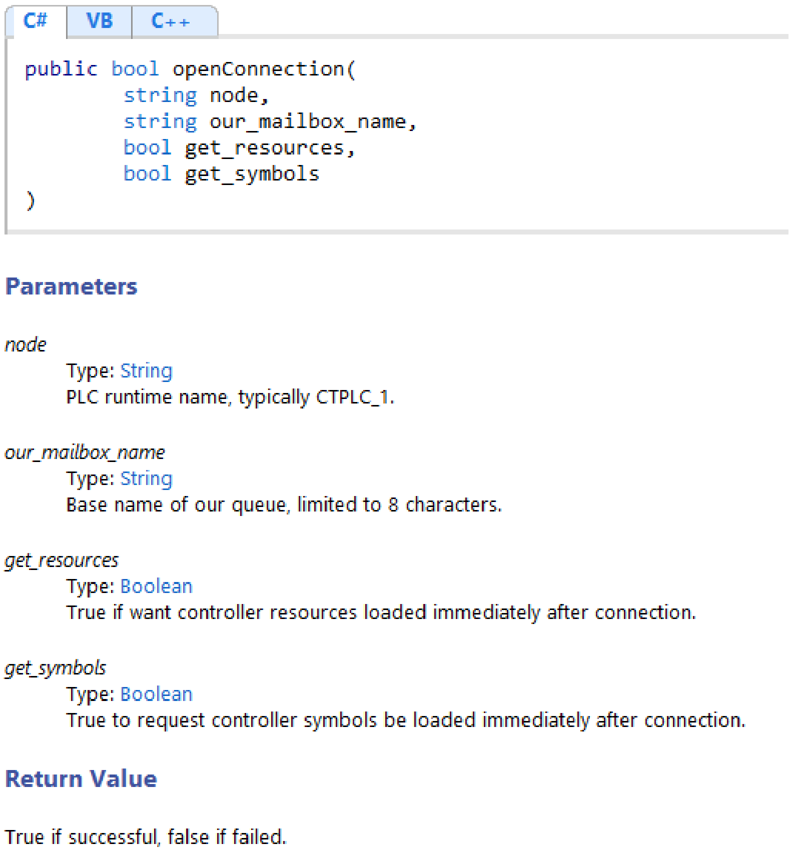
Name	Enabled	Filename	Parameters
5300 PLC	Disabled	C:\5300PC\5300PLC.rta	
Gobs_net	Enabled	C:\Program Files (x86)\INtime\bin\gobs_net.rta	
Legacy Network	Disabled	System	
Network	Enabled	System	
PC COMM	Disabled	C:\Program Files (x86)\INtime\bin\compc.rta	
SpinDoctor	Disabled	System	
USB	Disabled	C:\Program Files (x86)\INtime\bin\usb3.rta	

Gobs_net uses UDP traffic on port 48271 for communication and broadcasts of Hostname information and status periodically, therefore not relying on a DNS server or dhcp for name registration. Due to the high volume of UDP traffic it is advisable to isolate groups of computers using Gobs_net from normal LAN traffic. For example you would not want 100 computers on the same network running the peer to peer communications without using a switch to isolate some of the traffic.

Simple API Programming Concepts

PLClogic Class

All API transactions typically begin with an openConnection call (reference the Incentive.chm help file). This establishes the communication queues between the non-realtime Windows environment and the realtime Incentive. As detailed in the previous section you must specify a node name and desired unique mailbox name.



```
public bool openConnection(
        string node,
        string our_mailbox_name,
        bool get_resources,
        bool get_symbols
)
```

Parameters

node
Type: String
PLC runtime name, typically CTPLC_1.

our_mailbox_name
Type: String
Base name of our queue, limited to 8 characters.

get_resources
Type: Boolean
True if want controller resources loaded immediately after connection.

get_symbols
Type: Boolean
True to request controller symbols be loaded immediately after connection.

Return Value

True if successful, false if failed.

The PLCLogic class is used for communications with the QuickBuilder Logic environment. There you will primarily being accessing registers and variants. Both are where QuickBuilder programs store their program information. Registers are of type Integer and can also be used to access IO data and certain aspects of operation. Reference the Model 5300 Quick Reference Register Guide: http://support.ctc-control.com/customer/techinfo/docs/5300_951/951-530006.pdf. Variants store any type of information and can also consist of one and two dimensional tables, reference the QuickBuilder Reference Guide: http://support.ctc-control.com/customer/techinfo/docs/5300_951/951-530020.pdf.

For the PLCLogic class you can also load the QuickBuilder symbol table for symbolic name access, ‘get_symbols’. Typically direct numeric register access is used to expedite reads since symbols tend to slow performance. The ‘get_resources’ optionConnection option is useful to find out how much IO is present in the system before interacting with it. In a local environment the node name is always ‘CTPLC_1’, with a unique ‘our_mailbox_name’:

```
using CTC_Incentive;
…

   Controller.PLCLogic testplc = new Controller.PLCLogic();
   int value = 0;
  try
  {
     if (testplc.openConnection(“CTPLC_1”, “PQ001”, true, false))
    {
          // Connection established… Now do something, lets read system tick register 13002.
          testplc.getRegister((int)Controller.PLCLogic.REGISTERS.MILLISECOND_COUNTER, ref value);
          // ‘value’ now has the tick value read.

          // To poke the value of 1 into register 5 you could do the following.
          testplc.putRegister(5, (int)1);
    }
  }
  catch (Controller.PLCLogic.IncentivePLCException e2)
  {
          // Error processing of Incentive specific error.
          MessageBox.Show("Error occurred:  " + e2.ErrMessage);
  }
  catch (Exception)
  {
         // Handle other errors...
  }

  // When all done and ready to exit your program you should close the connection.
 testplc.closeConnection();
```

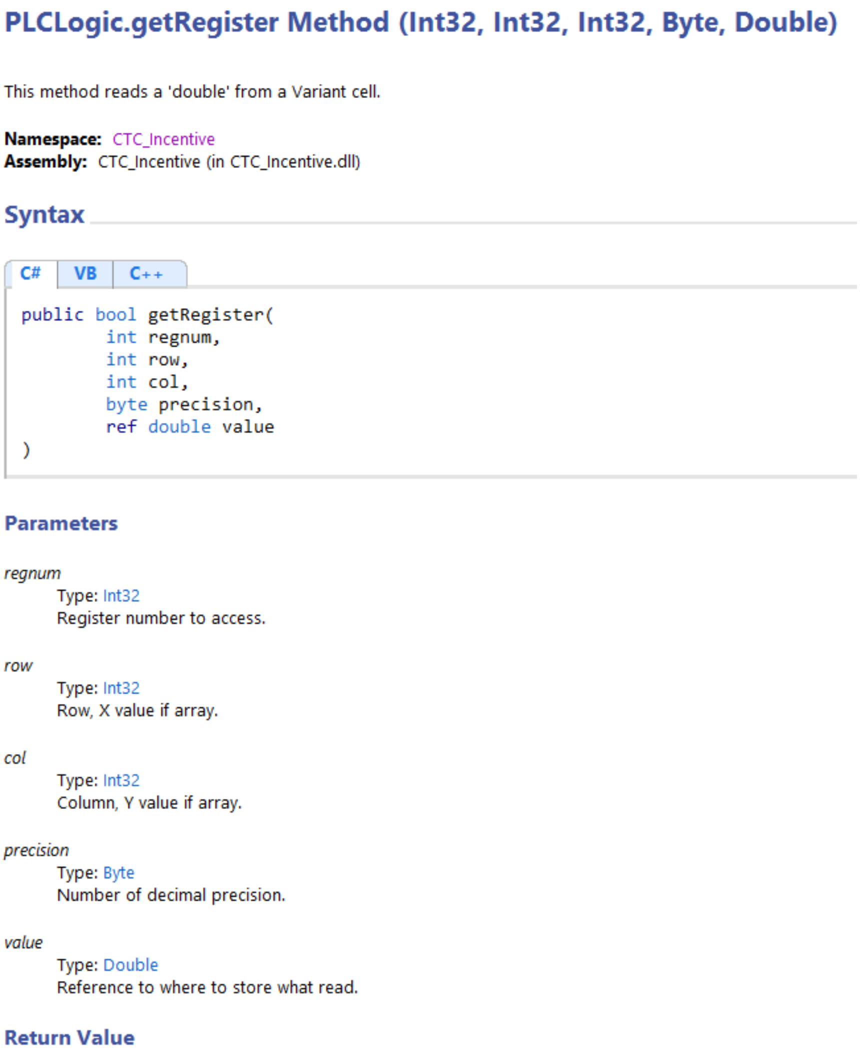
Each thread needs its own private connection; multiple connections can be made for parallel operations.

Beyond simple integer register storage there is also something called Variants. Variants can automatically assume the types you wish them do. They can be integer, double, float, and/or string. As defined in the 5300 PLC manual, certain register blocks have certain storage capabilities.

1 – 500: volatile integer registers
501-1000: non-volatile integer registers
32001-36000: non-volatile integer registers
36101 – 36700: 600 volatile Variant registers
36701 – 36800: 100 non-volatile Variant registers

Non-volatile registers are stored on the disk, and thus the importance of a UPS, so the file storage can be updated and closed when written to. Also the NTFS file system should be used, not FAT32.

Variants not only can be of any type they also can be one (vector) and two dimensional (table) arrays. The getRegister and putRegister methods have optional parameters to handle this, where the basic call is the same as previously described. The enhanced method lists the row, column, and precision:



PLCLogic.getRegister Method (Int32, Int32, Int32, Byte, Double)

This method reads a 'double' from a Variant cell.

Namespace: CTC_Incentive
Assembly: CTC_Incentive (in CTC_Incentive.dll)

Syntax

```
public bool getRegister(
        int regnum,
        int row,
        int col,
        byte precision,
        ref double value
)
```

Parameters

regnum
Type: Int32
Register number to access.

row
Type: Int32
Row, X value if array.

col
Type: Int32
Column, Y value if array.

precision
Type: Byte
Number of decimal precision.

value
Type: Double
Reference to where to store what read.

Return Value

Thus to read a double from a table, 36105[3][5], register 36105, row 3 and column 5:

```
double dValue = 0;
testplc.getRegister(36105, 3, 5, 6, ref dValue);
```

The precision by default is set to 6. Precision is only use on a read operation where a string is involved and defines the number of decimal places for the conversion. Therefore doing the same operation but in this case reading the data as a string where the double value was 26.157604532678:

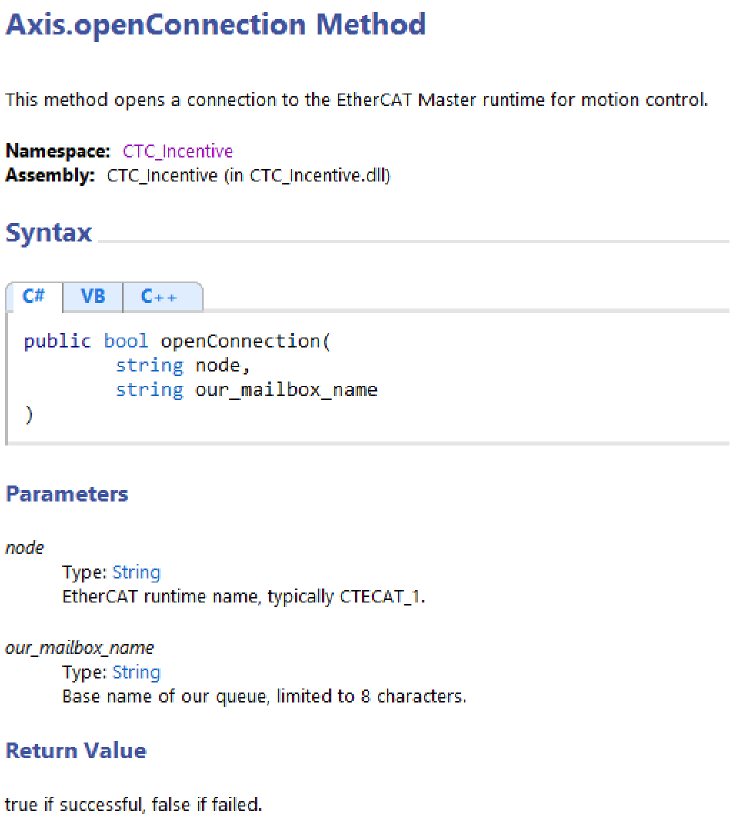
```
string sValue = 0;
testplc.getRegister(36105, 3, 5, 6, ref sValue);
```

After execution the string ‘sValue’ would contain “26.157604”.
Writing to a Variant or integer is similar to reading except instead of reference the value, it is passed to the method:

```
string sValue = “26.157604”;
testplc.putRegister(36105, 3, 5, 6, sValue);

float fValue = 2.6753;
testplc.putRegister(36105, 3, 5, 6, fValue);
```

Axis Class

Since Incentive consists of multiple realtime processes and threads we may also have a requirement to access the EtherCAT environment directly. Possibly to simply read servo position, interact with MSB (QuickBuilder Motion Sequence Blocks), or totally control motion from within our Windows program. Like the PLCLogic class there is the Axis class. The Axis class contains the entire MSB program language, mapped to Windows, as well as access to all motion and EtherCAT IO variables. An open connection is required as well to establish direct communications with the EtherCAT process node, CTECAT_1 (CTECAT_2 would be a second parallel network if used).



Axis.openConnection Method

This method opens a connection to the EtherCAT Master runtime for motion control.

Namespace: CTC_Incentive
Assembly: CTC_Incentive (in CTC_Incentive.dll)

Syntax

```
public bool openConnection(
        string node,
        string our_mailbox_name
)
```

Parameters

node
Type: String
EtherCAT runtime name, typically CTECAT_1.

our_mailbox_name
Type: String
Base name of our queue, limited to 8 characters.

Return Value

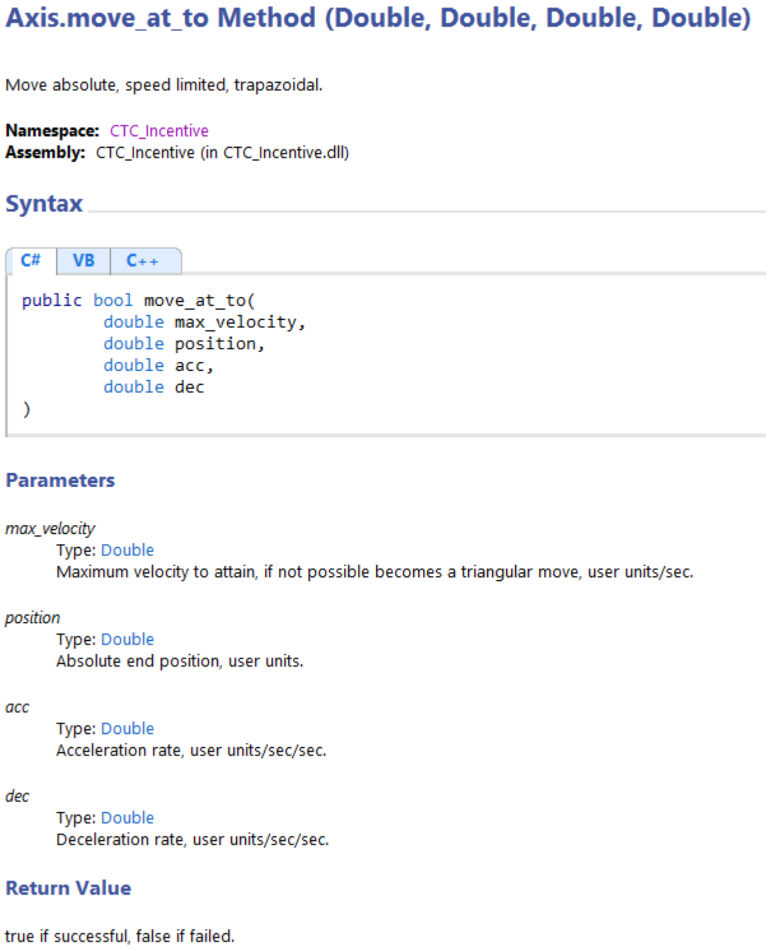
true if successful, false if failed.

In a local environment the node name is always ‘CTECAT_1’, with a unique ‘our_mailbox_name’:

```
using CTC_Incentive;
…

   Controller.Axis testAxis = new Controller.Axis();
   int value = 0;
  try
  {
     if (testAxis.openConnection(“CTECAT_1”, “MQ001”))
    {
           // Connection established…  Now do something, read the current feedback position.
           double ourPos = testAxis.fpos;

           // Let spin the motor 100 revolutions greater than present position.
           // Velocity is 30 revs/sec, acceleration 50 revs/sec2, deceleration 250 revs/sec2.
           testAxis.move_at_to(30, ourPos + 100, 50, 250);

           // Lets wait until we are in position.
           testAxis.wait_for_in_pos(-1);
    }
  }
  catch (Controller.Axis.IncentiveAxisException e2)
  {
          // Error processing of Incentive specific error.
          MessageBox.Show("Motion error occurred:  " + e2.ErrMessage);
  }
  catch (Exception)
  {
         // Handle other errors...
  }

  // When all done and ready to exit your program you should close the connection.
 testAxis.closeConnection();
```

Each thread needs its own private connection; multiple connections can be made for parallel operations. Note that each command runs as its own simulated MSB.

The Axis move_at_to command was introduced above and is described in a bit more detail below:



Axis.move_at_to Method (Double, Double, Double, Double)

Move absolute, speed limited, trapazoidal.

Namespace: CTC_Incentive
Assembly: CTC_Incentive (in CTC_Incentive.dll)

Syntax

```
public bool move_at_to(
        double max_velocity,
        double position,
        double acc,
        double dec
)
```

Parameters

max_velocity
Type: Double
Maximum velocity to attain, if not possible becomes a triangular move, user units/sec.

position
Type: Double
Absolute end position, user units.

acc
Type: Double
Acceleration rate, user units/sec/sec.

dec
Type: Double
Deceleration rate, user units/sec/sec.

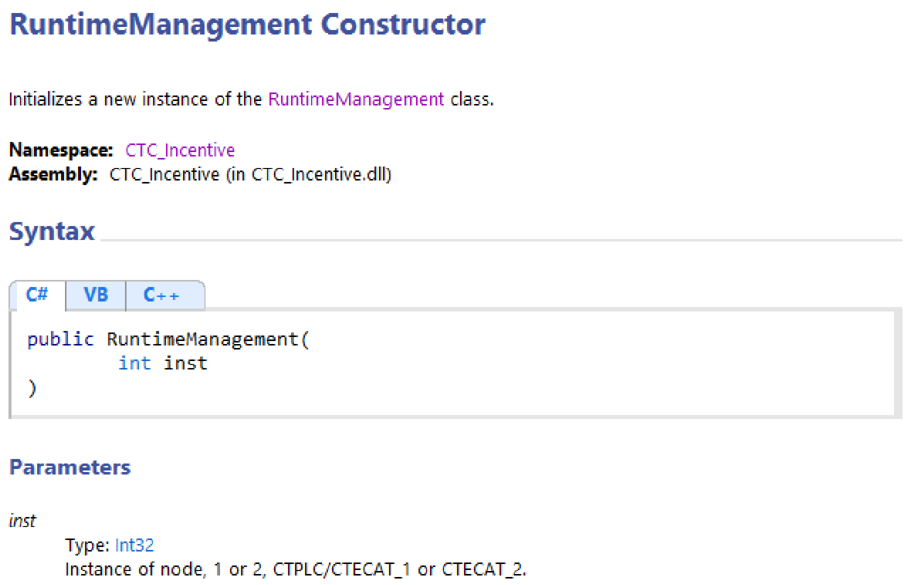
Return Value

true if successful, false if failed.

RuntimeManagement Class

The Incentive API uses a realtime operating system created by TenAsys called INtime. The API contains a system management class called RuntimeManagement. This class can be used to start and stop the INtime nodes as well as determine what state they are in with regards to EtherCAT being executed.

The updateCurrentStateInformation method updates the class properties EtherCAT_started, Plclogic_started, and SameCore with the proper current values:



RuntimeManagement Constructor

Initializes a new instance of the RuntimeManagement class.

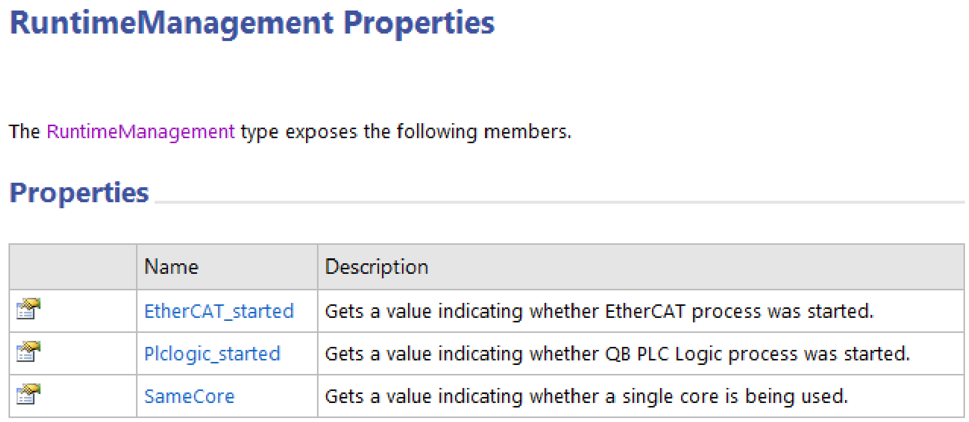
Namespace: CTC_Incentive
Assembly: CTC_Incentive (in CTC_Incentive.dll)

Syntax

```
public RuntimeManagement(
        int inst
)
```

Parameters

inst
Type: Int32
Instance of node, 1 or 2, CTPLC/CTECAT_1 or CTECAT_2.

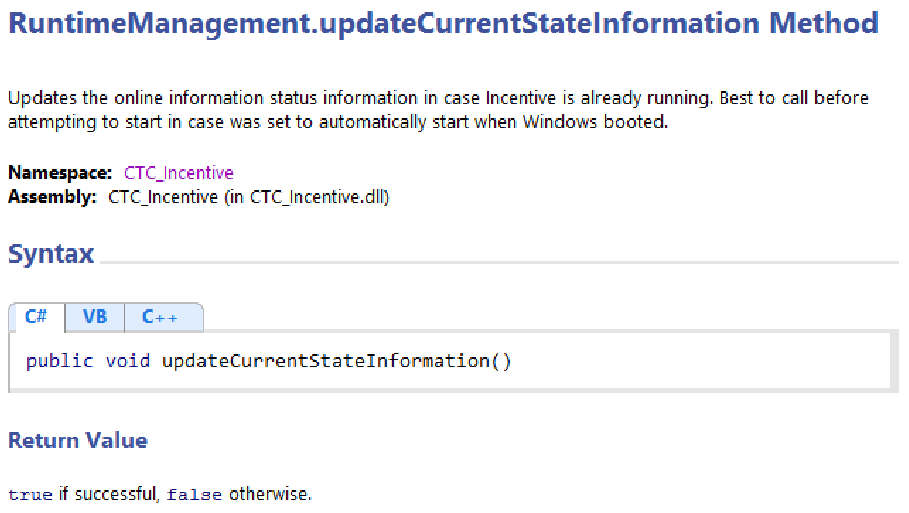


RuntimeManagement Properties

The RuntimeManagement type exposes the following members.

Properties

	Name	Description
	EtherCAT_started	Gets a value indicating whether EtherCAT process was started.
	Plclogic_started	Gets a value indicating whether QB PLC Logic process was started.
	SameCore	Gets a value indicating whether a single core is being used.



RuntimeManagement.updateCurrentStateInformation Method

Updates the online information status information in case Incentive is already running. Best to call before attempting to start in case was set to automatically start when Windows booted.

Namespace: CTC_Incentive
Assembly: CTC_Incentive (in CTC_Incentive.dll)

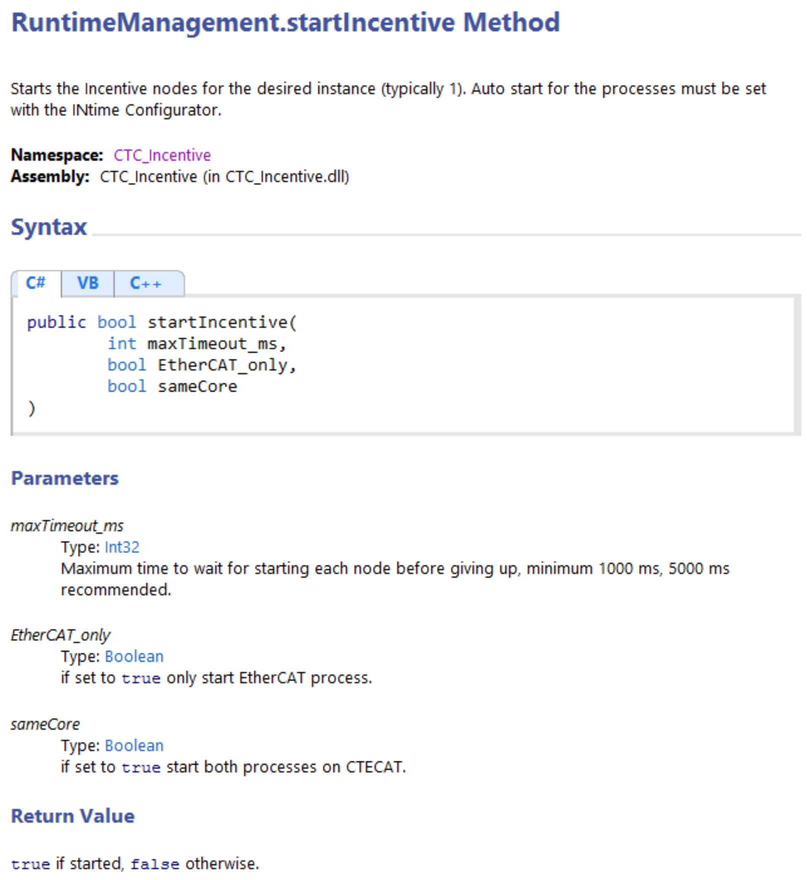
Syntax

```
public void updateCurrentStateInformation()
```

Return Value

`true` if successful, `false` otherwise.

The start/stop methods are equally as simple:



RuntimeManagement.startIncentive Method

Starts the Incentive nodes for the desired instance (typically 1). Auto start for the processes must be set with the INtime Configurator.

Namespace: CTC_Incentive
Assembly: CTC_Incentive (in CTC_Incentive.dll)

Syntax

```
public bool startIncentive(
        int maxTimeout_ms,
        bool EtherCAT_only,
        bool sameCore
)
```

Parameters

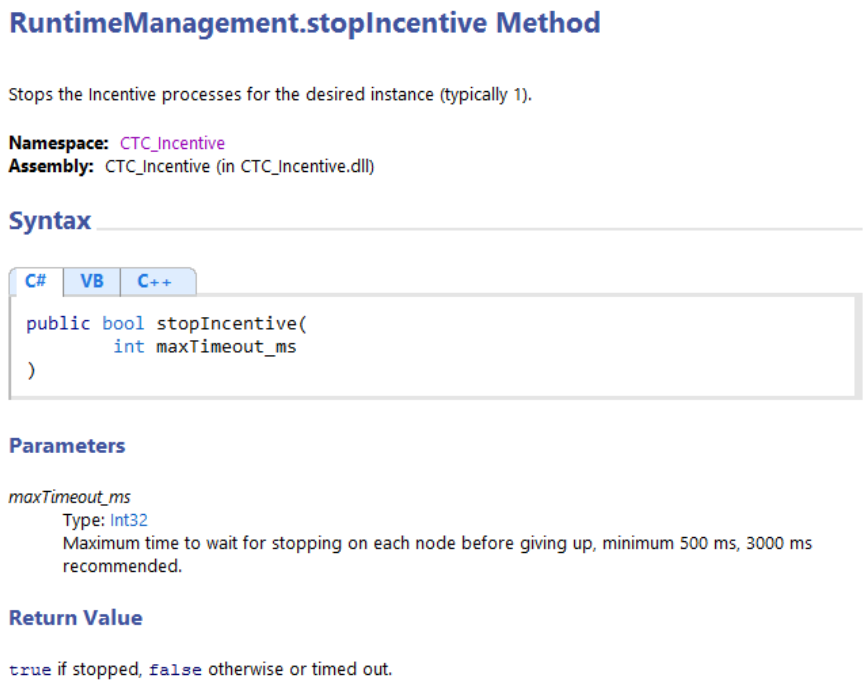
maxTimeout_ms
Type: Int32
Maximum time to wait for starting each node before giving up, minimum 1000 ms, 5000 ms recommended.

EtherCAT_only
Type: Boolean
if set to `true` only start EtherCAT process.

sameCore
Type: Boolean
if set to `true` start both processes on CTECAT.

Return Value

`true` if started, `false` otherwise.



RuntimeManagement.stopIncentive Method

Stops the Incentive processes for the desired instance (typically 1).

Namespace: CTC_Incentive
Assembly: CTC_Incentive (in CTC_Incentive.dll)

Syntax

```
public bool stopIncentive(
        int maxTimeout_ms
)
```

Parameters

maxTimeout_ms
Type: Int32
Maximum time to wait for stopping on each node before giving up, minimum 500 ms, 3000 ms recommended.

Return Value

`true` if stopped, `false` otherwise or timed out.

```
using CTC_Incentive;
```

```
…

  Controller.RuntimeManagement RuntimeSystem = new Controller.RuntimeManagement(1);
  // Update the latest node execution information.
  RuntimeSystem.updateCurrentStateInformation();
  If (Plclogic_started == false)
  {
      // Start the INtime kernel on their nodes with a 5 second timeout.
      RuntimeSystem.startIncentive(5000);
  }
```

Hot Swap of EtherCAT Devices

The Incentive API contains methods to support the setting of slave devices to an online or offline state within a fully active network. This can be useful when problems arise and a device needs to be replaced without shutting the network down, such as a production environment where multiple lines are operating in parallel.

Without any special provisions EtherCAT devices are daisy chained, into one device, out and into the next. This means once you break the daisy chain all following devices will not respond. If a device does not respond it is an error condition to Incentive and a network fault will occur. The way around this is to notify Incentive the device is being taken offline and allow it to prepare for it.

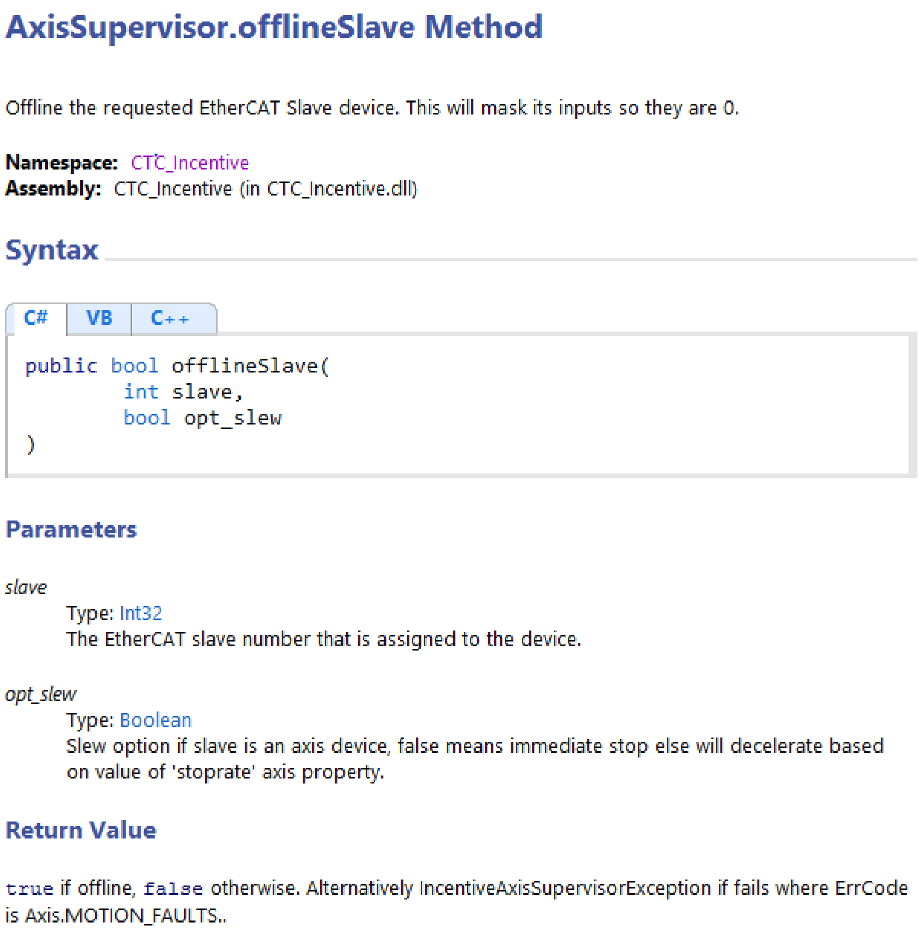
Certain guidelines must be followed to have this work properly and prevent devices from faulting or gaps in EtherCAT cabling causing packet loss:

1. ***All Online at Startup*** - It is important that when the EtherCAT network is first brought online, all devices are present. This allows Incentive to establish the maximum required packet size for communications and the data placement for each within a packet.
2. ***Offline all following devices*** – If removing a device you must offline that device and all following in the daisy chain before performing service.
3. ***Omron EtherCAT Junction GS-JC06*** – To fully take advantage of this feature it is best to use an Omron EtherCAT junction as the first slave device. This device provides up to 5 independent EtherCAT segments that are self-healing as well as a 64 bit clock for distribution. If you disconnect a slave device on one of the segments the network can continue to operate the other segments. The same rules apply with daisy chaining, all devices after the device you are powering off must be offline as well or an error will result, with the EtherCAT junction you have up to 5 independent segments instead of just one.

When a device is offline it is placed in its EtherCAT INIT state, outputs (off) do not change or work, and data is not available from the device. It does replicate EtherCAT packets to the next device. It is only when you power off the device or disconnect the EtherCAT cable that communications is affected. Normally each slave device increments a counter in the packet, by warning Incentive you are disconnecting devices it will know how many counts are allowed to be missing in the packet before it is a real error.

AxisSupervisor Class – online/offline Methods

The AxisSupervisor Class is used to interface with EtherCAT at a system level. API functionality such as restarting an EtherCAT network or placing a slave device online or offline is available. These methods are as follows:



AxisSupervisor.offlineSlave Method

Offline the requested EtherCAT Slave device. This will mask its inputs so they are 0.

Namespace: CTC_Incentive
Assembly: CTC_Incentive (in CTC_Incentive.dll)

Syntax

```
public bool offlineSlave(
	int slave,
	bool opt_slew
)
```

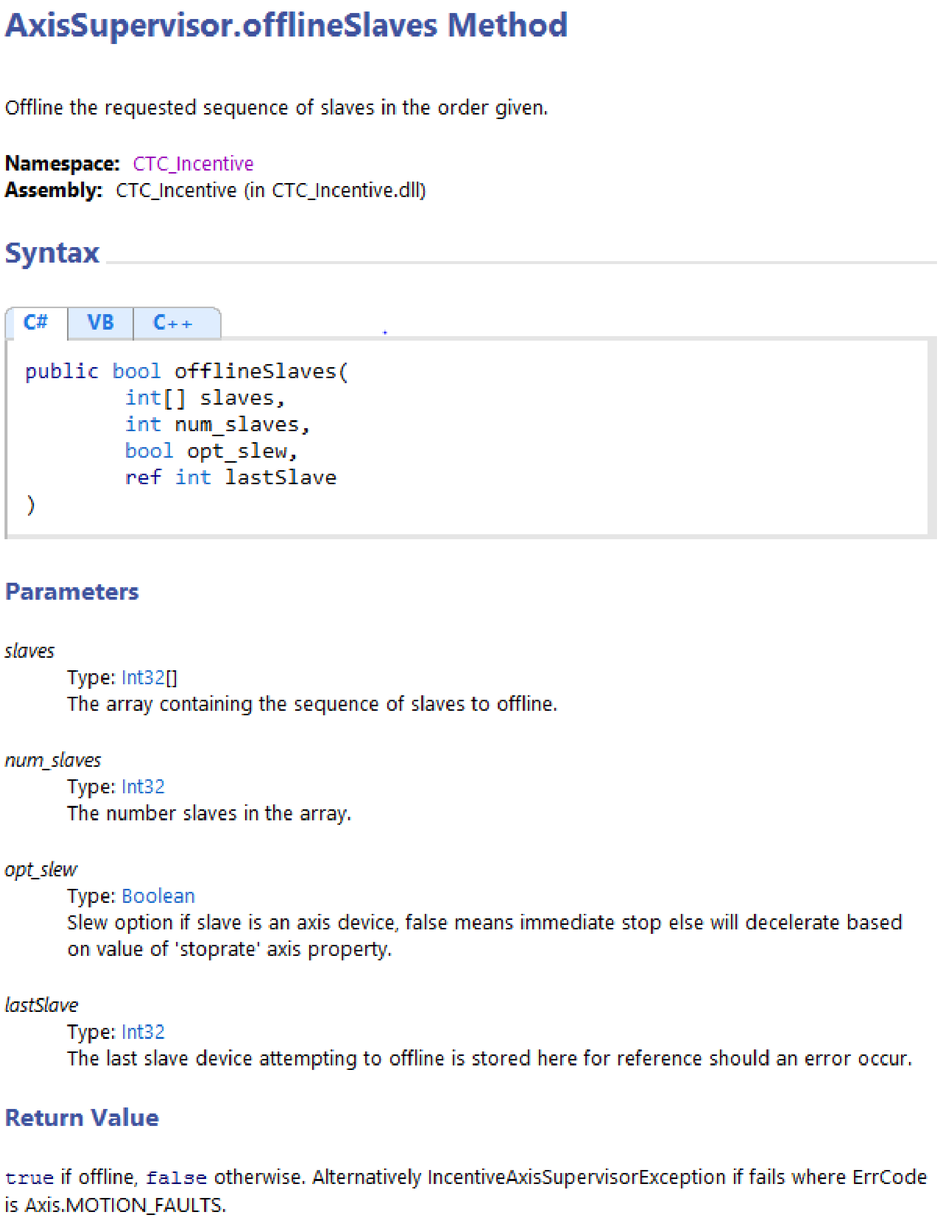
Parameters

slave
Type: Int32
The EtherCAT slave number that is assigned to the device.

opt_slew
Type: Boolean
Slew option if slave is an axis device, false means immediate stop else will decelerate based on value of 'stoprate' axis property.

Return Value

`true` if offline, `false` otherwise. Alternatively IncentiveAxisSupervisorException if fails where ErrCode is Axis.MOTION_FAULTS..



AxisSupervisor.offlineSlaves Method

Offline the requested sequence of slaves in the order given.

Namespace: CTC_Incentive
Assembly: CTC_Incentive (in CTC_Incentive.dll)

Syntax

C# VB C++

```
public bool offlineSlaves(
        int[] slaves,
        int num_slaves,
        bool opt_slew,
        ref int lastSlave
)
```

Parameters

slaves
Type: Int32[]
The array containing the sequence of slaves to offline.

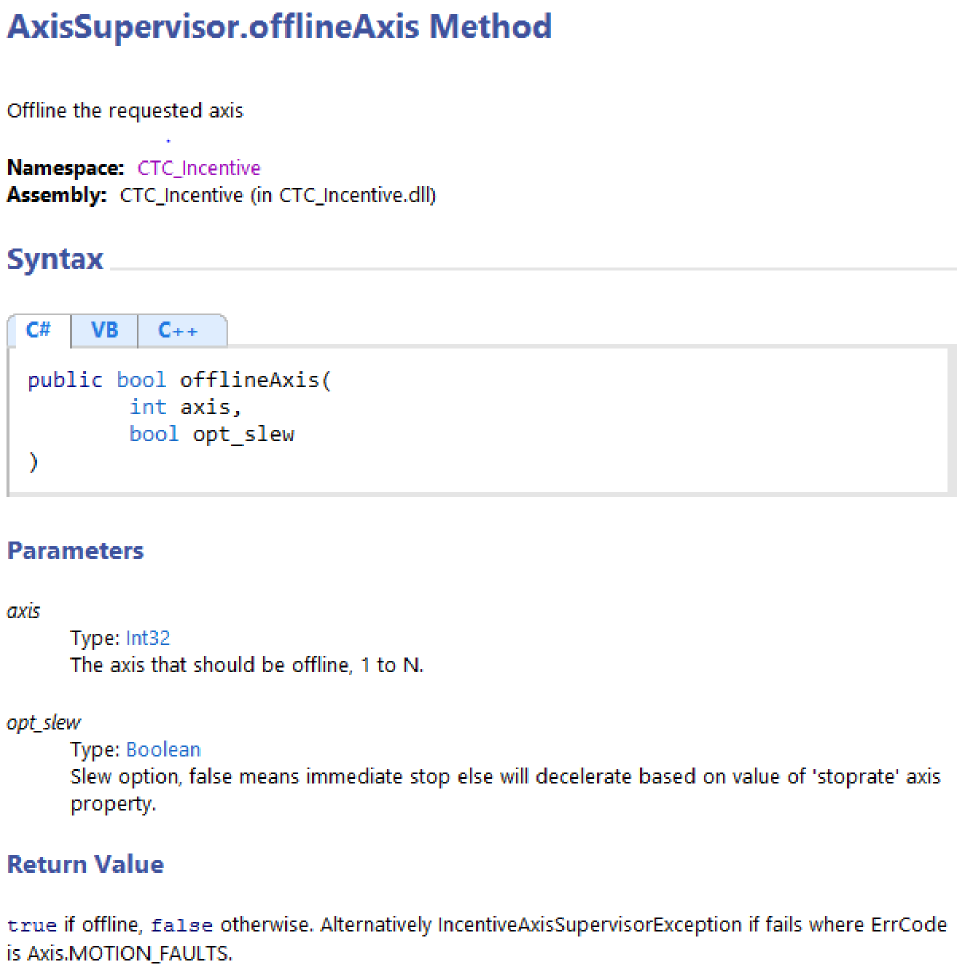
num_slaves
Type: Int32
The number slaves in the array.

opt_slew
Type: Boolean
Slew option if slave is an axis device, false means immediate stop else will decelerate based on value of 'stoprate' axis property.

lastSlave
Type: Int32
The last slave device attempting to offline is stored here for reference should an error occur.

Return Value

`true` if offline, `false` otherwise. Alternatively IncentiveAxisSupervisorException if fails where ErrCode is Axis.MOTION_FAULTS.



AxisSupervisor.offlineAxis Method

Offline the requested axis

Namespace: CTC_Incentive
Assembly: CTC_Incentive (in CTC_Incentive.dll)

Syntax

C# | VB | C++

```
public bool offlineAxis(
        int axis,
        bool opt_slew
)
```

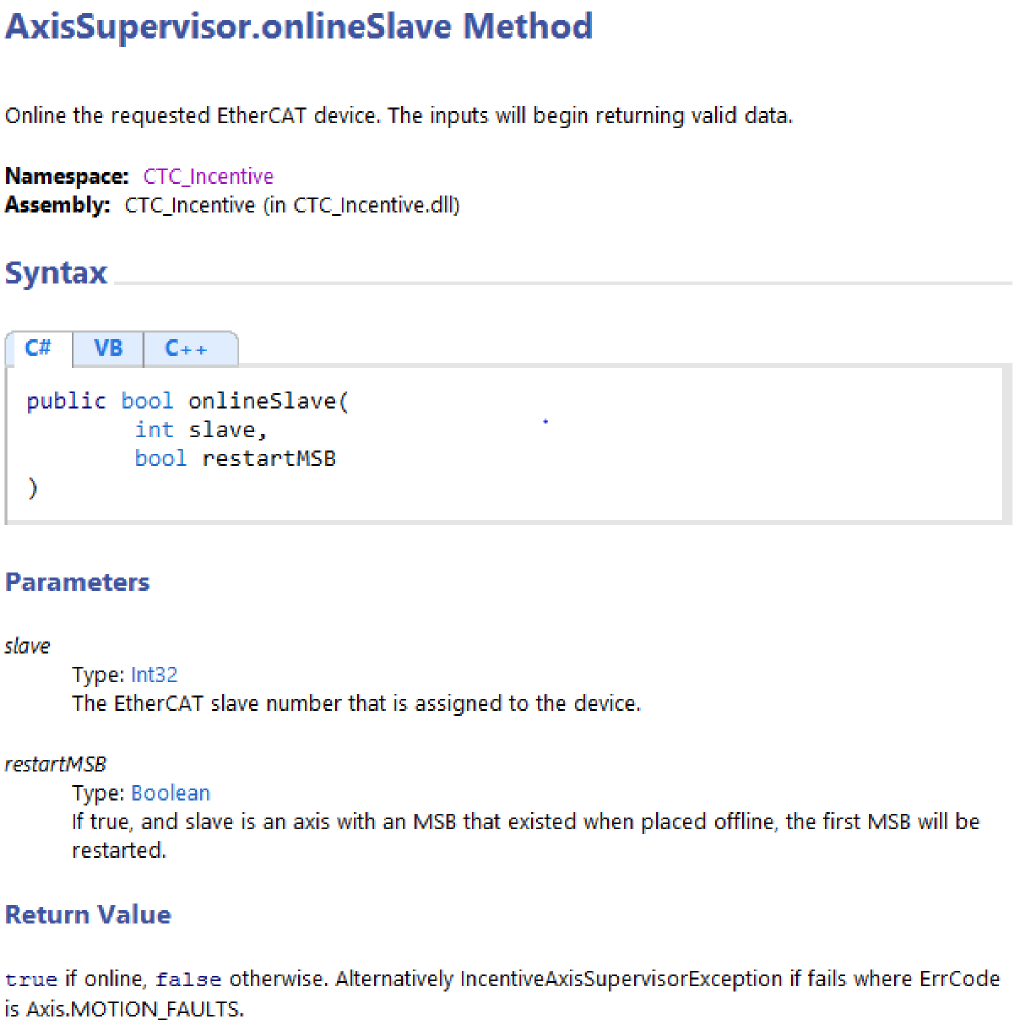
Parameters

axis
Type: Int32
The axis that should be offline, 1 to N.

opt_slew
Type: Boolean
Slew option, false means immediate stop else will decelerate based on value of 'stoprate' axis property.

Return Value

`true` if offline, `false` otherwise. Alternatively IncentiveAxisSupervisorException if fails where ErrCode is Axis.MOTION_FAULTS.



AxisSupervisor.onlineSlave Method

Online the requested EtherCAT device. The inputs will begin returning valid data.

Namespace: CTC_Incentive
Assembly: CTC_Incentive (in CTC_Incentive.dll)

Syntax

```
public bool onlineSlave(
        int slave,
        bool restartMSB
)
```

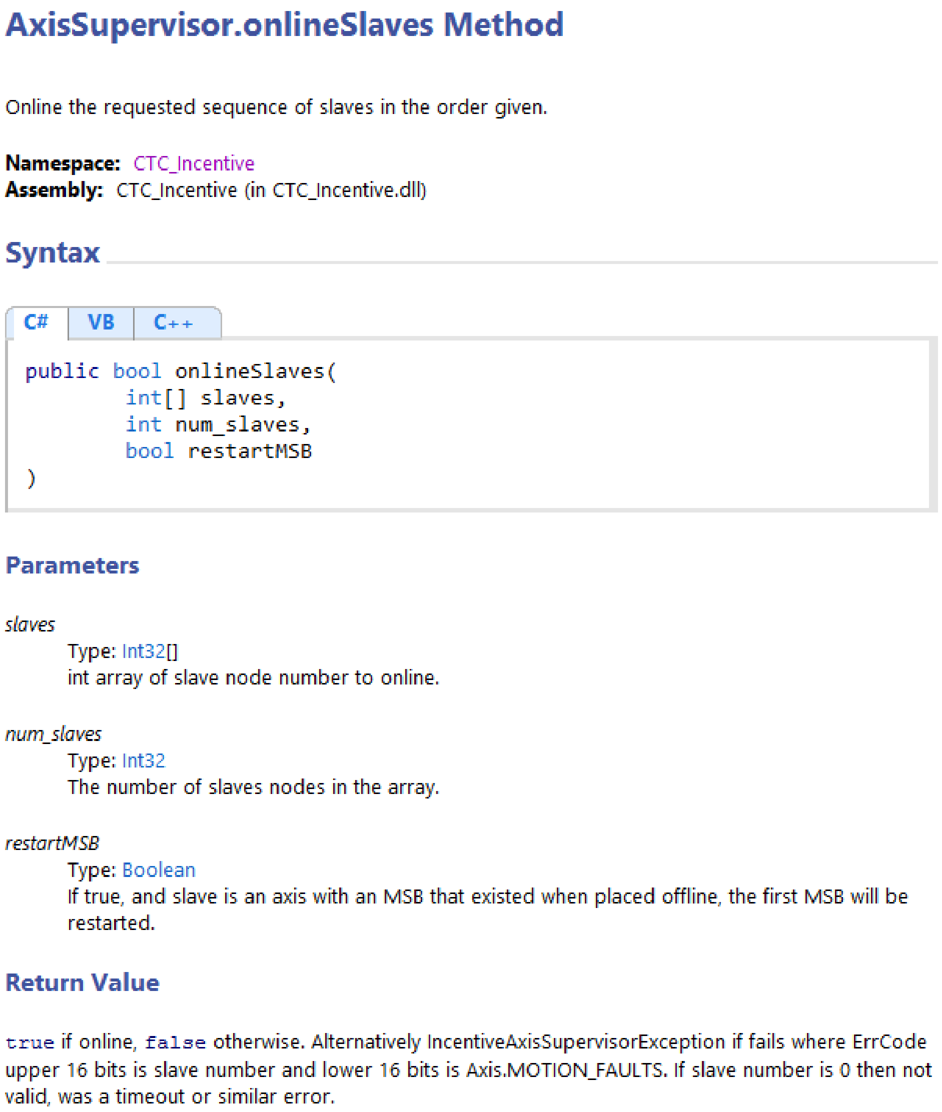
Parameters

slave
Type: Int32
The EtherCAT slave number that is assigned to the device.

restartMSB
Type: Boolean
If true, and slave is an axis with an MSB that existed when placed offline, the first MSB will be restarted.

Return Value

`true` if online, `false` otherwise. Alternatively IncentiveAxisSupervisorException if fails where ErrCode is Axis.MOTION_FAULTS.



AxisSupervisor.onlineSlaves Method

Online the requested sequence of slaves in the order given.

Namespace: CTC_Incentive
Assembly: CTC_Incentive (in CTC_Incentive.dll)

Syntax

```
public bool onlineSlaves(
        int[] slaves,
        int num_slaves,
        bool restartMSB
)
```

Parameters

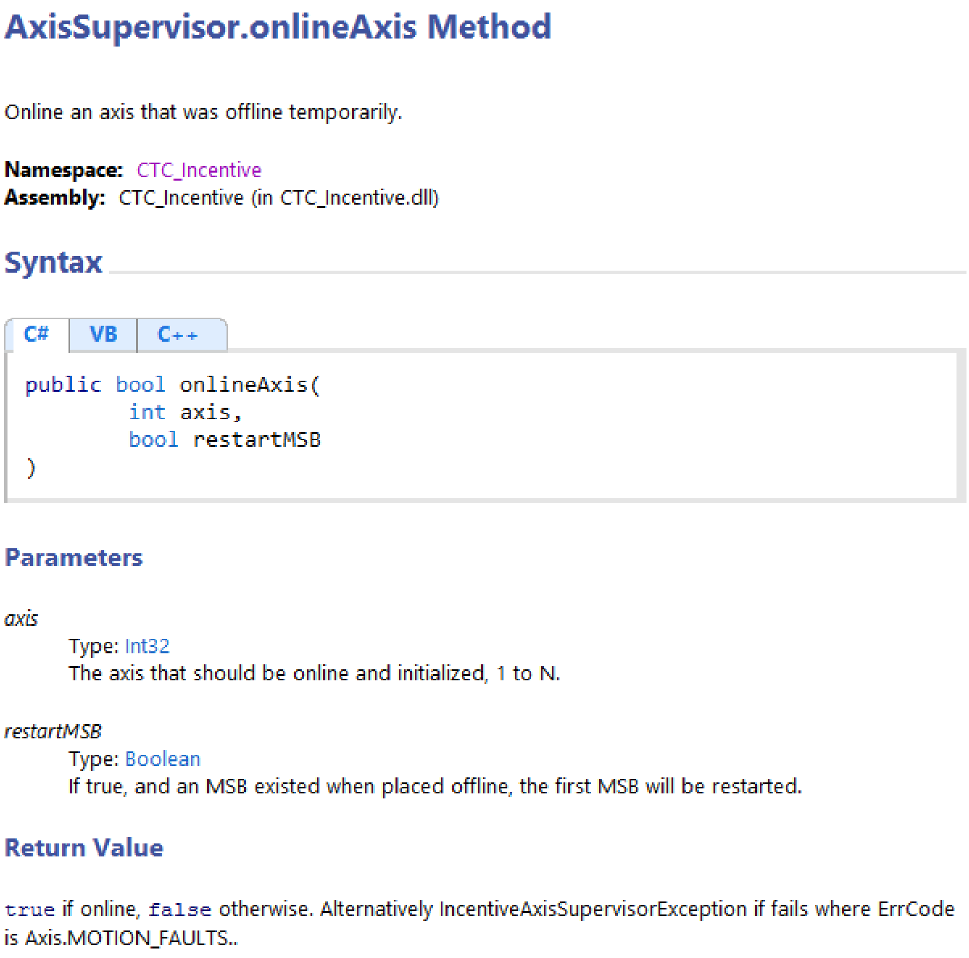
slaves
Type: Int32[]
int array of slave node number to online.

num_slaves
Type: Int32
The number of slaves nodes in the array.

restartMSB
Type: Boolean
If true, and slave is an axis with an MSB that existed when placed offline, the first MSB will be restarted.

Return Value

`true` if online, `false` otherwise. Alternatively IncentiveAxisSupervisorException if fails where ErrCode upper 16 bits is slave number and lower 16 bits is Axis.MOTION_FAULTS. If slave number is 0 then not valid, was a timeout or similar error.



AxisSupervisor.onlineAxis Method

Online an axis that was offline temporarily.

Namespace: CTC_Incentive
Assembly: CTC_Incentive (in CTC_Incentive.dll)

Syntax

```
public bool onlineAxis(
        int axis,
        bool restartMSB
)
```

Parameters

axis
Type: Int32
The axis that should be online and initialized, 1 to N.

restartMSB
Type: Boolean
If true, and an MSB existed when placed offline, the first MSB will be restarted.

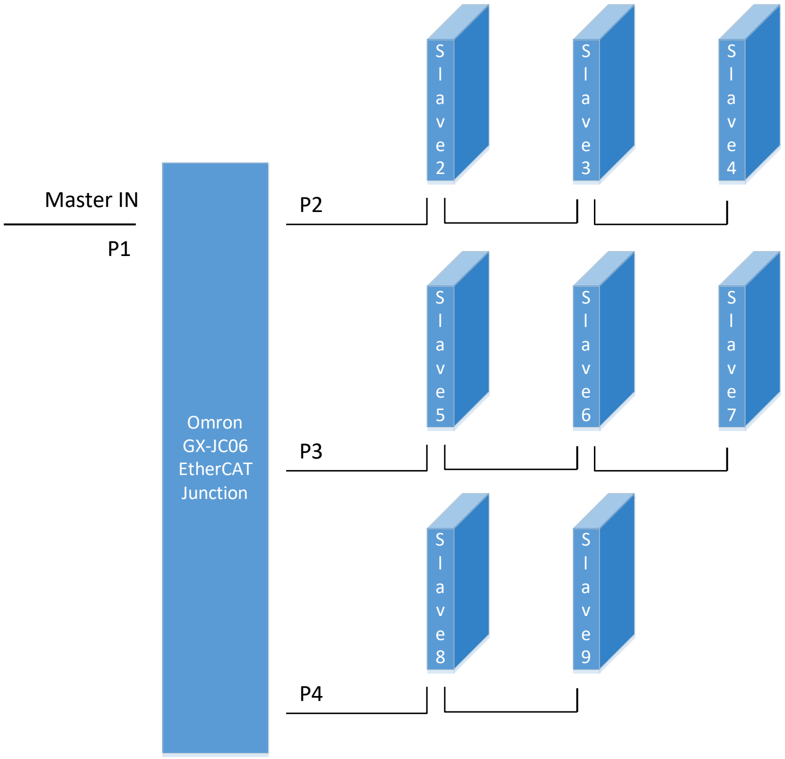
Return Value

`true` if online, `false` otherwise. Alternatively IncentiveAxisSupervisorException if fails where ErrCode is Axis.MOTION_FAULTS..

If you would like to know the slave # of a particular axis you can reference that axis property dwSlaveID. Additionally, if you would like an axis to always be assigned to a specific drive reference the 'Station Alias' capability in chapter 3 of this manual. Be careful when you are replacing a drive and are using the alias feature, don't install a new drive with the wrong alias or unknown motion could occur with more than one drive assigned the same axis number.

Offline/Online Example

Below is an example of a network with the first slave device being the Omron EtherCAT Junction, GX-JC06. The slave devices are daisy chained on each port, 5 are available, each acting as a separate network segment.



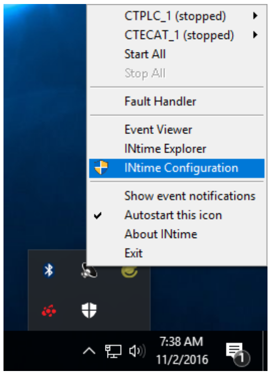
With the example configuration we could execute the offline on each slave in a group (P2, 3, or 4) and then power down that group for service. To bring it back online simply power the devices back up, in the same order and execute the online method.

```
int[] slaves = {5, 6, 7};
int lastSlave = 0;
offlineSlaves(slaves, 3, true, ref lastSlave);  // Offline the 3 slave devices, slew first if axis.

… service is performed…

onlineSlaves(slaves, 3, true);  // Online the 3 slave devices and restart any first MSB.
```

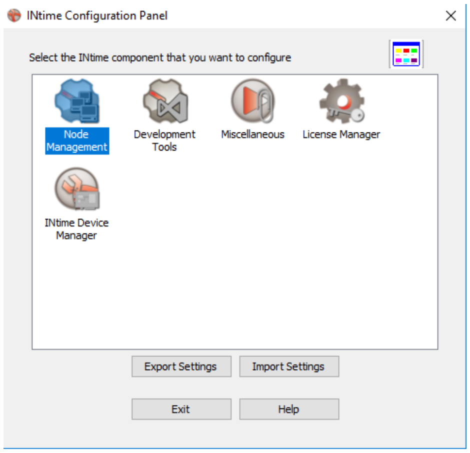
Removal of CTPLC_1 for Standalone Operation

When running in standalone operation the EtherCAT Master interfaces with a Windows program directly and QuickBuilder is not present. In this mode we can run on a single core with just the EtherCAT_Master.rta program running, being driven by the Incentive API. It is assumed that two cores have been reserved for Incentive and you wish to remove CTPLC_1, freeing it up for Windows operation.

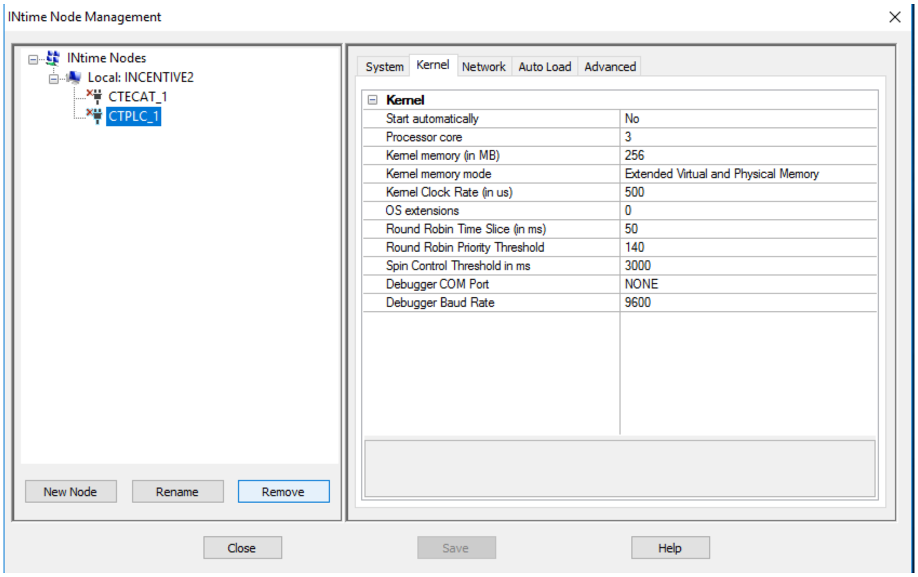
Begin by invoking the INtime Configurator, right clicking on the olive green INtime icon and selecting INtime Configuration:



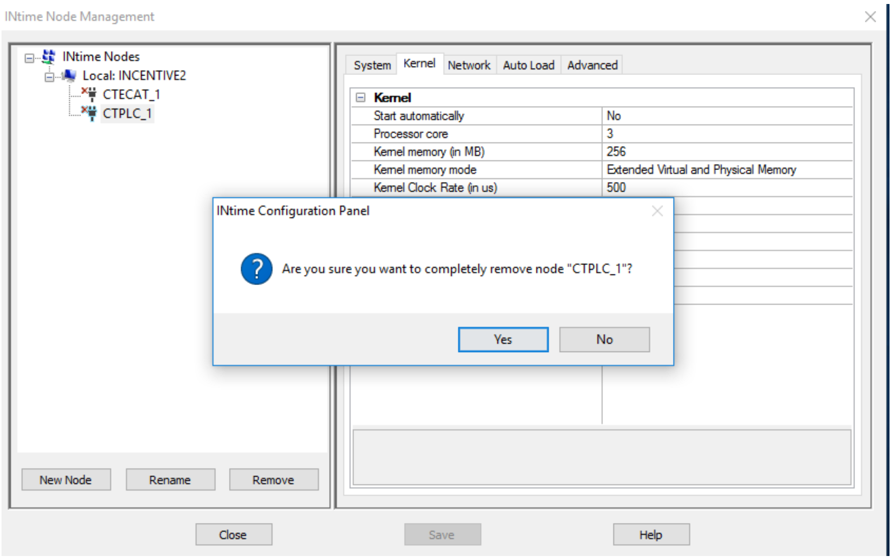
Select Node Management:



Once the Node Management screen appears select CTPLC_1, followed by Remove. Select ‘Yes’ when prompted “Are you sure?”:



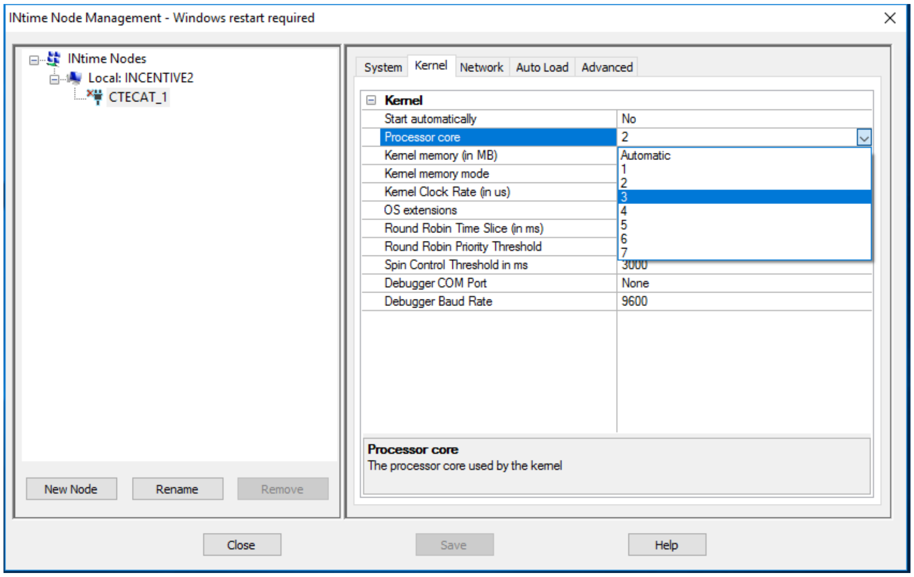
Kernel	
Start automatically	No
Processor core	3
Kernel memory (in MB)	256
Kernel memory mode	Extended Virtual and Physical Memory
Kernel Clock Rate (in us)	500
OS extensions	0
Round Robin Time Slice (in ms)	50
Round Robin Priority Threshold	140
Spin Control Threshold in ms	3000
Debugger COM Port	NONE
Debugger Baud Rate	9600



Kernel	
Start automatically	No
Processor core	3
Kernel memory (in MB)	256
Kernel memory mode	Extended Virtual and Physical Memory
Kernel Clock Rate (in us)	500

Are you sure you want to completely remove node "CTPLC_1"?

Select the CTECAT_1 node and change the Processor core to 3, if it is not already that, and click 'Save':

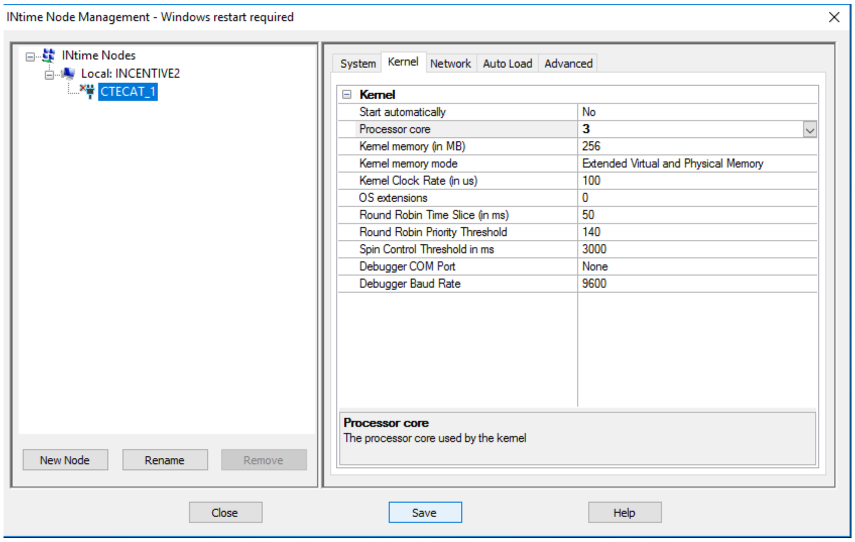


INtime Node Management - Windows restart required

- INtime Nodes
  - Local: INCENTIVE2
    - CTECAT_1

Kernel	
Start automatically	No
Processor core	2
Kernel memory (in MB)	
Kernel memory mode	
Kernel Clock Rate (in us)	
OS extensions	
Round Robin Time Slice (in ms)	
Round Robin Priority Threshold	
Spin Control Threshold in ms	3000
Debugger COM Port	None
Debugger Baud Rate	9600

Processor core
The processor core used by the kernel



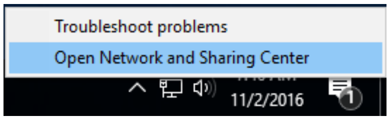
INtime Node Management - Windows restart required

- INtime Nodes
  - Local: INCENTIVE2
    - CTECAT_1

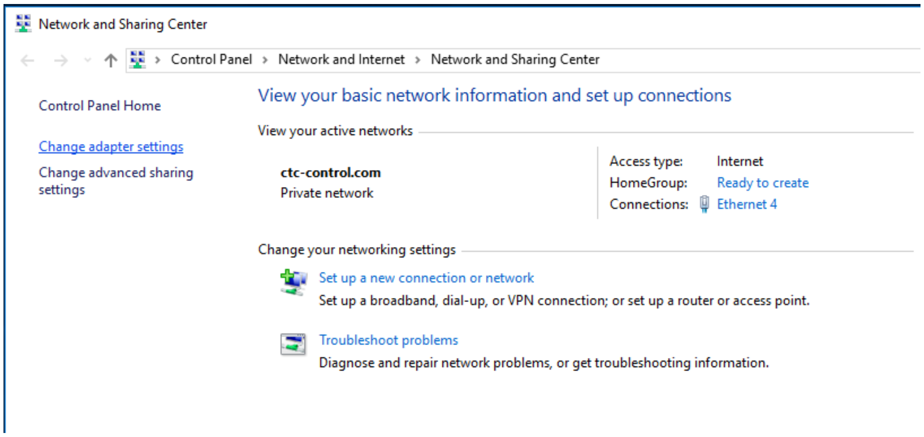
Kernel	
Start automatically	No
Processor core	3
Kernel memory (in MB)	256
Kernel memory mode	Extended Virtual and Physical Memory
Kernel Clock Rate (in us)	100
OS extensions	0
Round Robin Time Slice (in ms)	50
Round Robin Priority Threshold	140
Spin Control Threshold in ms	3000
Debugger COM Port	None
Debugger Baud Rate	9600

Processor core
The processor core used by the kernel

Close the Node Management screen and exit out of the INtime Configuration Panel. You will be prompted to reboot for the changes to take effect. Prior to rebooting we want to remove any bridged network adapters. Invoke 'Open Network and Sharing Center' by right clicking on the network tray icon or using Control Panel:



Select ‘Change adapter settings’:



Network and Sharing Center

Control Panel > Network and Internet > Network and Sharing Center

Control Panel Home

Change adapter settings

Change advanced sharing settings

View your basic network information and set up connections

View your active networks

ctc-control.com
Private network

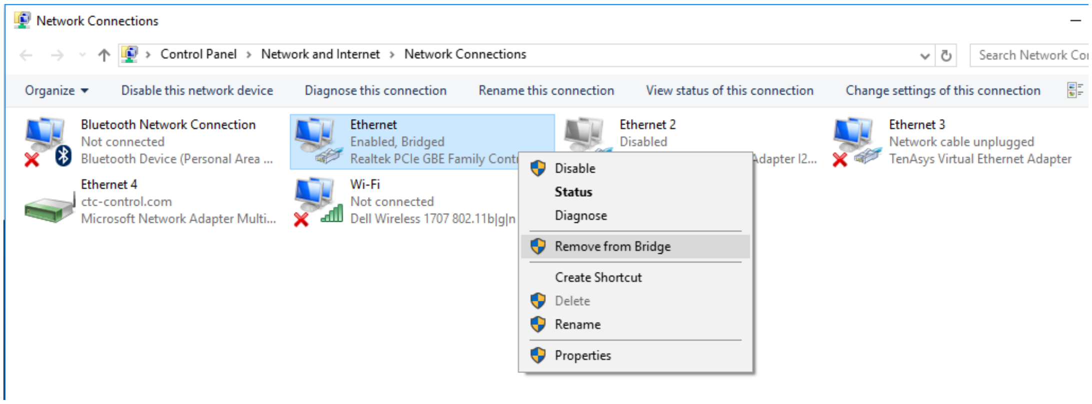
Access type: Internet
HomeGroup: Ready to create
Connections: Ethernet 4

Change your networking settings

Set up a new connection or network
Set up a broadband, dial-up, or VPN connection; or set up a router or access point.

Troubleshoot problems
Diagnose and repair network problems, or get troubleshooting information.

Right click on the bridged network adapters and select ‘Remove from bridge’. This is usually your main network adapter and the TenAsys Virtual Ethernet Adapter:



Network Connections

Control Panel > Network and Internet > Network Connections

Search Network Co

Organize
Disable this network device
Diagnose this connection
Rename this connection
View status of this connection
Change settings of this connection

Bluetooth Network Connection
Not connected
Bluetooth Device (Personal Area ...

Ethernet
Enabled, Bridged
Realtek PCIe GBE Family Cont

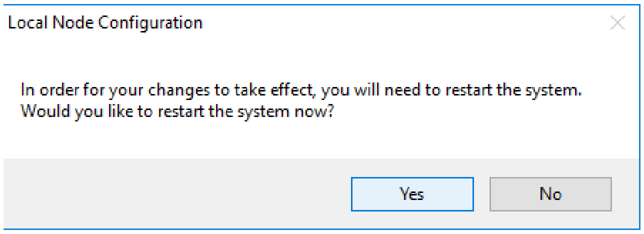
Ethernet 2
Disabled
Adapter I2...

Ethernet 3
Network cable unplugged
TenAsys Virtual Ethernet Adapter

Ethernet 4
ctc-control.com
Microsoft Network Adapter Multi...

Wi-Fi
Not connected
Dell Wireless 1707 802.11b|g|n

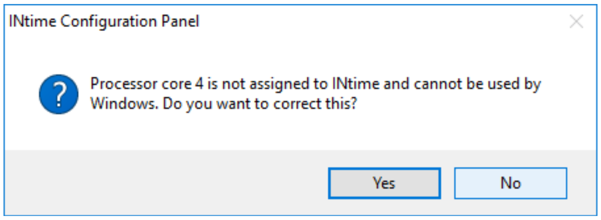
You may now reboot your PC by selecting ‘Yes’ on the Local Node Configuration prompt you received when you exited the Configurator:



Local Node Configuration

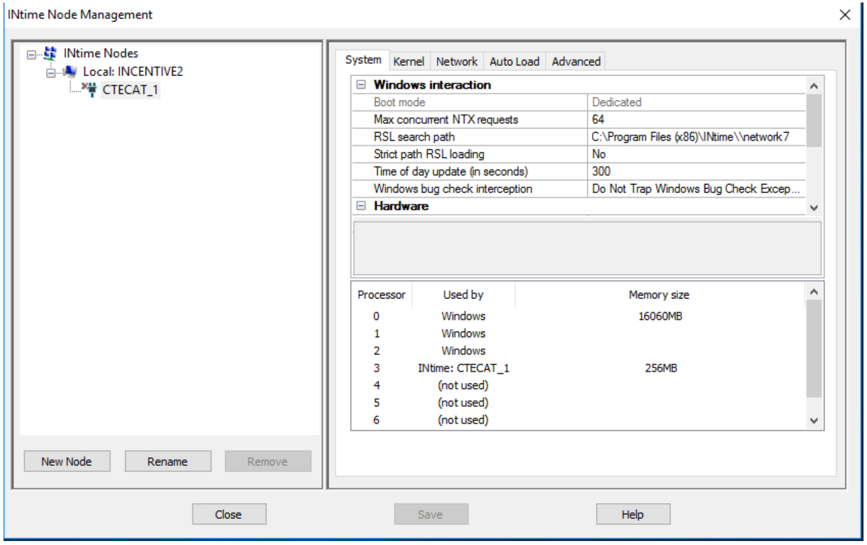
In order for your changes to take effect, you will need to restart the system.
Would you like to restart the system now?

After rebooting open the INtime Configuration Panel again and you may be prompted that 'Processor core 4 is not assigned to INtime…', select 'No':



Processor core 4 is not assigned to INtime and cannot be used by Windows. Do you want to correct this?

Return to the INtime Node Management screen again and select 'System' on the tabs. Note that Windows now has 3 cores and core 3 is assigned to CTECAT_1:



Windows interaction	
Boot mode	Dedicated
Max concurrent NTX requests	64
RSL search path	C:\Program Files (x86)\INtime\\network7
Strict path RSL loading	No
Time of day update (in seconds)	300
Windows bug check interception	Do Not Trap Windows Bug Check Excep...
Hardware	

Processor	Used by	Memory size
0	Windows	16060MB
1	Windows	
2	Windows	
3	INtime: CTECAT_1	256MB
4	(not used)	
5	(not used)	
6	(not used)	

To execute in standalone mode make sure a file called EtherCAT_Master_Only.txt exists in the C:\5300PC directory. Contents of the file do not matter, it is just checked for its existence. You may now run your application under full control of the CTC_Incentive API, without QuickBuilder or its CTPLC_1 process.

QuickBuilder Programming and Atomicity

This is just a quick recommendation of something to think about as you structure your QuickBuilder program and your system architecture. Access to variables, whether normal registers or variants, is atomic and protected during read and write access. Also when writing a QuickBuilder program things within a conditional or loop { } are protected from having other QuickBuilder threads from accessing the same variables. Where subtle problems can occur is in the simple instance of writing something like this with inline code:

```
i = i+1;
```

Now that may seem straight forward and is protected between QuickBuilder threads but if you have something on the network or via the API poking 'i' with a value, say a 0, the result may not always be what you think it might. Consider the situation where QuickBuilder reads the value of 'i' (say containing 2), it then adds a 1 to it but before it can be stored back to 'i' the network pokes a 0 in 'i'. Well for a brief period of time 'i' will be 0 but then the result of 2 + 1, or 3 will overwrite the 0. Atomicity or protection will not last for the full arithmetic formula unless it is between QuickBuilder tasks. A way around this is to have the network or API write public variables that are not involved in QuickBuilder calculations and have QuickBuilder access those variables at the beginning of their calculations, copying them to internal variables.

Some Common Issues and Resolutions

Shutdown and Restart Leave Power Switch Lit

Some computers have a problem with the INtime environment with Shutdown and Restart where Restart actually attempts a Shutdown and Shutdown does close Windows but the power switch stays lit. Only way out of this is to hold the power switch for 5 seconds to completely shutdown. This was recently observed on a Dell Optiplex system. An easy resolution to this is a registry change:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\rtif\Parameters

Change Flags from 0x05 to 0x25.

Bridged Network Fails after Windows Shutdown

This problem is a bug in some Windows 10 network drivers. We have noticed doing a Restart of Windows works fine but some computers after Shutdown you cannot communicate to our Incentive environment via UDP or TCP where we share a network controller with Windows (TenAsys virtual adapter). When Windows does a Shutdown it actually saving a memory image and not doing a full reboot. In doing so it is not properly saving the bridged network information. The resolution is simple, have Windows do a full shutdown (not a fast boot which is default) so a fresh memory image is loaded. The solution to this issue is detailed in the "*Startup and Network Configuration*" section.

Appendix

A

[A] ABB

ABB manufactures a number of drives. The drive currently supported is the single axis Microflex e150 servo drive. This section provides information that may be specific to this manufacturer.

eCAT_driveType – 12

DC Sync

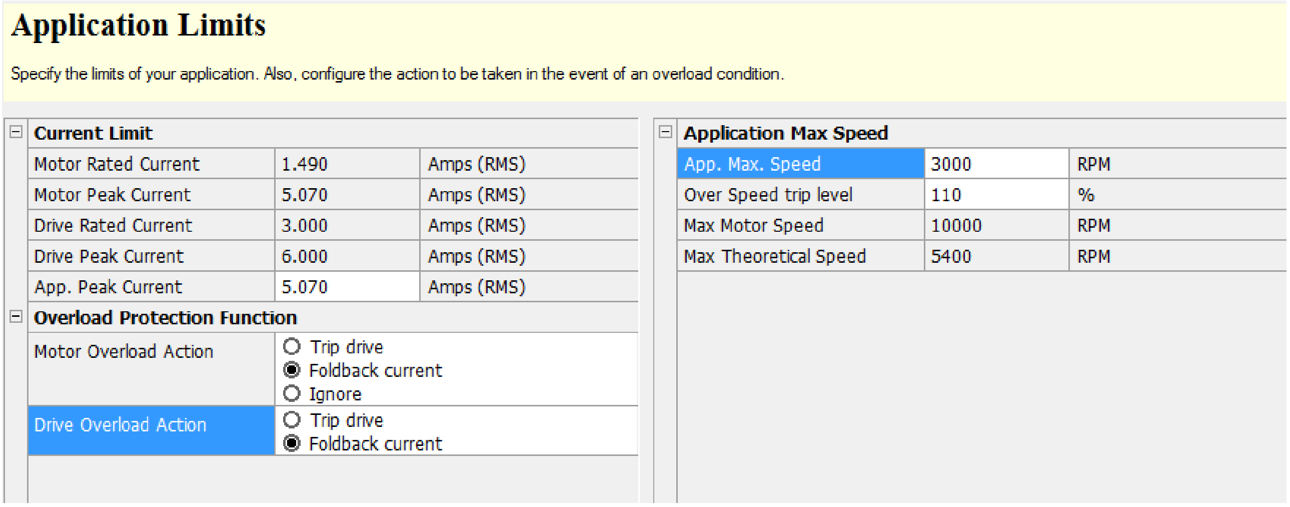
ABB drives must have DC Sync0 enabled prior to enabling the drive as well as a 32 or 64-bit slave reference as the first node on the EtherCAT network. Reference the 'dcsync' command example within the main body of this manual.

MSB code sample prior to drive enable:

```
// This is only needed for ABB & Emerson/Control Techniques
delay 2000 ms;  // needed in case restart so syncs when cycle DC Sync on/off.
// Cycle time is 1 ms, start it 100 ms in the future.
// Note that we need to make sure that the first slave device in the EtherCAT
// Network supports 32/64-bit distributed clocks for this to work properly.
// Thus far that is Beckhoff, Wago, ABB, and Sanyo Denki.
dcsync -1, 1000000, 0, 0, 100000000;
delay 200 ms;     // starts 100 milliseconds into the future

/**************** ENABLE DRIVE *******************/
[Drive_Enable]
// Issue any hardware enable output commands
//
// Power up the drive amplifier
drive enable;
```

Drive Configuration

When configuring the drive and application using the ABB Mint Workbench a few parameters should be modified from the factory defaults. The first is the Application Maximum Speed and the next is the Motor & Drive Overload Action being set to 'Foldback current'. Failure to modify the overload action can result in a 'Following Error' as the drive attempts to increase torque to attain the commanded position from the EtherCAT Master.



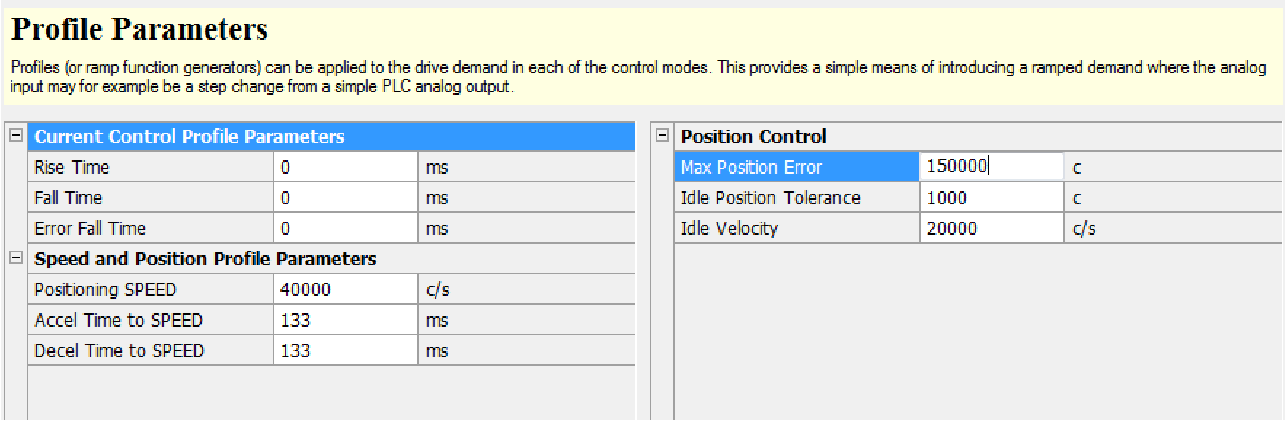
Application Limits

Specify the limits of your application. Also, configure the action to be taken in the event of an overload condition.

Current Limit		
Motor Rated Current	1.490	Amps (RMS)
Motor Peak Current	5.070	Amps (RMS)
Drive Rated Current	3.000	Amps (RMS)
Drive Peak Current	6.000	Amps (RMS)
App. Peak Current	5.070	Amps (RMS)
Overload Protection Function		
Motor Overload Action	○ Trip drive ● Foldback current ○ Ignore	
Drive Overload Action	○ Trip drive ● Foldback current	

Application Max Speed		
App. Max. Speed	3000	RPM
Over Speed trip level	110	%
Max Motor Speed	10000	RPM
Max Theoretical Speed	5400	RPM

Another parameter that must be changed is the acceptable position error. This is dependent upon the 'ppr' of the drive. The default is typically around 1K counts but with a 512K/rev ppr this is a very small value and can happen quite easily. Below shows a setting of 150K pulses. This is how much the commanded EtherCAT Master position can lag before the drive issues a 'Following Error'. A large lag is acceptable since the drive will catch up the commanded position prior to stopping.



Profile Parameters

Profiles (or ramp function generators) can be applied to the drive demand in each of the control modes. This provides a simple means of introducing a ramped demand where the analog input may for example be a step change from a simple PLC analog output.

Current Control Profile Parameters		
Rise Time	0	ms
Fall Time	0	ms
Error Fall Time	0	ms
Speed and Position Profile Parameters		
Positioning SPEED	40000	c/s
Accel Time to SPEED	133	ms
Decel Time to SPEED	133	ms

Position Control		
Max Position Error	150000	c
Idle Position Tolerance	1000	c
Idle Velocity	20000	c/s

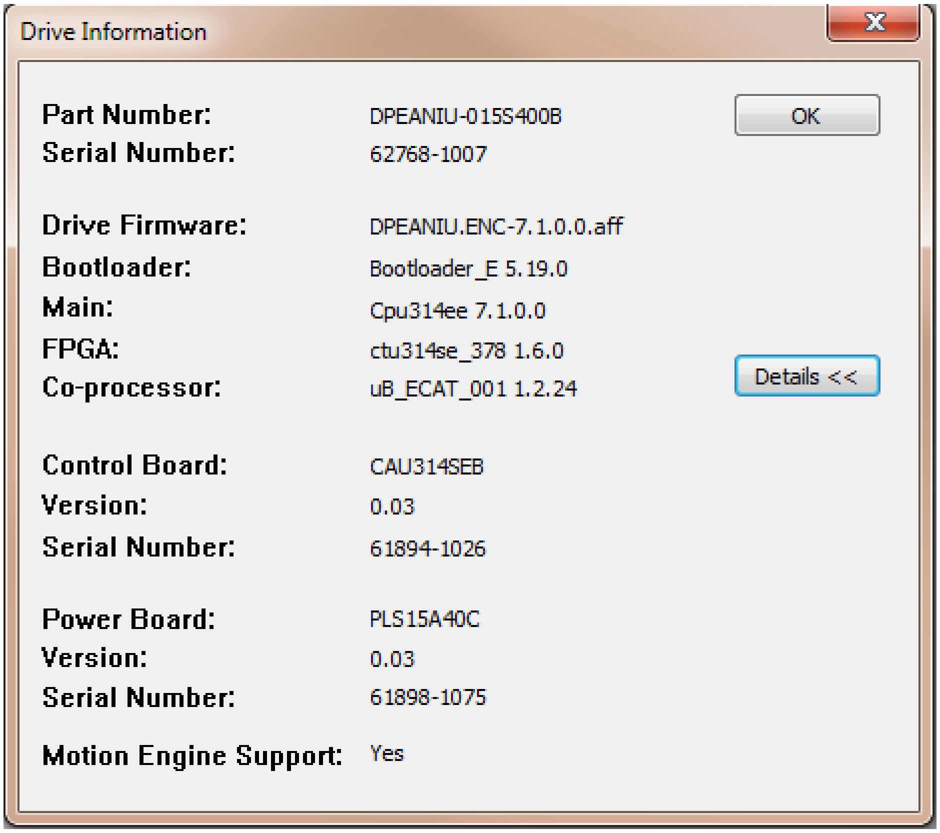
Appendix B

[B] Advanced Motion Controls

Advanced Motion Controls (AMC) manufactures a number of drives. The drive currently supported is the single axis DigiFlex servo drive (DPE series). This section provides information that may be specific to this manufacturer.

eCAT_driveType – 8

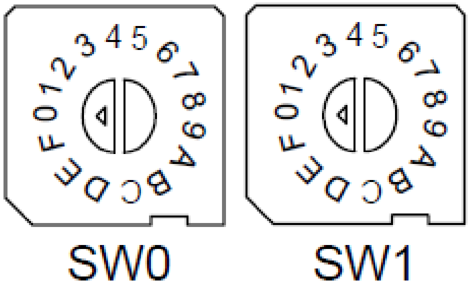
Drive Information & Firmware

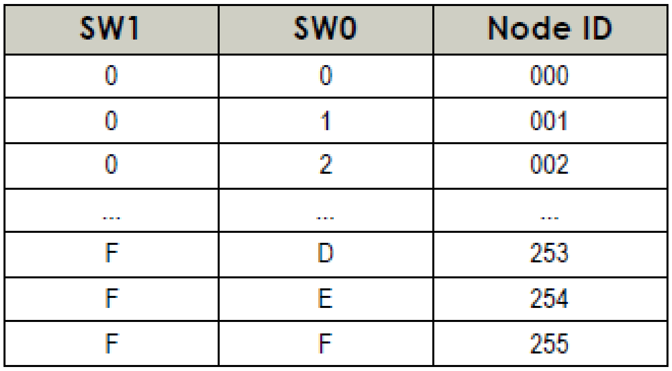


Part Number:	DPEANIU-015S400B
Serial Number:	62768-1007
Drive Firmware:	DPEANIU.ENC-7.1.0.0.aff
Bootloader:	Bootloader_E 5.19.0
Main:	Cpu314ee 7.1.0.0
FPGA:	ctu314se_378 1.6.0
Co-processor:	uB_ECAT_001 1.2.24
Control Board:	CAU314SEB
Version:	0.03
Serial Number:	61894-1026
Power Board:	PLS15A40C
Version:	0.03
Serial Number:	61898-1075
Motion Engine Support:	Yes

Station Alias

In an EtherCAT network, slaves are automatically assigned addresses based on their position in the bus. When a device, such as a drive, must have a fixed assigned identification that is independent of cabling, a Station Alias is needed. Advanced Motion provides two 16-position rotary switches with hexadecimal encoding for this purpose. This allows for a setting of 0 to 255 (FFh), where 0 defaults to the automatic address assignment. As an example, if SW0 is set to a 1 and SW1 to an A this would be 1Ah or 1 X 16 + 10 = 26. Since the M3-41 only supports up to 16 drives SW0 would always be set to 0 and only SW1 used. Setting both switches to 0 defaults to automatic addressing.





SW1	SW0	Node ID
0	0	000
0	1	001
0	2	002
...	...	...
F	D	253
F	E	254
F	F	255

DC Sync

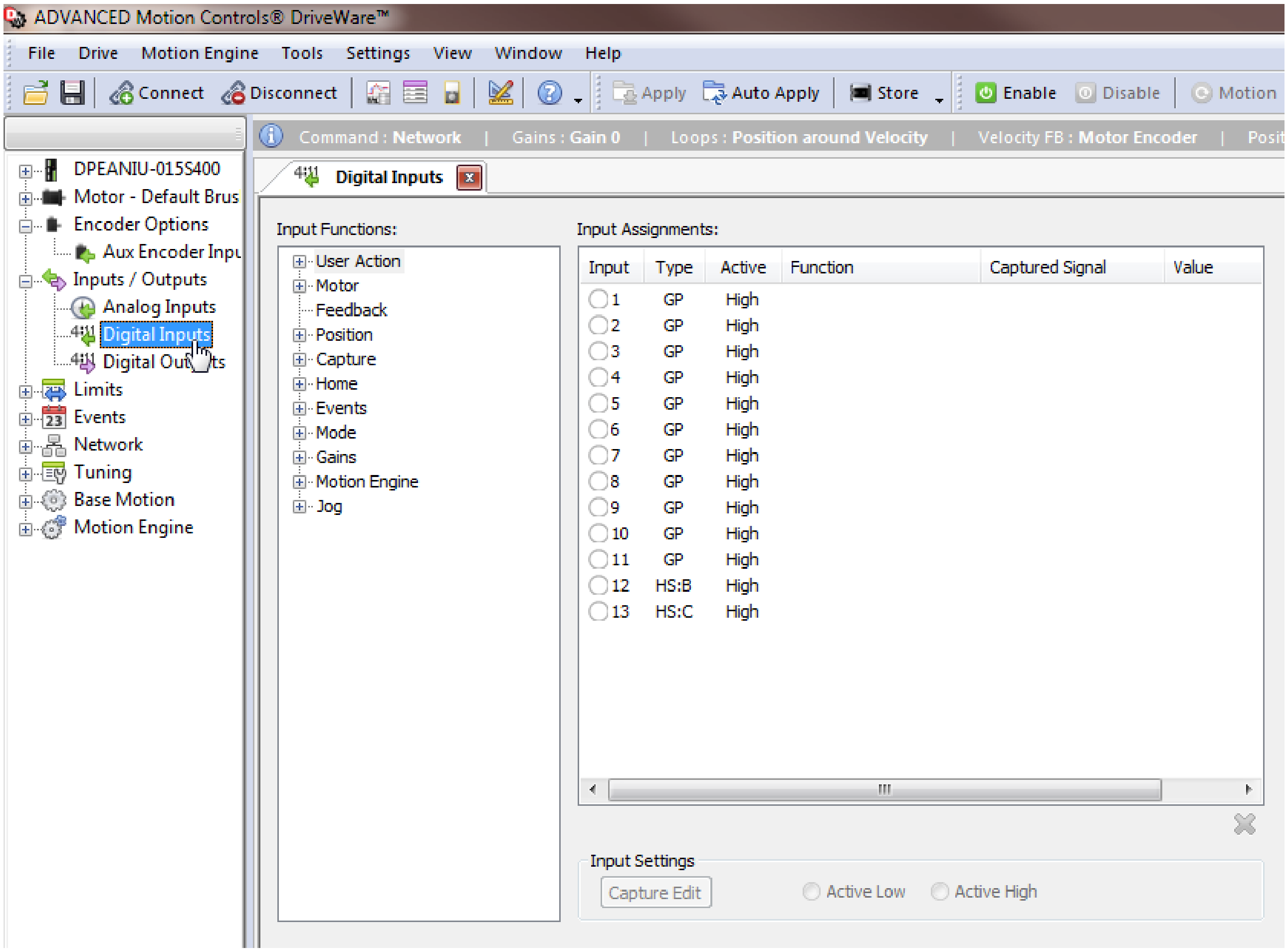
Advanced Motion Controls drives do not currently support a network-based DC Sync, only slave based. Therefore this option is not available.

Inputs/Outputs

The inputs and outputs of the AMC drive are highly configurable and it is left to the user as to how they should be set up and used. The MSB ‘din’ variable maps to object 0x2023.1 (Digital Input Value) and ‘dout’ maps to object 0x2001.3 (User Bit Control). These objects are part of the PDO update cycle and refreshed every millisecond.

DriveWare can be used to configure the functionality of the inputs and outputs. (Make sure you apply the changes to the drive and save the modified configuration to non-volatile memory.)

Inputs:



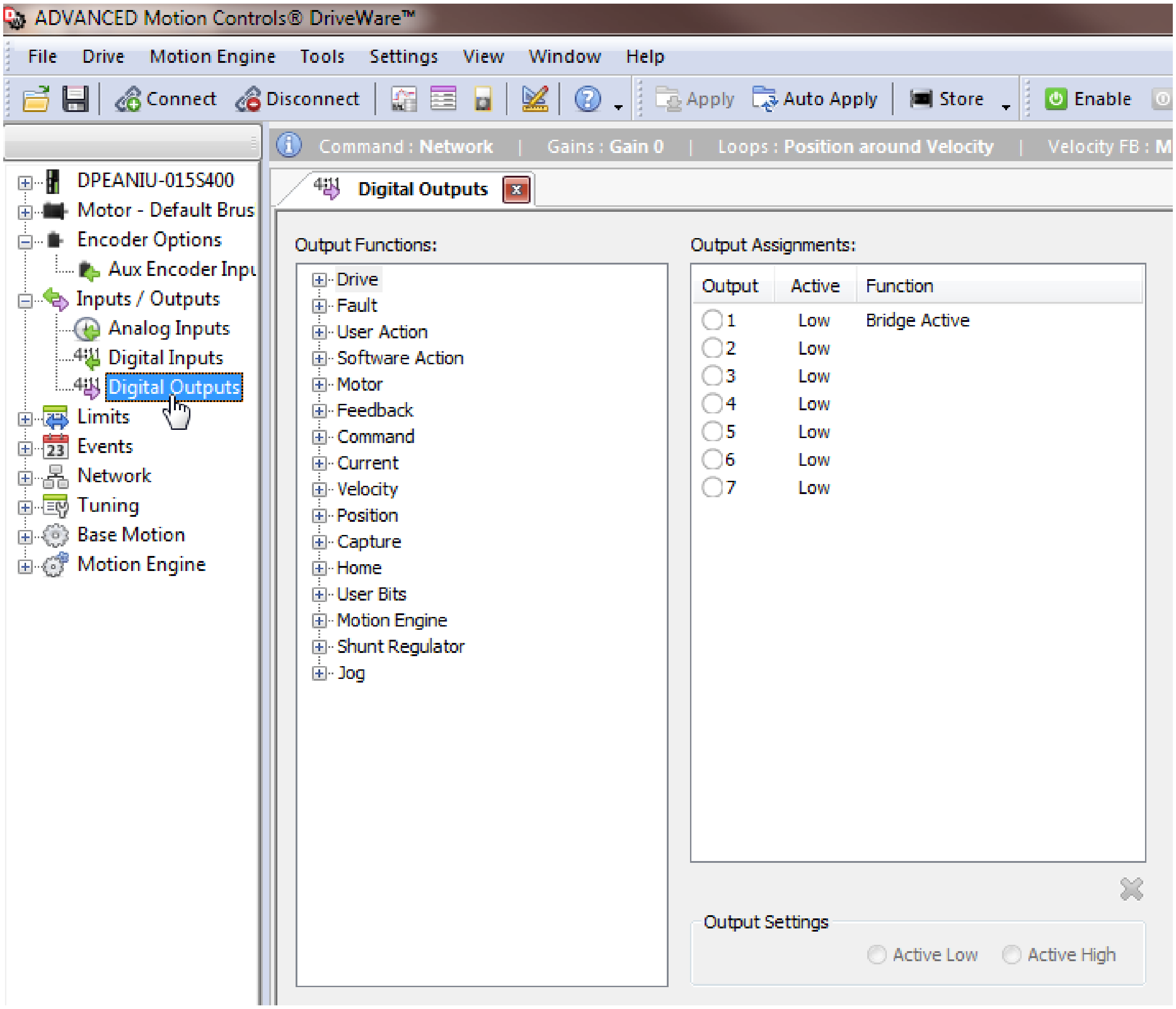
Command : Network | Gains : Gain 0 | Loops : Position around Velocity | Velocity FB : Motor Encoder | Posit

Input Assignments:

Input	Type	Active	Function	Captured Signal	Value
○ 1	GP	High			
○ 2	GP	High			
○ 3	GP	High			
○ 4	GP	High			
○ 5	GP	High			
○ 6	GP	High			
○ 7	GP	High			
○ 8	GP	High			
○ 9	GP	High			
○ 10	GP	High			
○ 11	GP	High			
○ 12	HS:B	High			
○ 13	HS:C	High			

○ Active Low ○ Active High

Outputs:

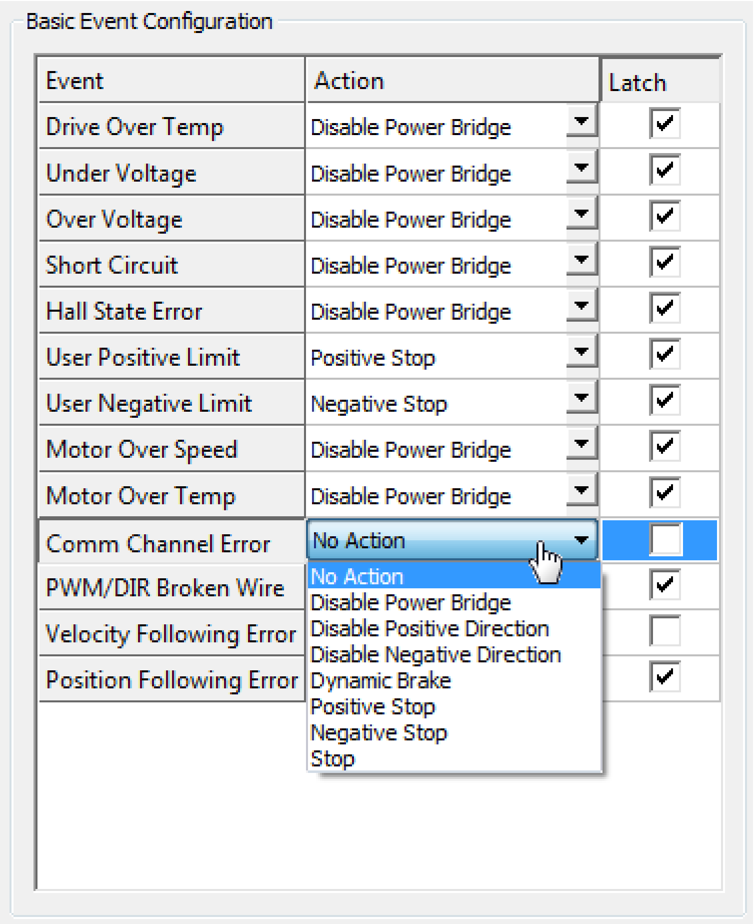


Command : Network | Gains : Gain 0 | Loops : Position around Velocity | Velocity FB : M

Output	Active	Function
○ 1	Low	Bridge Active
○ 2	Low	
○ 3	Low	
○ 4	Low	
○ 5	Low	
○ 6	Low	
○ 7	Low	

Communications Error

Typically when a communication error occurs it is ignored by the AMC drive. When running EtherCAT this is not acceptable — it means the M3-41 has lost control of the drive. DriveWare allows a very detailed configuration of what should happen when an Error Event occurs. If the “Comm Channel Error” check box is not selected, the M3-41 module automatically enables it at the drive level with the “Disable Power Bridge” setting. If the user selects some other setting within DriveWare (other than No Action) this setting will override that of the M3-41 module and become the default setting.

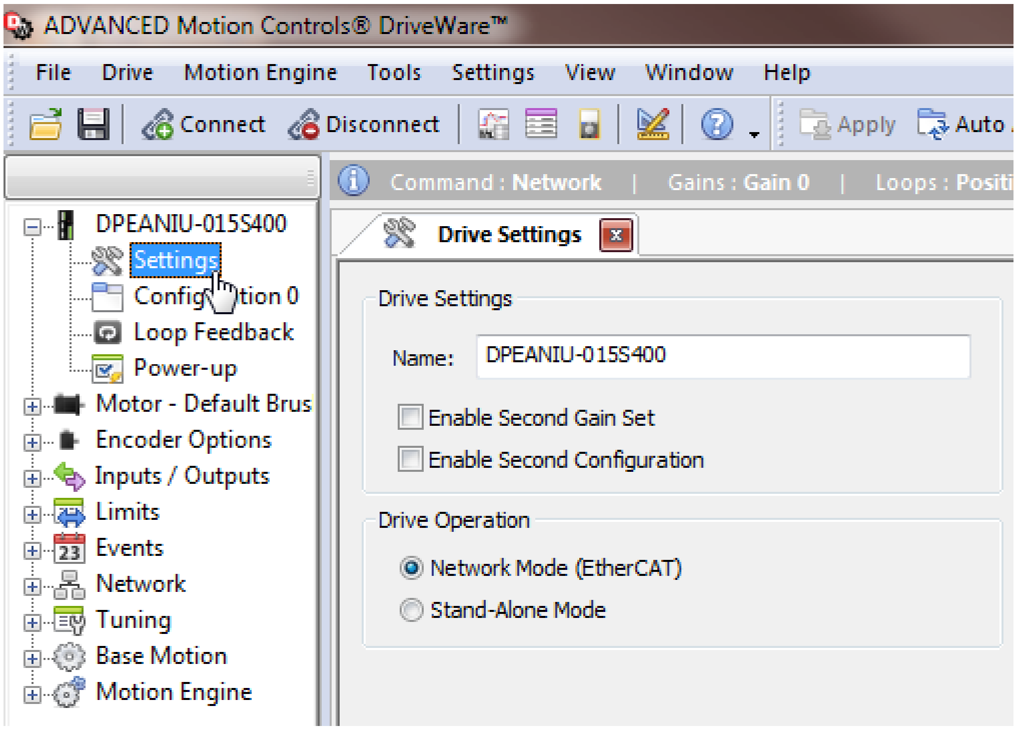


Basic Event Configuration

Event	Action	Latch
Drive Over Temp	Disable Power Bridge	☑
Under Voltage	Disable Power Bridge	☑
Over Voltage	Disable Power Bridge	☑
Short Circuit	Disable Power Bridge	☑
Hall State Error	Disable Power Bridge	☑
User Positive Limit	Positive Stop	☑
User Negative Limit	Negative Stop	☑
Motor Over Speed	Disable Power Bridge	☑
Motor Over Temp	Disable Power Bridge	☑
Comm Channel Error	No Action	□
PWM/DIR Broken Wire		☑
Velocity Following Error		□
Position Following Error		☑

DriveWare

CTC used DriveWare 7.1 on a Windows 7 64-bit computer to test and configure the drives. Typically DriveWare will be needed for tuning and limited setup. Below are some setup screens and their typical parameters. Note that the PDO setup will be overridden by the M3-41 module and may be left at the default.



Drive Settings

Name: DPEANIU-015S400

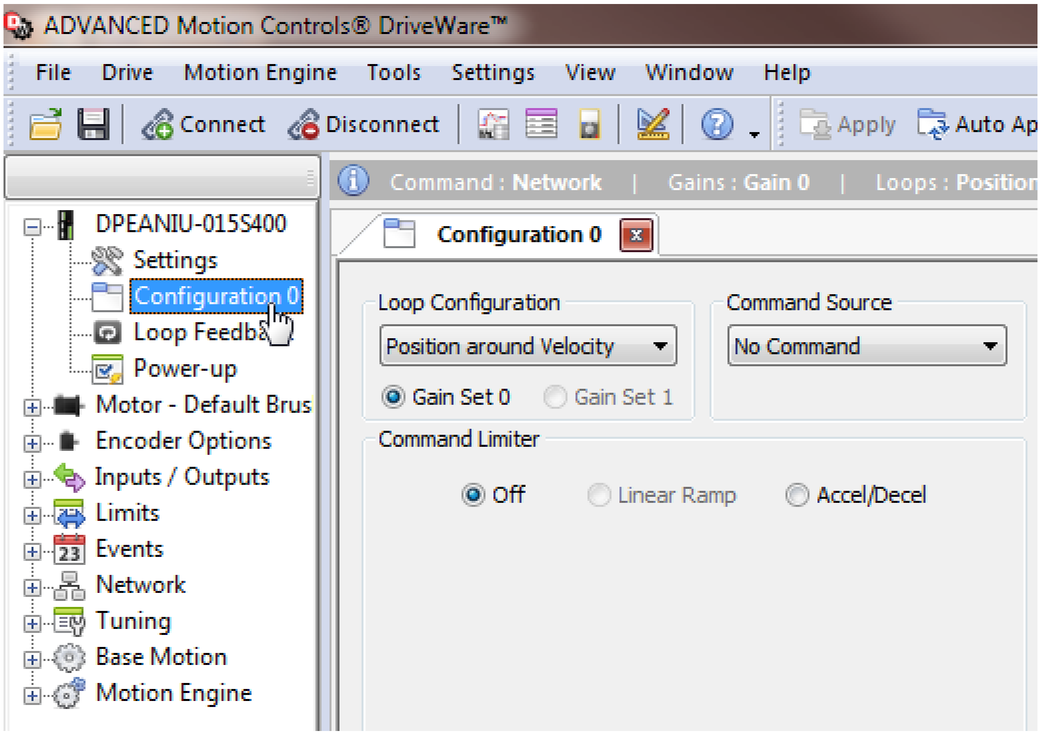
□ Enable Second Gain Set

□ Enable Second Configuration

Drive Operation

● Network Mode (EtherCAT)

○ Stand-Alone Mode



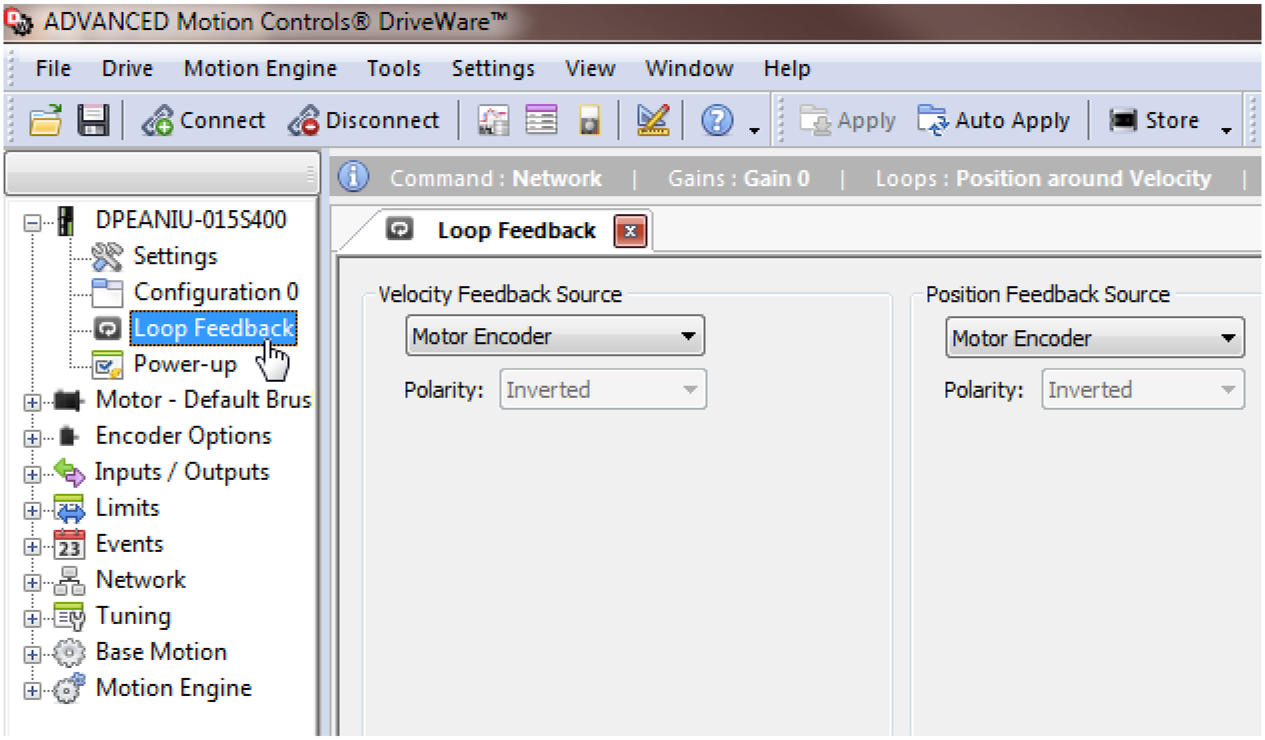
Loop Configuration: Position around Velocity

● Gain Set 0 ○ Gain Set 1

Command Source: No Command

Command Limiter

● Off ○ Linear Ramp ○ Accel/Decel

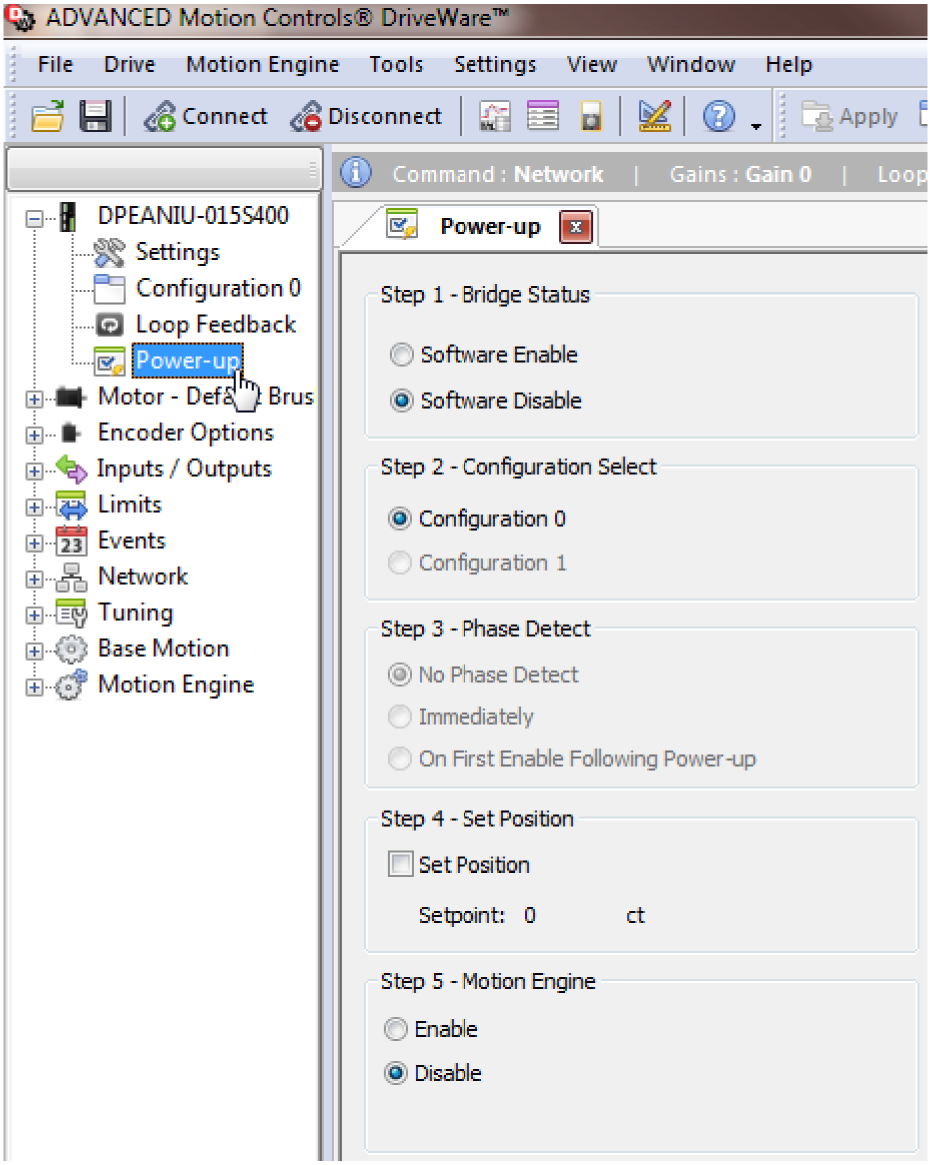


Velocity Feedback Source: Motor Encoder

Polarity: Inverted

Position Feedback Source: Motor Encoder

Polarity: Inverted



Step 1 - Bridge Status

○ Software Enable

● Software Disable

Step 2 - Configuration Select

● Configuration 0

○ Configuration 1

Step 3 - Phase Detect

● No Phase Detect

○ Immediately

○ On First Enable Following Power-up

Step 4 - Set Position

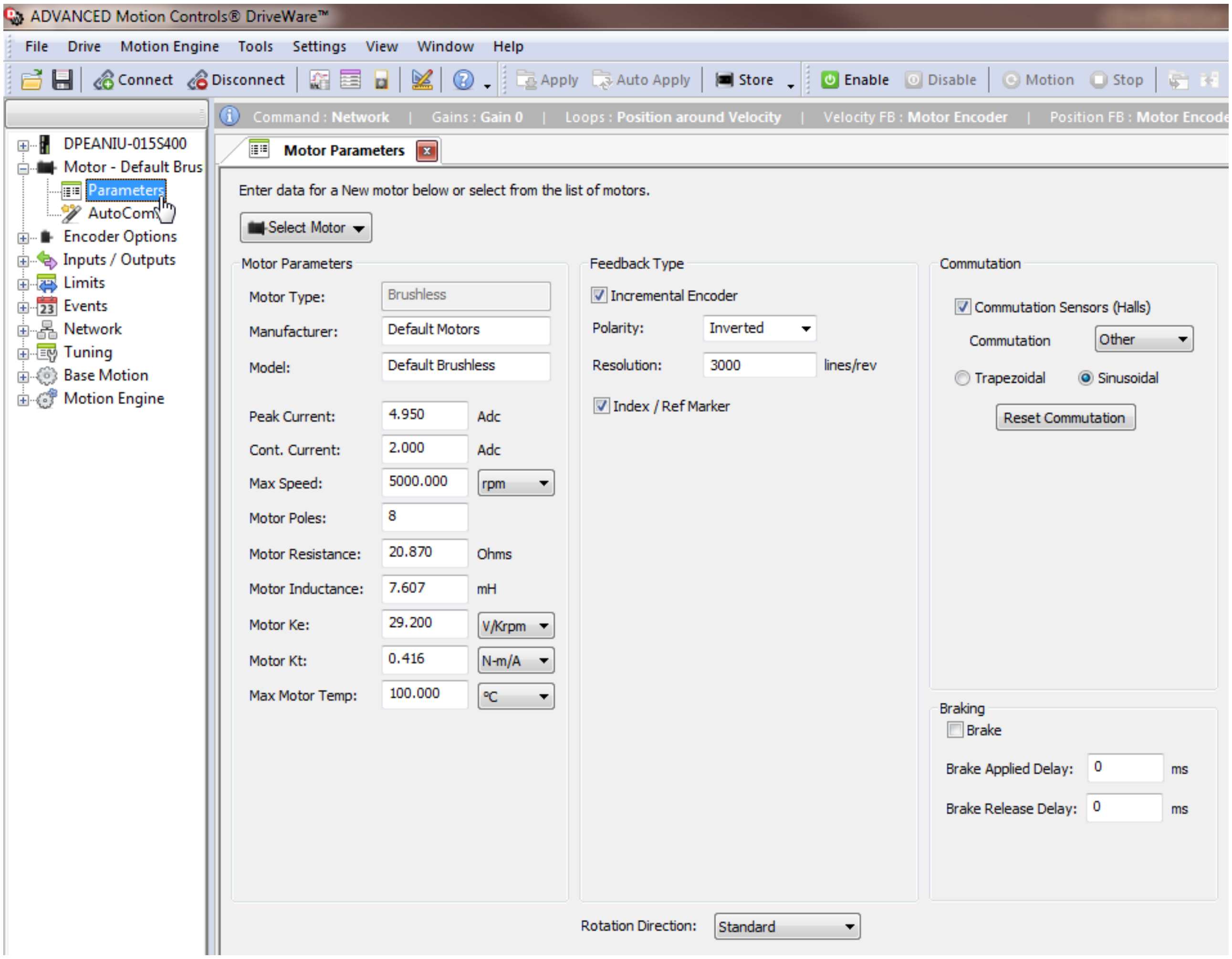
□ Set Position

Setpoint: 0 ct

Step 5 - Motion Engine

○ Enable

● Disable



Command : Network | Gains : Gain 0 | Loops : Position around Velocity | Velocity FB : Motor Encoder | Position FB : Motor Encode

Enter data for a New motor below or select from the list of motors.

Motor Parameters

Motor Type: Brushless
Manufacturer: Default Motors
Model: Default Brushless
Peak Current: 4.950 Adc
Cont. Current: 2.000 Adc
Max Speed: 5000.000 rpm
Motor Poles: 8
Motor Resistance: 20.870 Ohms
Motor Inductance: 7.607 mH
Motor Ke: 29.200 V/Krpm
Motor Kt: 0.416 N-m/A
Max Motor Temp: 100.000 ºC

Feedback Type

☑ Incremental Encoder
Polarity: Inverted
Resolution: 3000 lines/rev
☑ Index / Ref Marker

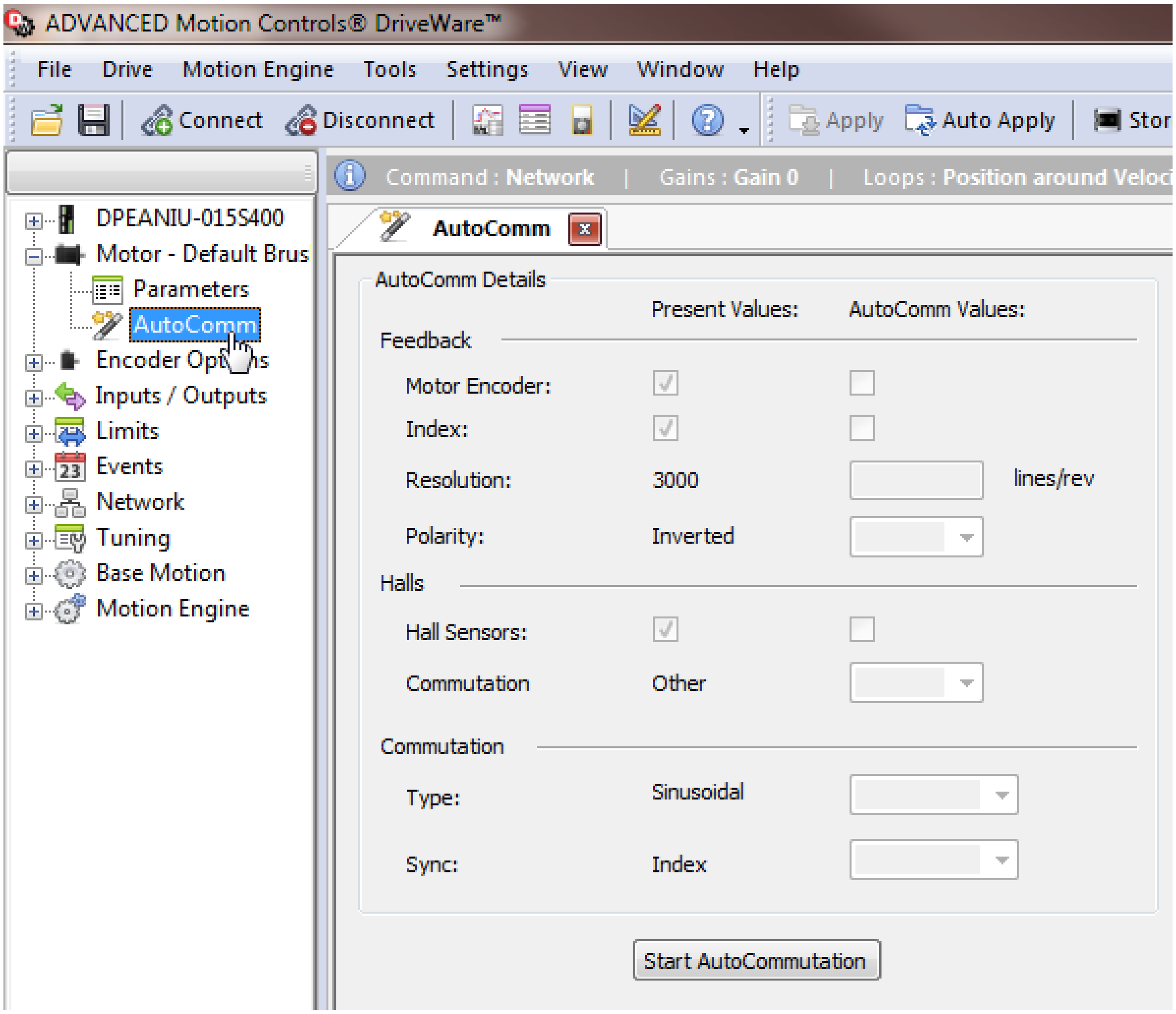
Commutation

☑ Commutation Sensors (Halls)
Commutation: Other
○ Trapezoidal ● Sinusoidal

Braking

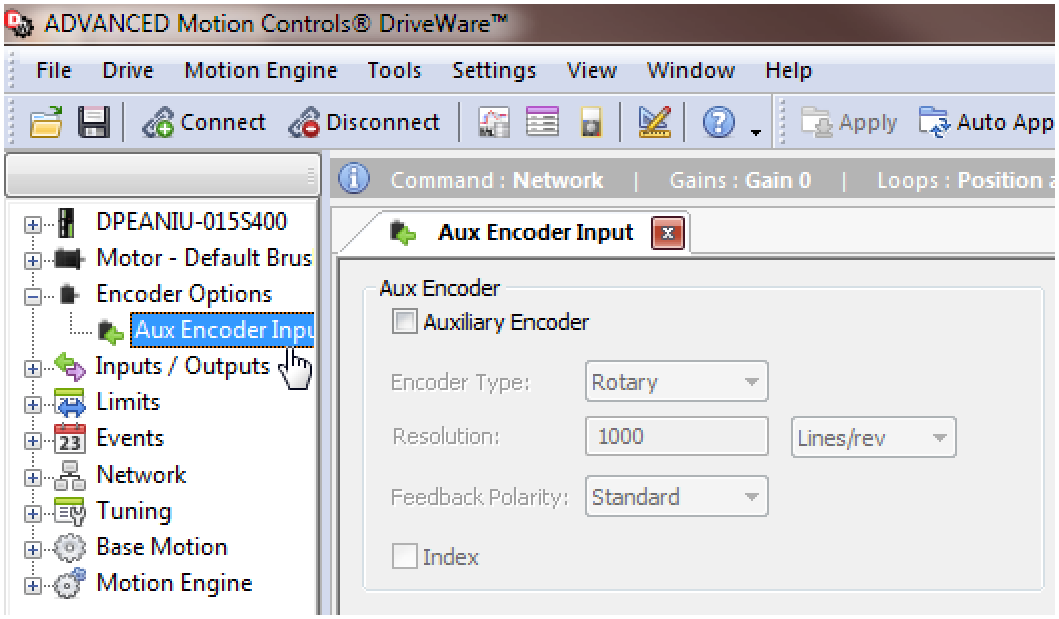
☐ Brake
Brake Applied Delay: 0 ms
Brake Release Delay: 0 ms

Rotation Direction: Standard



AutoComm Details

	Present Values:	AutoComm Values:	
Feedback			
Motor Encoder:	☑	☐	
Index:	☑	☐	
Resolution:	3000		lines/rev
Polarity:	Inverted		
Halls			
Hall Sensors:	☑	☐	
Commutation	Other		
Commutation			
Type:	Sinusoidal		
Sync:	Index		



Aux Encoder

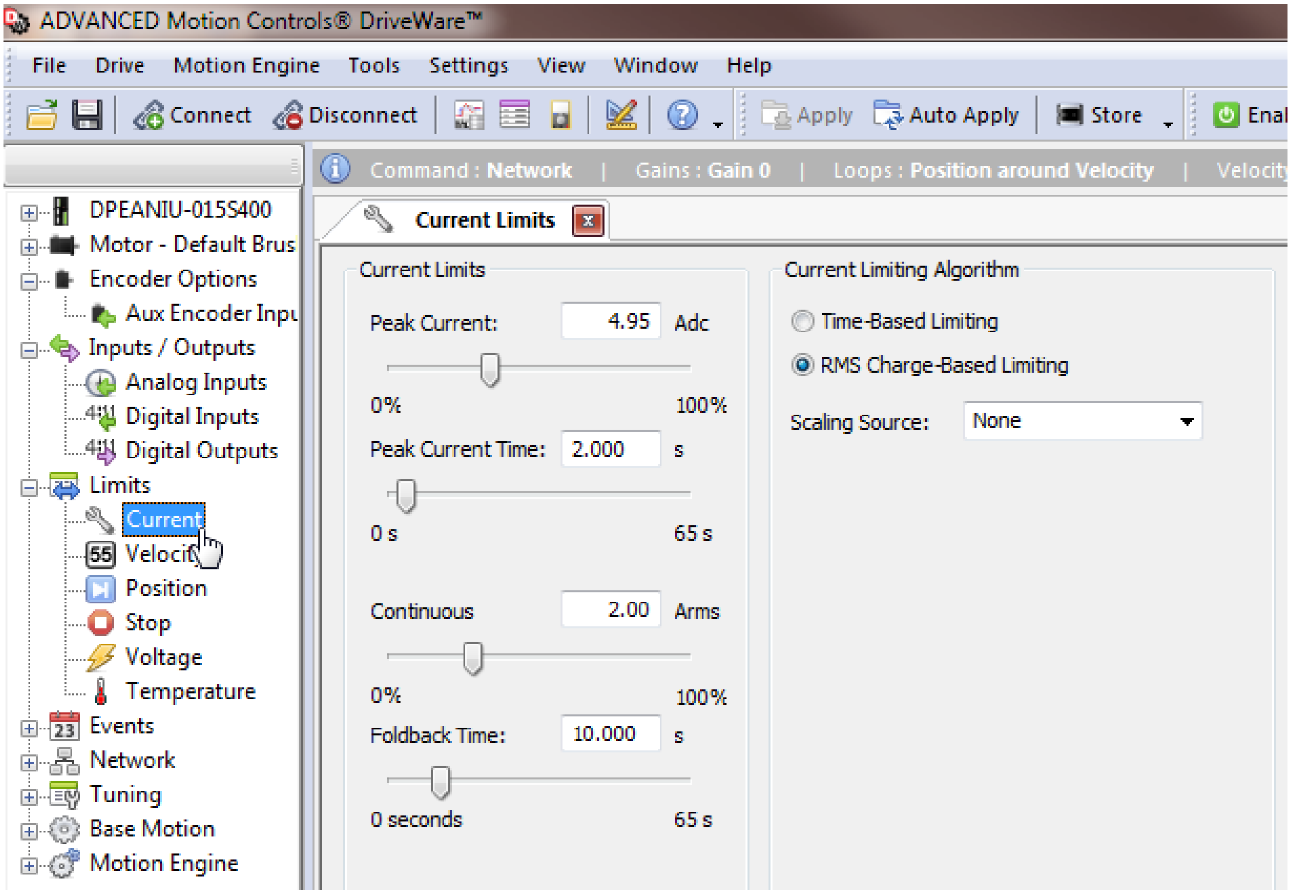
☐ Auxiliary Encoder

Encoder Type: Rotary

Resolution: 1000 Lines/rev

Feedback Polarity: Standard

☐ Index



Command : Network | Gains : Gain 0 | Loops : Position around Velocity | Velocity

Current Limits

Peak Current: 4.95 Adc

Peak Current Time: 2.000 s

Continuous 2.00 Arms

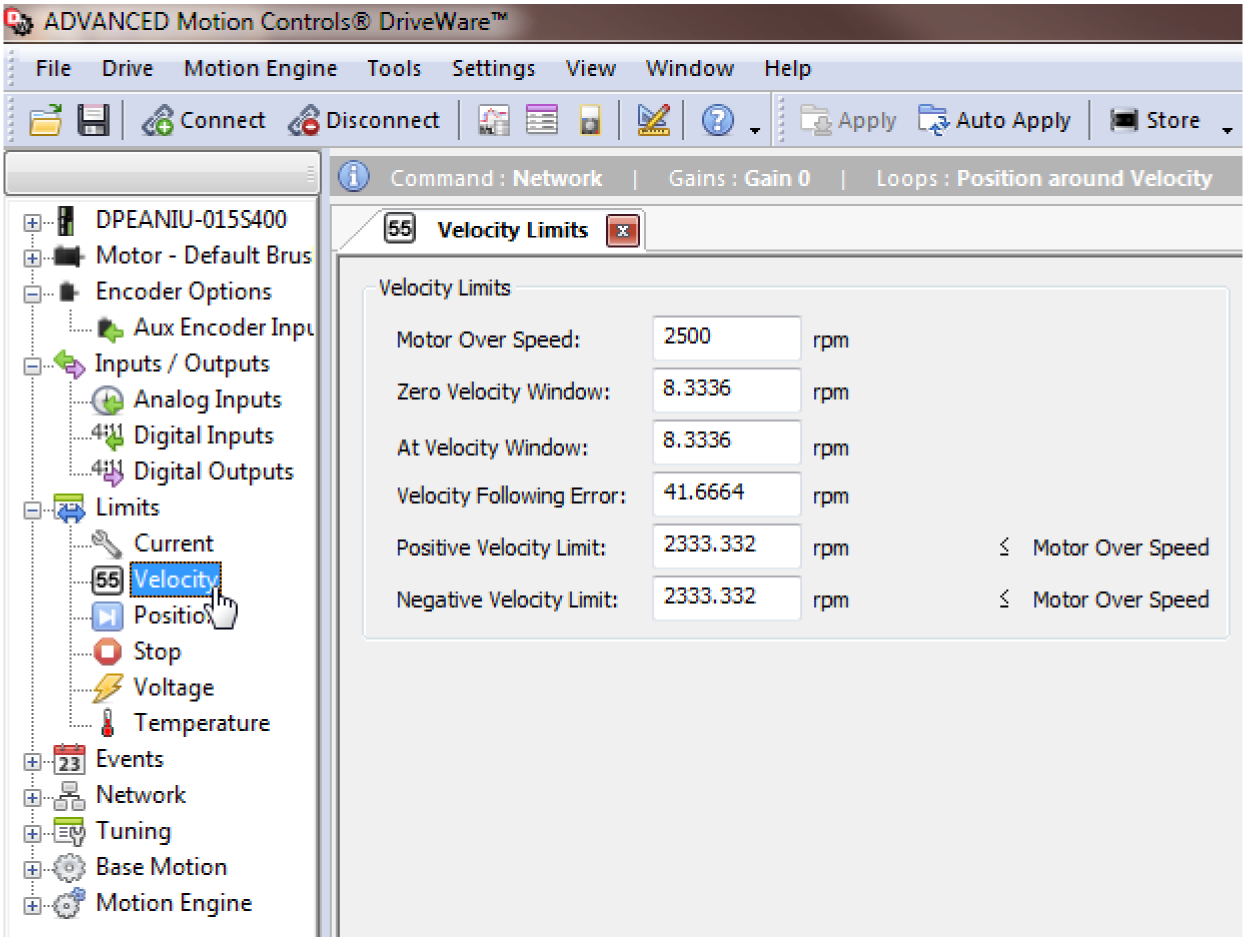
Foldback Time: 10.000 s

Current Limiting Algorithm

○ Time-Based Limiting

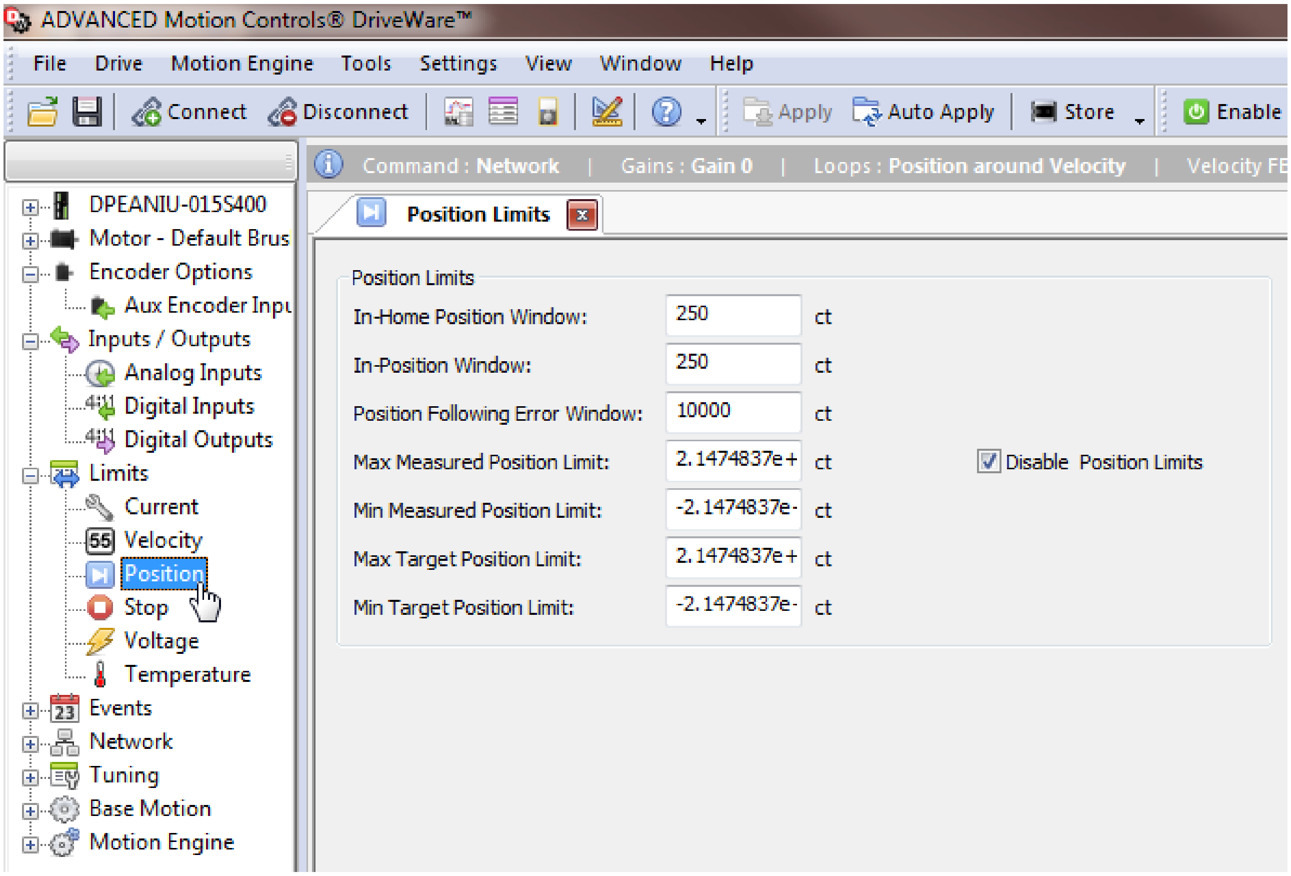
● RMS Charge-Based Limiting

Scaling Source: None



Velocity Limits

Motor Over Speed:	2500	rpm	
Zero Velocity Window:	8.3336	rpm	
At Velocity Window:	8.3336	rpm	
Velocity Following Error:	41.6664	rpm	
Positive Velocity Limit:	2333.332	rpm	≤ Motor Over Speed
Negative Velocity Limit:	2333.332	rpm	≤ Motor Over Speed


ADVANCED Motion Controls® DriveWare™
File Drive Motion Engine Tools Settings View Window Help
Connect Disconnect Apply Auto Apply Store Enable
Command : Network | Gains : Gain 0 | Loops : Position around Velocity | Velocity FE
DPEANIU-015S400
Motor - Default Brus
Encoder Options
Aux Encoder Inpu
Inputs / Outputs
Analog Inputs
Digital Inputs
Digital Outputs
Limits
Current
Velocity
Position
Stop
Voltage
Temperature
Events
Network
Tuning
Base Motion
Motion Engine

Position Limits

Position Limits

In-Home Position Window: 250 ct

In-Position Window: 250 ct

Position Following Error Window: 10000 ct

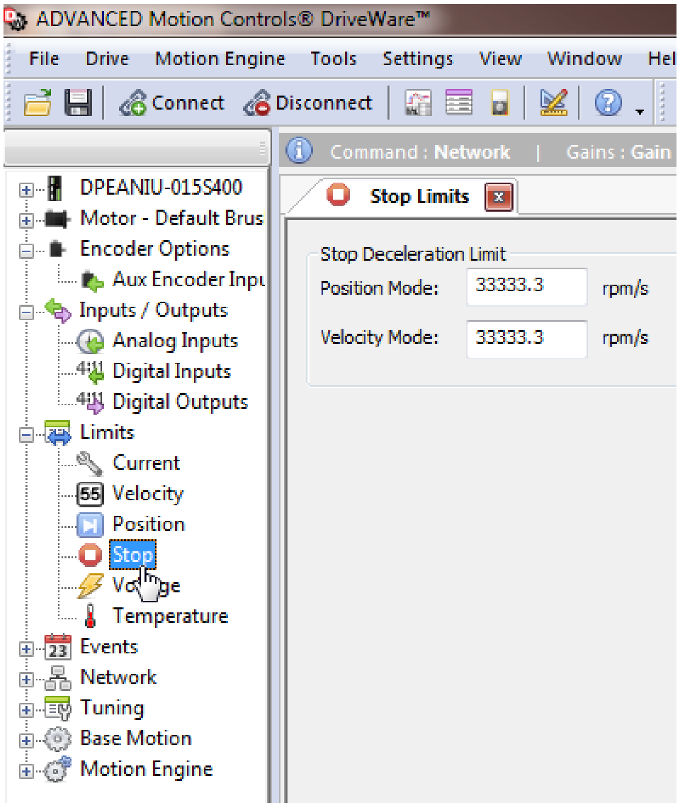
Max Measured Position Limit: 2.1474837e+ ct

Min Measured Position Limit: -2.1474837e- ct

Max Target Position Limit: 2.1474837e+ ct

Min Target Position Limit: -2.1474837e- ct

☑ Disable Position Limits

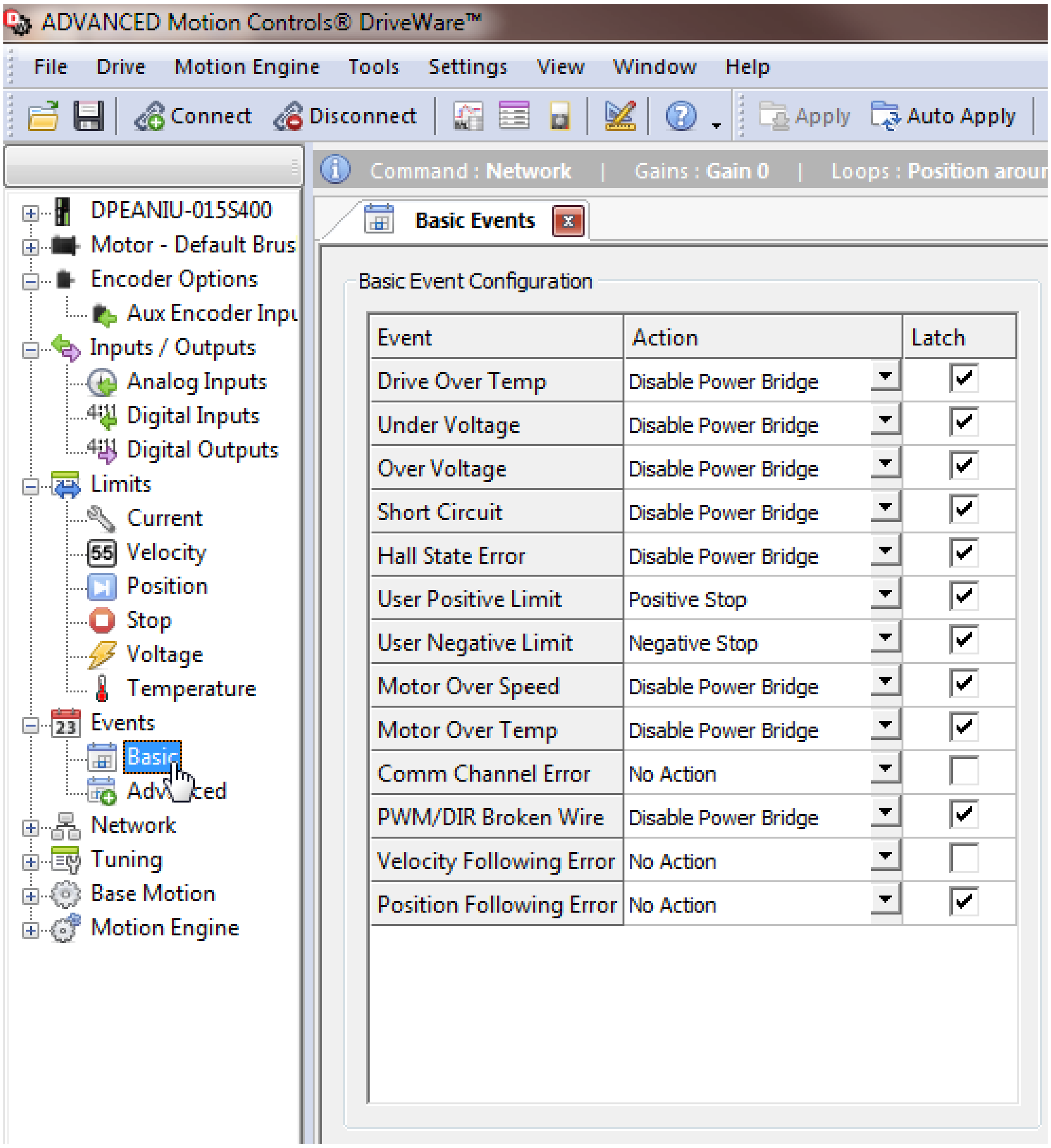


Stop Limits

Stop Deceleration Limit

Position Mode: 33333.3 rpm/s

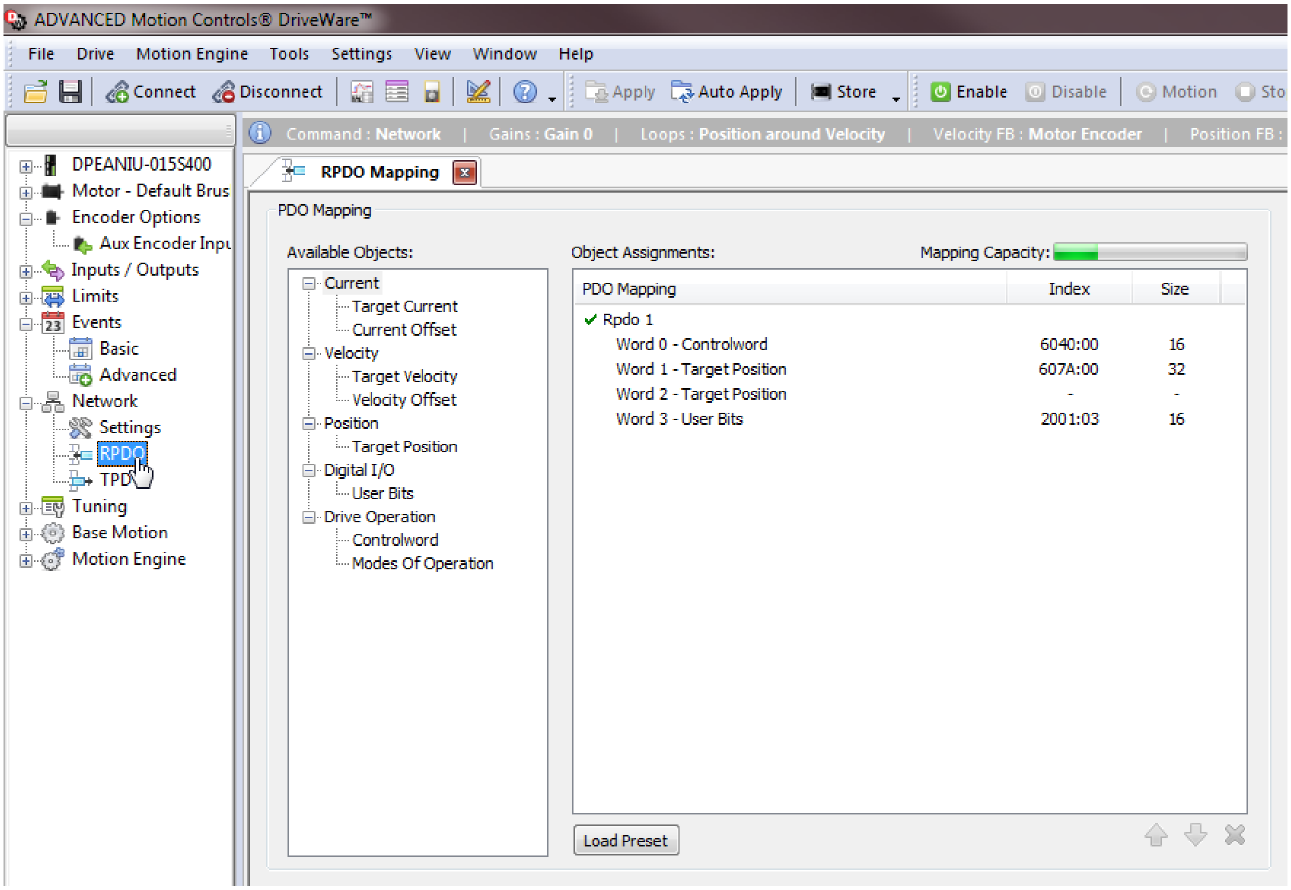
Velocity Mode: 33333.3 rpm/s



Basic Event Configuration

Event	Action	Latch
Drive Over Temp	Disable Power Bridge	☑
Under Voltage	Disable Power Bridge	☑
Over Voltage	Disable Power Bridge	☑
Short Circuit	Disable Power Bridge	☑
Hall State Error	Disable Power Bridge	☑
User Positive Limit	Positive Stop	☑
User Negative Limit	Negative Stop	☑
Motor Over Speed	Disable Power Bridge	☑
Motor Over Temp	Disable Power Bridge	☑
Comm Channel Error	No Action	□
PWM/DIR Broken Wire	Disable Power Bridge	☑
Velocity Following Error	No Action	□
Position Following Error	No Action	☑

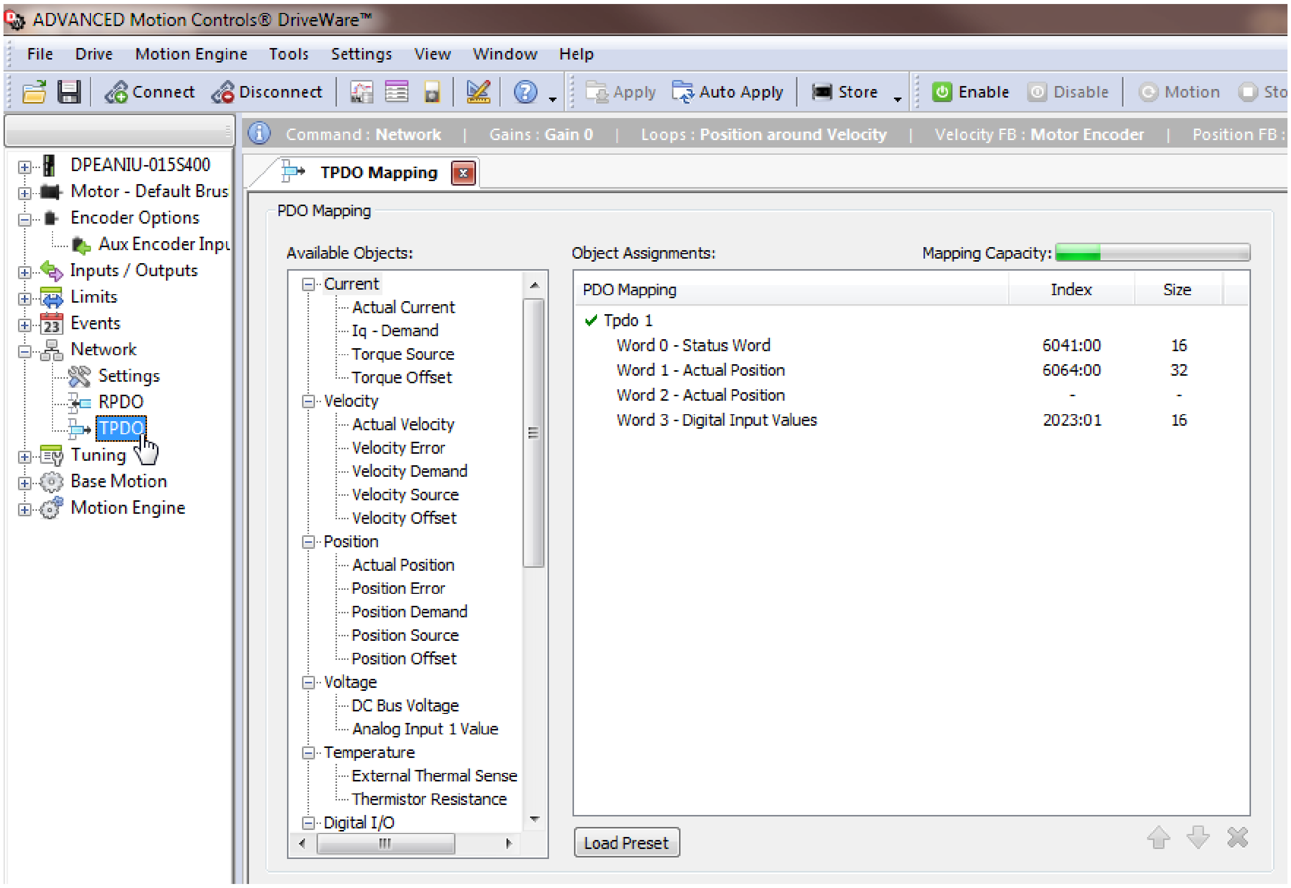
Comm Channel Error Event will automatically be enabled by the M3-41 module, with “Disable Power Bridge” as the default, unless otherwise selected on the above setup screen.



Command : **Network** | Gains : **Gain 0** | Loops : **Position around Velocity** | Velocity FB : **Motor Encoder** | Position FB :

PDO Mapping	Index	Size
✓ Rpdo 1		
Word 0 - Controlword	6040:00	16
Word 1 - Target Position	607A:00	32
Word 2 - Target Position	-	-
Word 3 - User Bits	2001:03	16

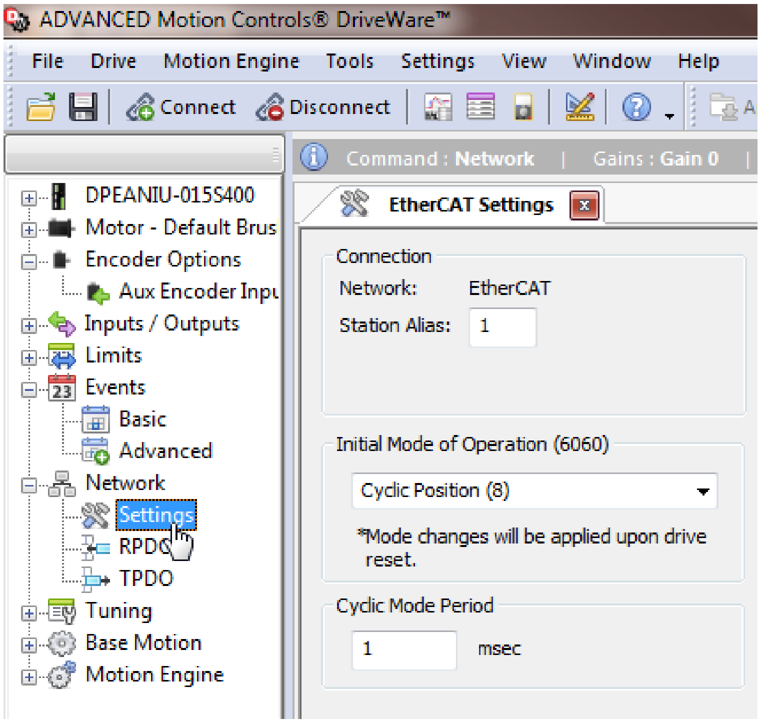
RPDO will automatically be set by the M3-41 module, overriding any setting done in the above window.



Command : Network | Gains : Gain 0 | Loops : Position around Velocity | Velocity FB : Motor Encoder | Position FB :

PDO Mapping	Index	Size
✓ Tpdo 1		
Word 0 - Status Word	6041:00	16
Word 1 - Actual Position	6064:00	32
Word 2 - Actual Position	-	-
Word 3 - Digital Input Values	2023:01	16

TPDO will automatically be set by the M3-41 module, overriding any setting done in the above window.



Command : Network | Gains : Gain 0 |

Connection

Network: EtherCAT

Station Alias: 1

Initial Mode of Operation (6060)

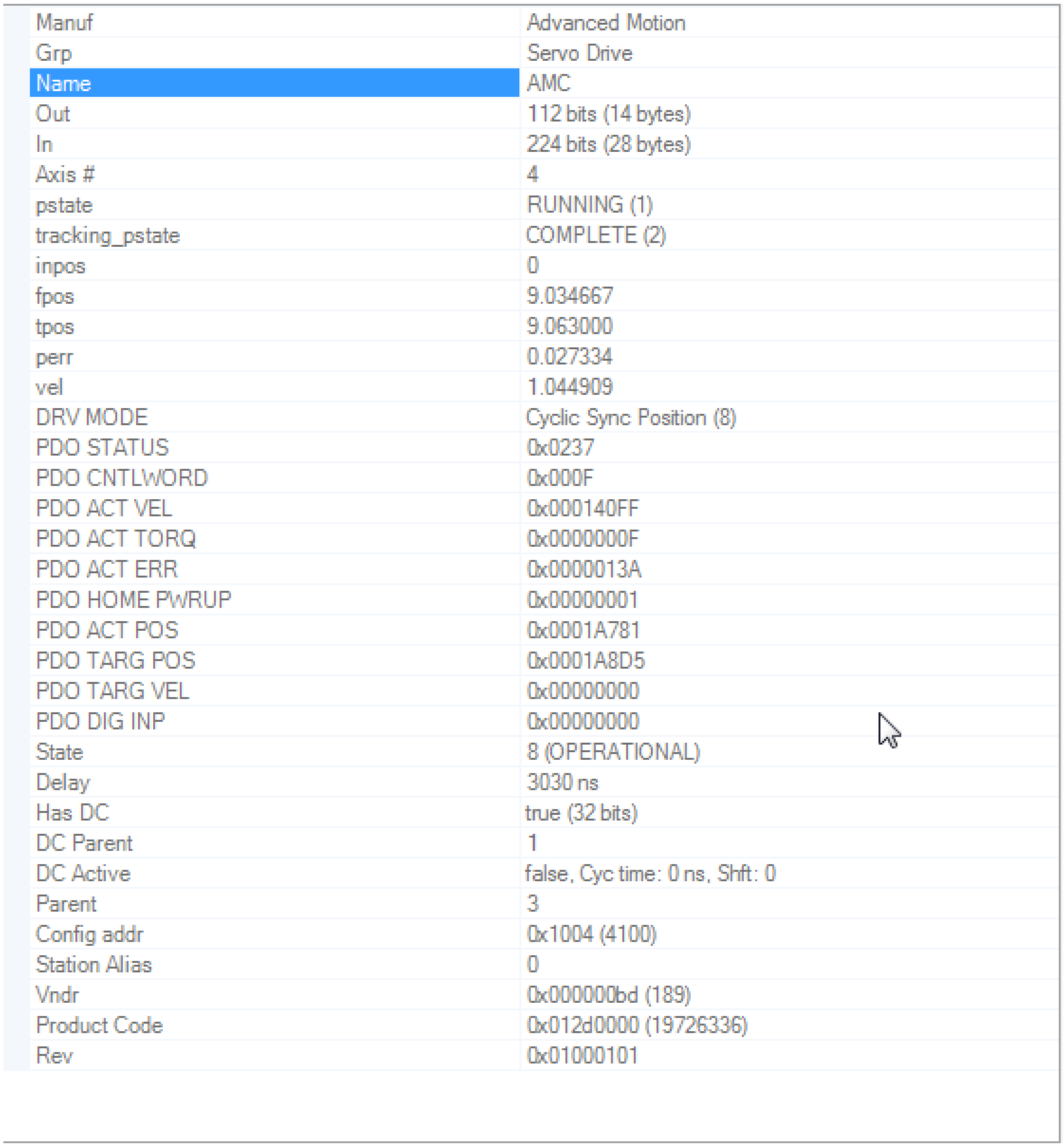
Cyclic Position (8)

*Mode changes will be applied upon drive reset.

Cyclic Mode Period

1 msec

EtherCAT Explorer View



Manuf	Advanced Motion
Grp	Servo Drive
Name	AMC
Out	112 bits (14 bytes)
In	224 bits (28 bytes)
Axis #	4
pstate	RUNNING (1)
tracking_pstate	COMPLETE (2)
inpos	0
fpos	9.034667
tpos	9.063000
perr	0.027334
vel	1.044909
DRV MODE	Cyclic Sync Position (8)
PDO STATUS	0x0237
PDO CNTLWORD	0x000F
PDO ACT VEL	0x000140FF
PDO ACT TORQ	0x0000000F
PDO ACT ERR	0x0000013A
PDO HOME PWRUP	0x00000001
PDO ACT POS	0x0001A781
PDO TARG POS	0x0001A8D5
PDO TARG VEL	0x00000000
PDO DIG INP	0x00000000
State	8 (OPERATIONAL)
Delay	3030 ns
Has DC	true (32 bits)
DC Parent	1
DC Active	false, Cyc time: 0 ns, Shft: 0
Parent	3
Config addr	0x1004 (4100)
Station Alias	0
Vndr	0x000000bd (189)
Product Code	0x012d0000 (19726336)
Rev	0x01000101

Blank

Appendix

C

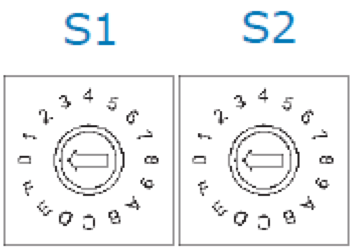
[C] Copley (Accelnet & Xenus Plus)

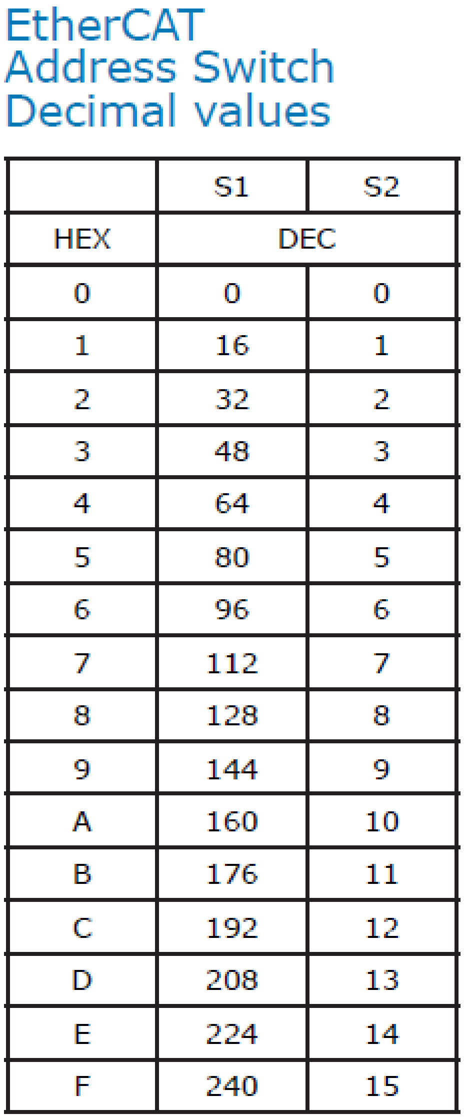
Copley Controls manufactures a number of drives. The drive currently supported is the single axis Accelnet servo drive. This section provides information that may be specific to this manufacturer.

eCAT_driveType – 2

Station Alias

In an EtherCAT network, slaves are automatically assigned addresses based on their position in the bus. When a device, such as a drive, must have a fixed assigned identification that is independent of cabling, a Station Alias is needed. Accelnet provides two 16-position rotary switches with hexadecimal encoding for this purpose. This allows for a setting of 0 to 255 (FFh), where 0 defaults to the automatic address assignment. As an example if S1 is set to a 1 and S2 to an A this would be 1Ah or 1 X 16 + 10 = 26. Since the M3-41 only supports up to 16 drives S1 would always be set to 0 and only S2 used.

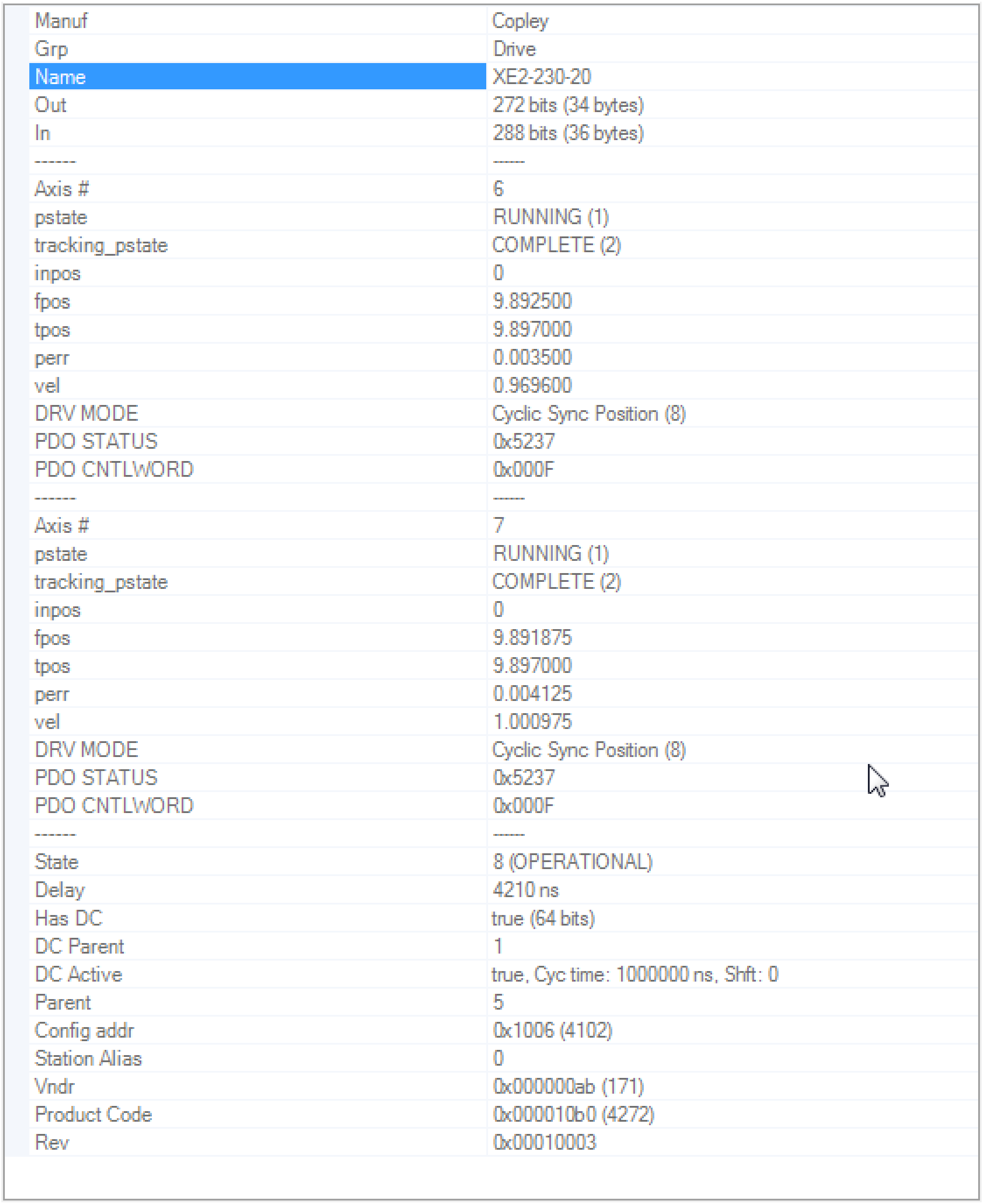




EtherCAT
Address Switch
Decimal values

	S1	S2
HEX	DEC	
0	0	0
1	16	1
2	32	2
3	48	3
4	64	4
5	80	5
6	96	6
7	112	7
8	128	8
9	144	9
A	160	10
B	176	11
C	192	12
D	208	13
E	224	14
F	240	15

EtherCAT Explorer View (Xenus Dual Axis)



Manuf	Copley
Grp	Drive
Name	XE2-230-20
Out	272 bits (34 bytes)
In	288 bits (36 bytes)
------	------
Axis #	6
pstate	RUNNING (1)
tracking_pstate	COMPLETE (2)
inpos	0
fpos	9.892500
tpos	9.897000
perr	0.003500
vel	0.969600
DRV MODE	Cyclic Sync Position (8)
PDO STATUS	0x5237
PDO CNTLWORD	0x000F
------	------
Axis #	7
pstate	RUNNING (1)
tracking_pstate	COMPLETE (2)
inpos	0
fpos	9.891875
tpos	9.897000
perr	0.004125
vel	1.000975
DRV MODE	Cyclic Sync Position (8)
PDO STATUS	0x5237
PDO CNTLWORD	0x000F
-------	------
State	8 (OPERATIONAL)
Delay	4210 ns
Has DC	true (64 bits)
DC Parent	1
DC Active	true, Cyc time: 1000000 ns, Shft: 0
Parent	5
Config addr	0x1006 (4102)
Station Alias	0
Vndr	0x000000ab (171)
Product Code	0x000010b0 (4272)
Rev	0x00010003

Blank

Appendix

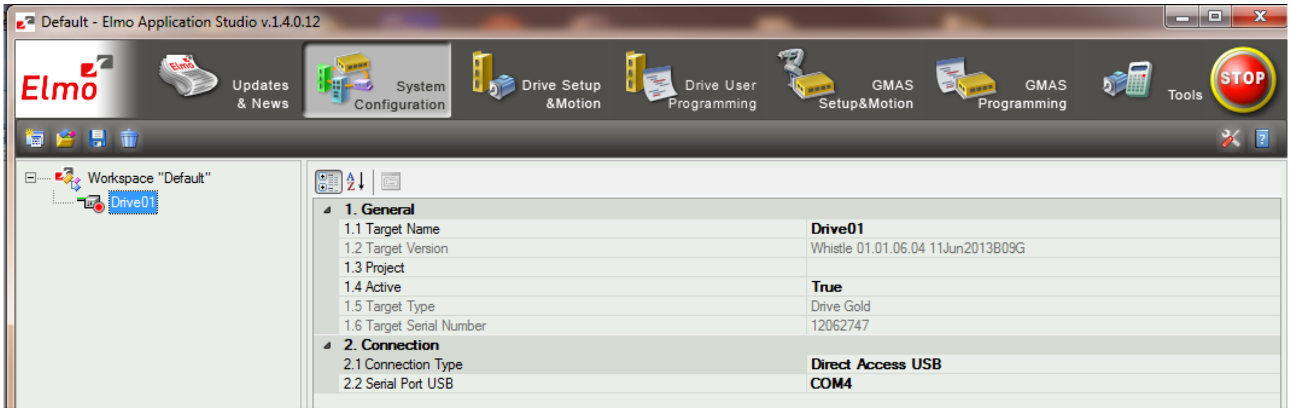
D

[D] Elmo (Not Released)

Elmo manufactures a number of drives. The drive currently supported is the single axis Elmo Gold Whistle servo drive. This section provides information that may be specific to this manufacturer.

eCAT_driveType – 4

Drive Information & Firmware



1. General	
1.1 Target Name	Drive01
1.2 Target Version	Whistle 01.01.06.04 11Jun2013B09G
1.3 Project	
1.4 Active	True
1.5 Target Type	Drive Gold
1.6 Target Serial Number	12062747
2. Connection	
2.1 Connection Type	Direct Access USB
2.2 Serial Port USB	COM4

Blank

Appendix

E

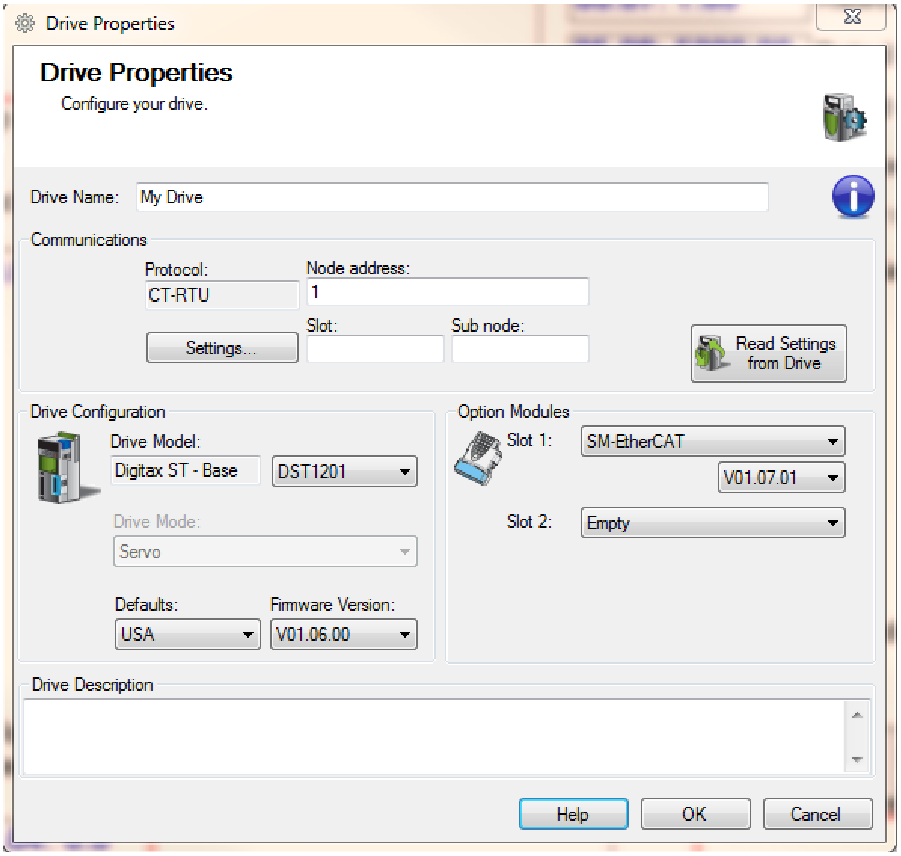
[E] Emerson (Control Techniques)

Emerson (Control Techniques) manufactures a number of drives. The drive currently supported is the single axis Unidrive SP and Digitax ST servo drive. This section provides information that may be specific to this manufacturer.

eCAT_driveType – 7

Drive Information & Firmware

Digitax:



Drive Properties

Drive Properties

Configure your drive.

Drive Name: My Drive

Communications

Protocol: CT-RTU

Node address: 1

Slot:

Sub node:

Drive Configuration

Drive Model: Digitax ST - Base DST1201

Drive Mode: Servo

Defaults: USA

Firmware Version: V01.06.00

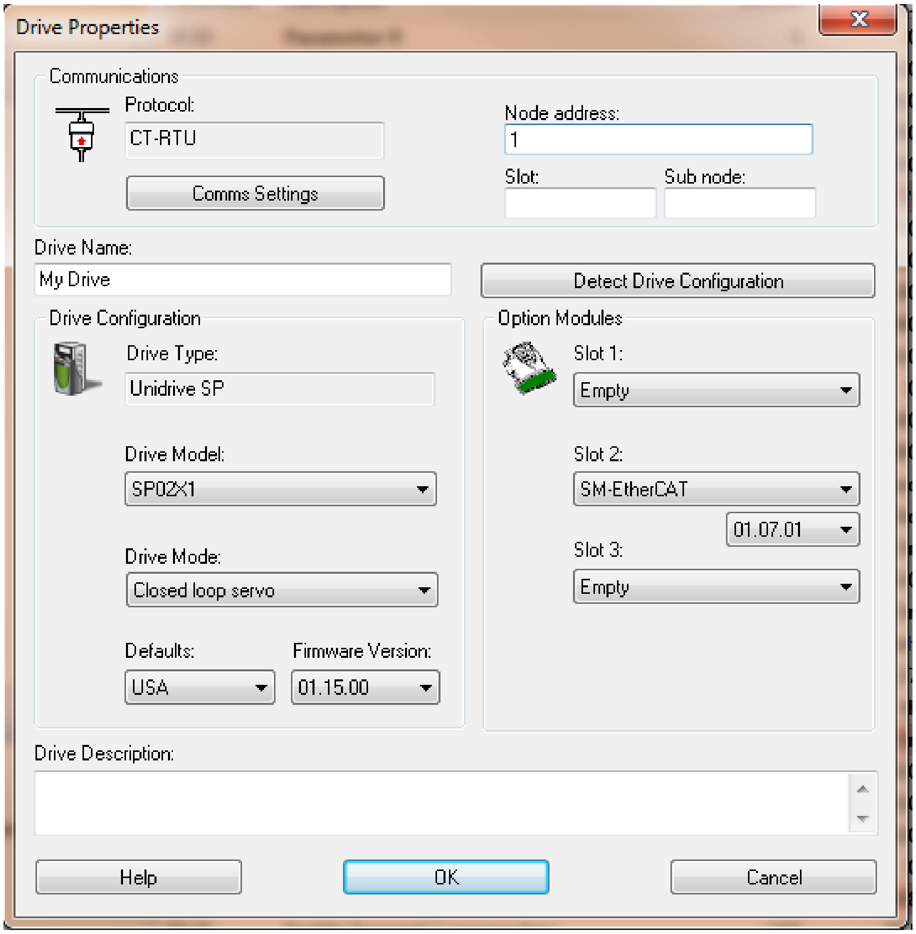
Option Modules

Slot 1: SM-EtherCAT V01.07.01

Slot 2: Empty

Drive Description

Unidrive SP:



Protocol: CT-RTU

Node address: 1

Slot:

Sub node:

Drive Name: My Drive

Drive Type: Unidrive SP

Drive Model: SP02X1

Drive Mode: Closed loop servo

Defaults: USA

Firmware Version: 01.15.00

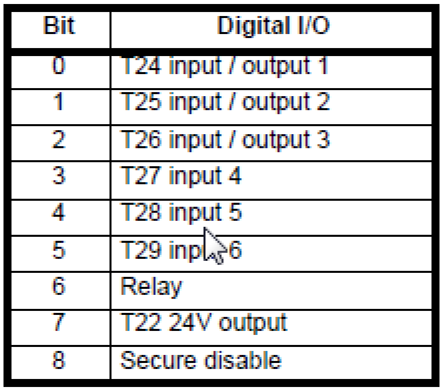
Slot 1: Empty

Slot 2: SM-EtherCAT 01.07.01

Slot 3: Empty

Inputs/Outputs

CTSoft Menu 8 is used to set up Digital I/O on the Unidrive SP. This is object 0x2008 when mapped to EtherCAT with the subindex being the menu item number. Thus item 20 is a read-only Digital I/O state object; item 20 in hex is subindex 0x14.



Bit	Digital I/O
0	T24 input / output 1
1	T25 input / output 2
2	T26 input / output 3
3	T27 input 4
4	T28 input 5
5	T29 input 6
6	Relay
7	T22 24V output
8	Secure disable

To activate outputs, a specific ‘sdo write’ operation is required, since the Emerson drive only has single-bit access to outputs, and the first 3 inputs are reconfigurable. In referencing the parameters as defined in the “Advanced User Guide Unidrive SP” (Part Number 0471-0002-10), subindex 31 to 33 enables T24 to T26 as outputs. Subindex 21 to 23 maps where the outputs would come from depending on whether an input or output.

DC Sync

Emerson drives must have DC Sync0 enabled prior to enabling the drive as well as a 32 or 64-bit slave reference as the first node on the EtherCAT network. Reference the ‘dcsync’ command example within the main body of this manual.

MSB code sample prior to drive enable:

```
// This is only needed for Emerson/Control Techniques
delay 2000 ms;  // needed in case restart so syncs when cycle DC Sync on/off.
// Cycle time is 1 ms, start it 100 ms in the future.
// Note that we need to make sure that the first slave device in the EtherCAT
// Network supports 32/64-bit distributed clocks for this to work properly.
// Thus far that is Beckhoff, Wago, and Sanyo Denki.
dcsync -1, 1000000, 0, 0, 100000000;
delay 200 ms;     // starts 100 milliseconds into the future

/**************** ENABLE DRIVE *******************/
[Drive_Enable]
// Issue any hardware enable output commands
//
// Power up the drive amplifier
drive enable;
```

Drive Enable Command

Emerson drives appear to have an anomaly which in some cases can cause the motor to creep or move when it is not commanded. This may have something to do with the Pr3.22 (Hard speed reference) setting but has not been verified. During EtherCAT initialization the drive is placed in Cyclic Sync Position mode and it target position is set equal to its present position, which should result in no motion. If an external hardware enable is used and this enable is turned on prior to the MSB executing unpredictable results can occur since the M3-41 is not in full control. From an EtherCAT perspective the drive is disabled via the control word but from a drive perspective it may move based upon internal parameter settings, overriding the EtherCAT command.

The ideal way to overcome this problem is to have the drive’s MSB issue the command to enable the hardware enable after DC Sync has been executed but just prior to the ‘drive enable’ command. This should be tested in a safe environment with the proper interlocks. A second approach, which has been proven to work, would be to start the axis MSB from QuickBuilder, have the MSB enable DC Sync and just prior to executing the ‘drive enable’ set a flag for the QuickBuilder application to observe. While the MSB hangs in the ‘drive enable’ command waiting for the hardware enable the QuickBuilder application would then activate the drives hardware enable using outputs under its control followed by monitoring the drive's 'enabled' flag to know when the drive amplifier has been fully powered up and is holding position.

Station Alias

The station alias may be set through the main menu screen, selection Pr 15.03. A setting of 0 is for automatic. Note that once an alias is set, if you cycle power, the Emerson drive will display a 0 even though that is not what is set. This was an anomaly in Emerson's firmware at the time testing was done. If an alias is displayed in the QuickBuilder EtherCAT Explorer and a 0 appears in Pr 15.03, you must manually select a different value and then go back to 0 and enter it, cycling power.

Menu to Object Mapping

The format used when mapping drive parameters to PDOs is as follows:

- Index: 0x2000 + menu number
- Sub-index: 0x00 + parameter number
- Size: Dependant on the size (in bytes) of the object to be mapped (range: 1-4)

For example Pr **20.21** would be index 0x2014, sub-index 0x15 and the size would be 4 (it is a 32-bit signed value when referenced in the manual).

RPM Limiting

Should an issue occur where the default maximum RPM of 3000 cannot be overridden by saving the parameters using the Emerson Configuration software, or it needs to be modified on the fly, you may do an SDO write to object 0x6080, unsigned 32 bit integer, with the desired maximum RPM.

Appendix F

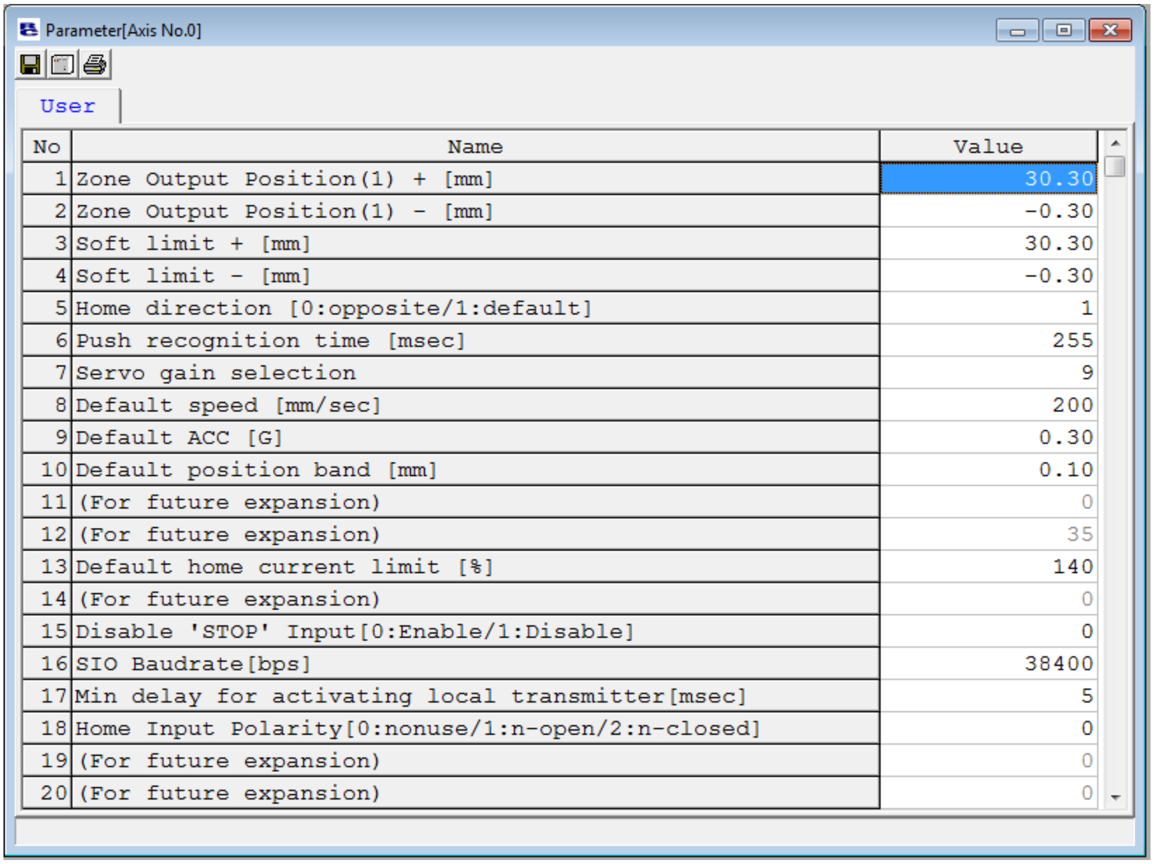
[F] IAI

IAI is a manufacturer of intelligent linear actuators. The M3-41 supports the ACON controller in Full Direct Value Mode. The IAI Windows based ROBO Cylinder Software was used for initial setup of the controller. This section provides information that may be specific to this manufacturer.

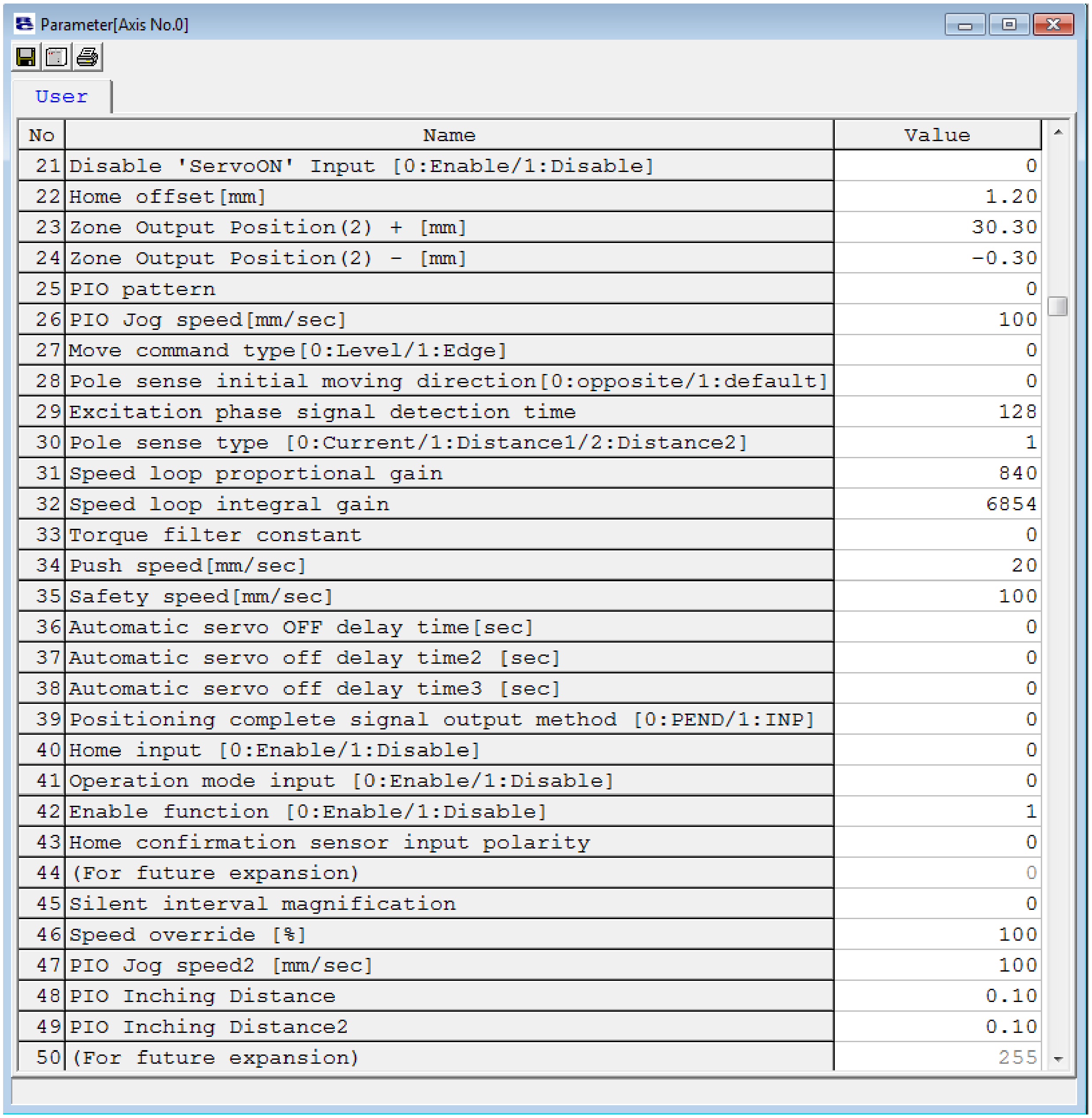
eCAT_driveType – 11

Parameter Information from Test Setup

Test setup: Controller – ACON-C-30I-EC-0-0, Actuator - RCA2-TF4N-I-20-4-30-A3-P



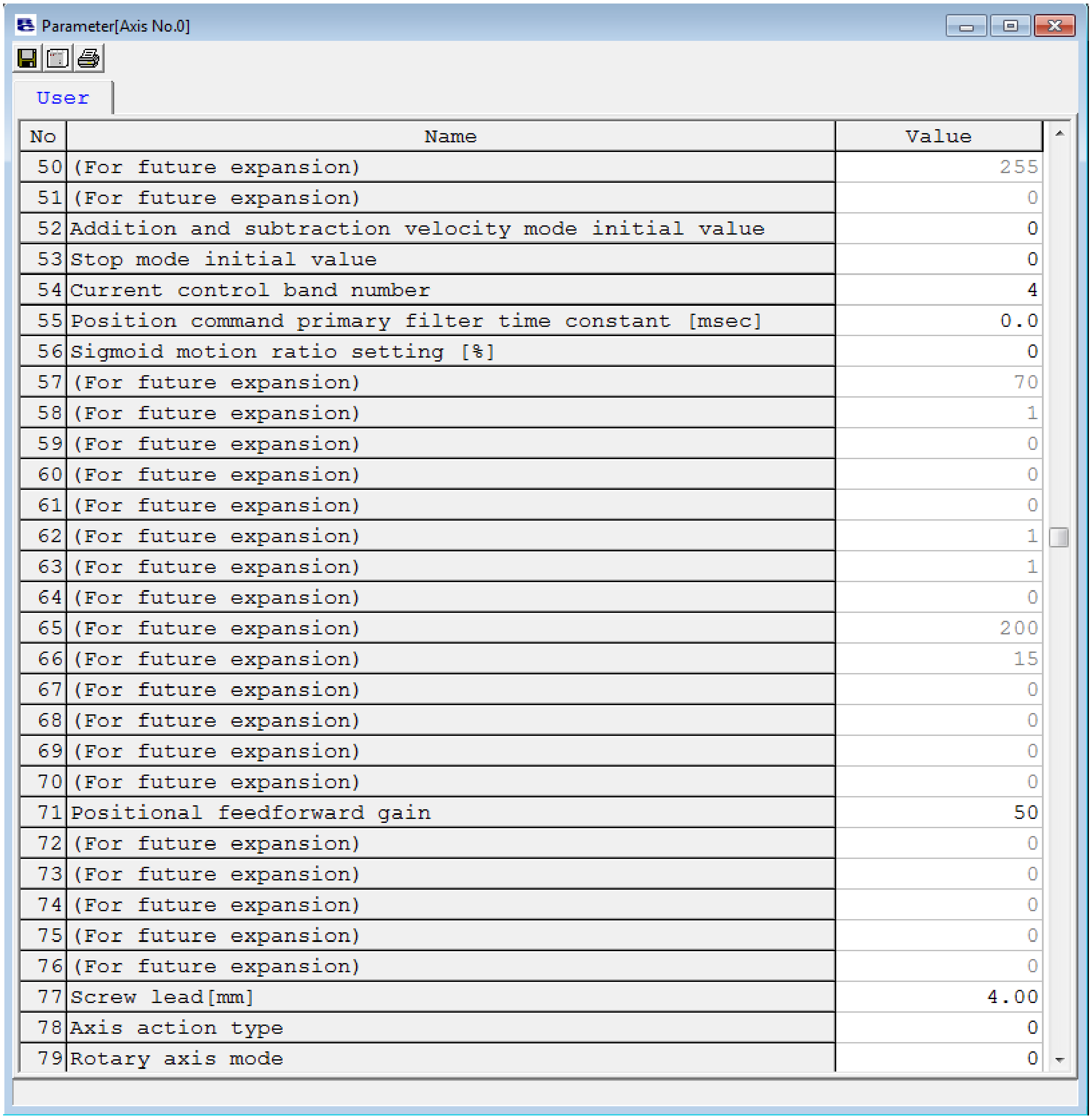
No	Name	Value
1	Zone Output Position(1) + [mm]	30.30
2	Zone Output Position(1) - [mm]	-0.30
3	Soft limit + [mm]	30.30
4	Soft limit - [mm]	-0.30
5	Home direction [0:opposite/1:default]	1
6	Push recognition time [msec]	255
7	Servo gain selection	9
8	Default speed [mm/sec]	200
9	Default ACC [G]	0.30
10	Default position band [mm]	0.10
11	(For future expansion)	0
12	(For future expansion)	35
13	Default home current limit [%]	140
14	(For future expansion)	0
15	Disable 'STOP' Input[0:Enable/1:Disable]	0
16	SIO Baudrate[bps]	38400
17	Min delay for activating local transmitter[msec]	5
18	Home Input Polarity[0:nonuse/1:n-open/2:n-closed]	0
19	(For future expansion)	0
20	(For future expansion)	0



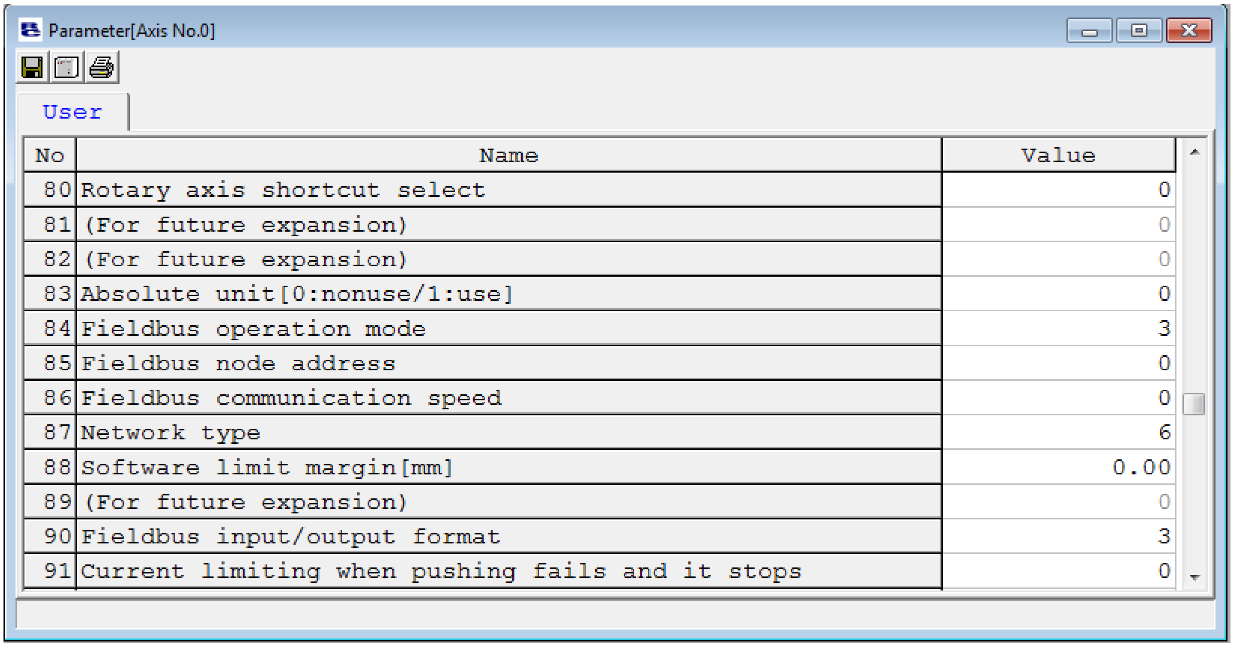
Parameter[Axis No.0]

User

No	Name	Value
21	Disable 'ServoON' Input [0:Enable/1:Disable]	0
22	Home offset[mm]	1.20
23	Zone Output Position(2) + [mm]	30.30
24	Zone Output Position(2) - [mm]	-0.30
25	PIO pattern	0
26	PIO Jog speed[mm/sec]	100
27	Move command type[0:Level/1:Edge]	0
28	Pole sense initial moving direction[0:opposite/1:default]	0
29	Excitation phase signal detection time	128
30	Pole sense type [0:Current/1:Distance1/2:Distance2]	1
31	Speed loop proportional gain	840
32	Speed loop integral gain	6854
33	Torque filter constant	0
34	Push speed[mm/sec]	20
35	Safety speed[mm/sec]	100
36	Automatic servo OFF delay time[sec]	0
37	Automatic servo off delay time2 [sec]	0
38	Automatic servo off delay time3 [sec]	0
39	Positioning complete signal output method [0:PEND/1:INP]	0
40	Home input [0:Enable/1:Disable]	0
41	Operation mode input [0:Enable/1:Disable]	0
42	Enable function [0:Enable/1:Disable]	1
43	Home confirmation sensor input polarity	0
44	(For future expansion)	0
45	Silent interval magnification	0
46	Speed override [%]	100
47	PIO Jog speed2 [mm/sec]	100
48	PIO Inching Distance	0.10
49	PIO Inching Distance2	0.10
50	(For future expansion)	255



No	Name	Value
50	(For future expansion)	255
51	(For future expansion)	0
52	Addition and subtraction velocity mode initial value	0
53	Stop mode initial value	0
54	Current control band number	4
55	Position command primary filter time constant [msec]	0.0
56	Sigmoid motion ratio setting [%]	0
57	(For future expansion)	70
58	(For future expansion)	1
59	(For future expansion)	0
60	(For future expansion)	0
61	(For future expansion)	0
62	(For future expansion)	1
63	(For future expansion)	1
64	(For future expansion)	0
65	(For future expansion)	200
66	(For future expansion)	15
67	(For future expansion)	0
68	(For future expansion)	0
69	(For future expansion)	0
70	(For future expansion)	0
71	Positional feedforward gain	50
72	(For future expansion)	0
73	(For future expansion)	0
74	(For future expansion)	0
75	(For future expansion)	0
76	(For future expansion)	0
77	Screw lead[mm]	4.00
78	Axis action type	0
79	Rotary axis mode	0



No	Name	Value
80	Rotary axis shortcut select	0
81	(For future expansion)	0
82	(For future expansion)	0
83	Absolute unit[0:nonuse/1:use]	0
84	Fieldbus operation mode	3
85	Fieldbus node address	0
86	Fieldbus communication speed	0
87	Network type	6
88	Software limit margin[mm]	0.00
89	(For future expansion)	0
90	Fieldbus input/output format	3
91	Current limiting when pushing fails and it stops	0

Note that Parameter #84, **Fieldbus operation mode is set to 3**. This setting is required for Full Direct Value Mode. The factory default setting is 0 therefore a change is required. Make sure the switch on the front panel is set to Manual, modify the parameter, then click the “Load to CTL” icon. Once loaded, restore the switch to the Auto position for EtherCAT communications.

Parameter #85 is the Fieldbus node address, in EtherCAT terminology this is the Station Alias. The factory default is 17. **For automatic axis assignment set this address to 0**, otherwise an axis number from 1 to 16 that is unique for the network being controlled by the M3-41 module.

Special Variable Mapping

Due to the unique nature of the IAI interface a number of MSB variables have been mapped to be cyclically updated on each EtherCAT scan cycle. These variables are:

Outputs:

cmode – Mode of motion control, must be set to $HOMING_MODE or $PROFILE_POSITION_MODE.

homing_speed1 – Homing speed in mm/sec. Used in conjunction with the $HOMING_MODE ‘cmode’ command.

zone_limit_pos – Zone Boundry + in mm. Positive zone boundary limit. After completion of home return, an effective zone signal can be output separately from the zone boundaries specified by parameters. The status signal PZONE turns ON when the current position is inside the Zone Boundary +/- boundaries.

zone_limit_neg – Zone Boundry - in mm. Negative zone boundary limit. After completion of home return, an effective zone signal can be output separately from the zone boundaries specified by parameters. The status signal PZONE turns ON when the current position is inside the Zone Boundary +/- boundaries.

IAI_conrol – Maps to IAI Control Signal 1 & 2. Reference the Control Output Mapping section that follows.

tmax – Pressing Current Limit (%), specifies the current limit value to be used during the push-motion operation. The allowable specification range is 0 to 100%. The actual settable range varies depending on each actuator. If a move command is issued by specifying a value exceeding the maximum push-motion current and alarm and fault will occur.

tlim – Load Current Threshold (%), sets the current threshold when whether or not the load current exceeds the threshold is judged. The allowable range is 0 to 100% with 0 the threshold judgment if it is not required.

Inputs:

dins – The 'dins' variable maps to the IAI Control Status inputs. Reference the Control Status Mapping section that follows.

rmstrq – IAI Current in mA.

IAI_alarmcode - Maps directly to the IAI Alarm Code for general monitoring. This is the code that is referenced when a fault occurs and an EtherCAT Explorer log is generated.

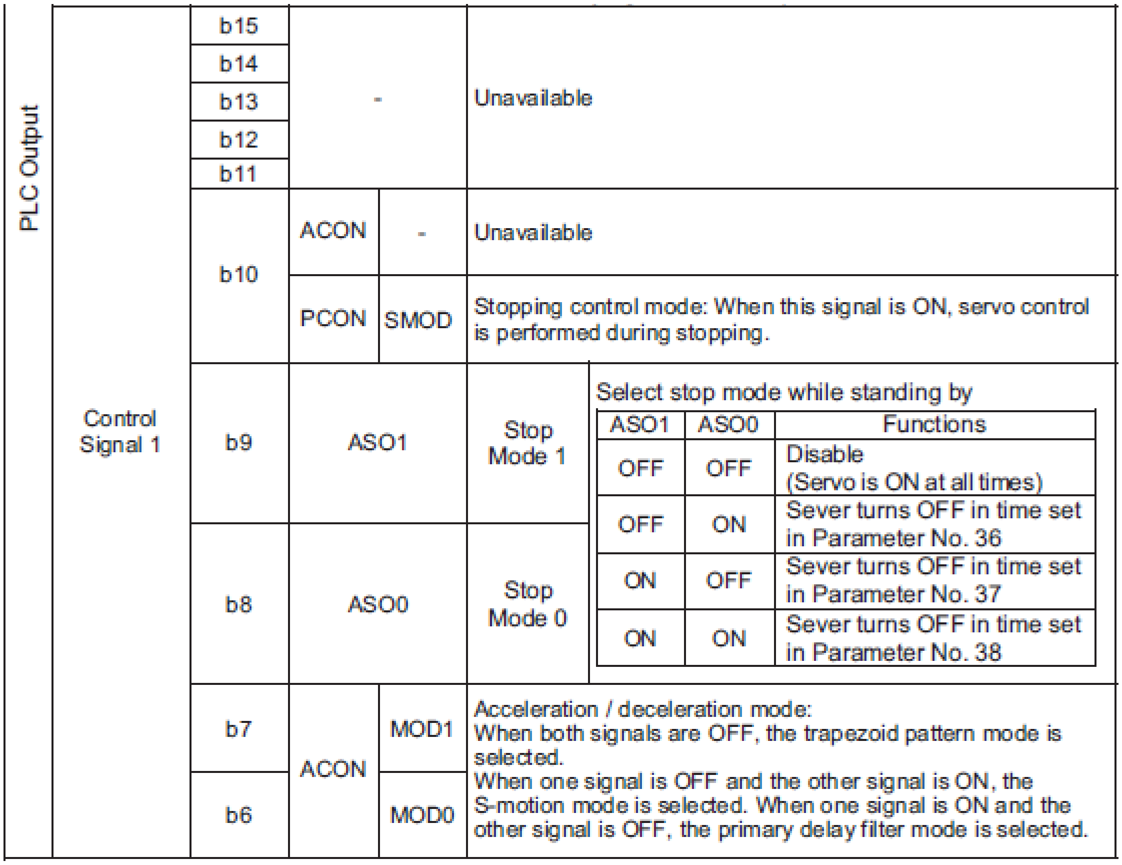
PDO Mappings for EtherCAT Explorer:

PDO CNTLWORD – IAI Control Signal 2 outputs.

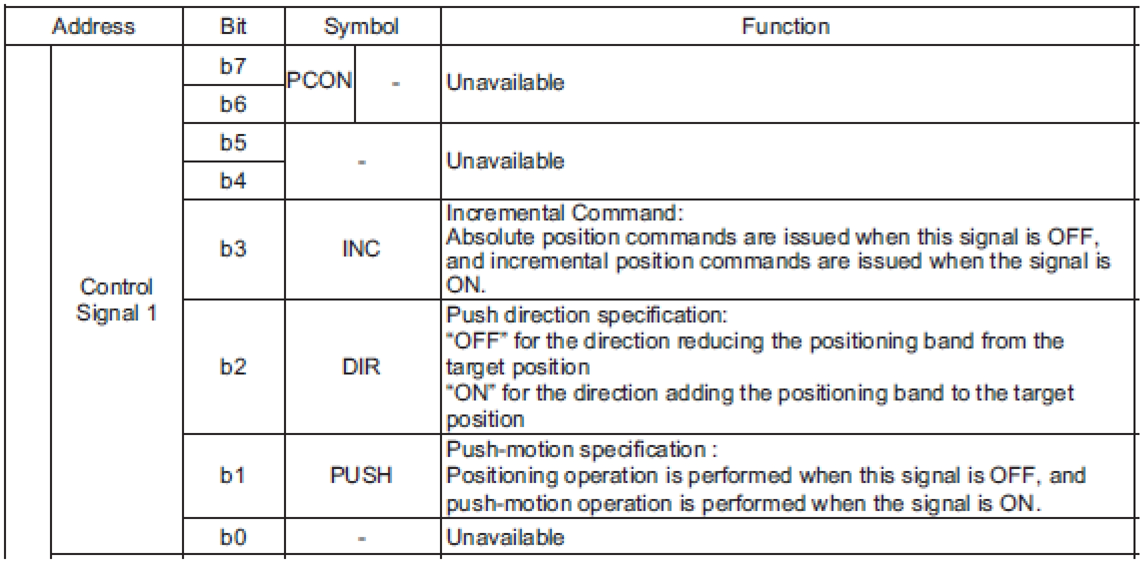
PDO STATUS – IAI Control Status inputs.

Control Output Mapping

The ACON controller has two Control Signal Output blocks, 1 & 2. These can be accessed directly referencing the 'IAI_control' MSB variable. Any changes to this variable will be output on the next EtherCAT scan cycle. 'IAI_control' is a 32 bit value with its lower 16 bit word being Control Signal 1 and upper 16 bits Control Signal 2:

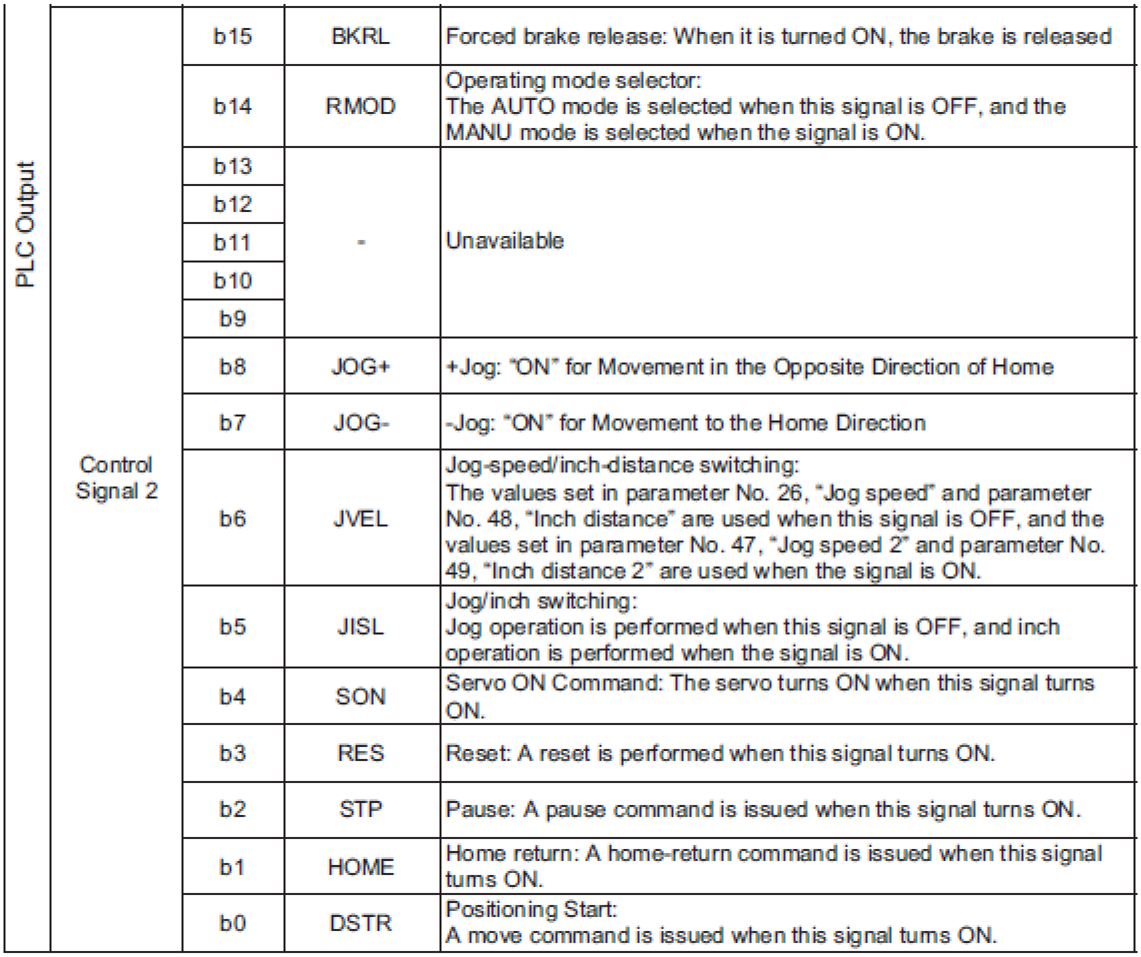


<table>
<tr><td rowspan="14">PLC Output</td><td rowspan="14">Control Signal 1</td><td>b15</td><td colspan="2" rowspan="5">-</td><td colspan="2" rowspan="5">Unavailable</td></tr>
<tr><td>b14</td></tr>
<tr><td>b13</td></tr>
<tr><td>b12</td></tr>
<tr><td>b11</td></tr>
<tr><td rowspan="2">b10</td><td>ACON</td><td>-</td><td colspan="2">Unavailable</td></tr>
<tr><td>PCON</td><td>SMOD</td><td colspan="2">Stopping control mode: When this signal is ON, servo control is performed during stopping.</td></tr>
<tr><td>b9</td><td colspan="2">ASO1</td><td>Stop Mode 1</td><td rowspan="2">Select stop mode while standing by<table><tr><th>ASO1</th><th>ASO0</th><th>Functions</th></tr><tr><td>OFF</td><td>OFF</td><td>Disable
(Servo is ON at all times)</td></tr><tr><td>OFF</td><td>ON</td><td>Sever turns OFF in time set in Parameter No. 36</td></tr><tr><td>ON</td><td>OFF</td><td>Sever turns OFF in time set in Parameter No. 37</td></tr><tr><td>ON</td><td>ON</td><td>Sever turns OFF in time set in Parameter No. 38</td></tr></table></td></tr>
<tr><td>b8</td><td colspan="2">ASO0</td><td>Stop Mode 0</td></tr>
<tr><td>b7</td><td rowspan="2">ACON</td><td>MOD1</td><td colspan="2" rowspan="2">Acceleration / deceleration mode:
When both signals are OFF, the trapezoid pattern mode is selected.
When one signal is OFF and the other signal is ON, the S-motion mode is selected. When one signal is ON and the other signal is OFF, the primary delay filter mode is selected.</td></tr>
<tr><td>b6</td><td>MOD0</td></tr>
</table>



Address	Bit	Symbol		Function
Control Signal 1	b7	PCON	-	Unavailable
	b6			
	b5	-		Unavailable
	b4			
	b3	INC		Incremental Command: Absolute position commands are issued when this signal is OFF, and incremental position commands are issued when the signal is ON.
	b2	DIR		Push direction specification: "OFF" for the direction reducing the positioning band from the target position "ON" for the direction adding the positioning band to the target position
	b1	PUSH		Push-motion specification : Positioning operation is performed when this signal is OFF, and push-motion operation is performed when the signal is ON.
	b0	-		Unavailable

All but 'b3' is masked low level and available to the programmer. 'b3' is forced to 0 since the MSB command language supports incremental commands and automatically converts to absolute. When not using PUSH mode Control Signal 1 defaults to a 0, as does the variable 'IAI_control'. Control Signal 1 represents bits 0 to 15 in 'IAI_control'.

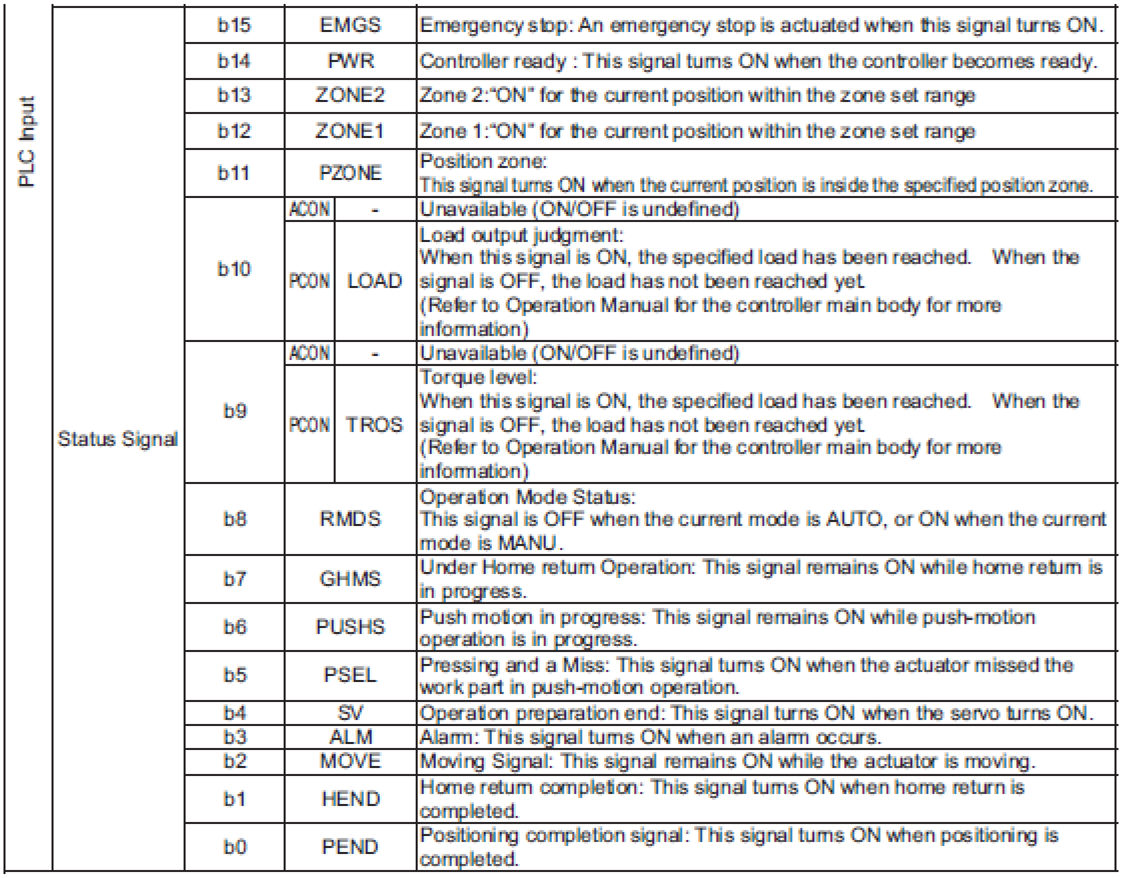


PLC Output	Control Signal 2	b15	BKRL	Forced brake release: When it is turned ON, the brake is released
		b14	RMOD	Operating mode selector: The AUTO mode is selected when this signal is OFF, and the MANU mode is selected when the signal is ON.
		b13	-	Unavailable
		b12		
		b11		
		b10		
		b9		
		b8	JOG+	+Jog: "ON" for Movement in the Opposite Direction of Home
		b7	JOG-	-Jog: "ON" for Movement to the Home Direction
		b6	JVEL	Jog-speed/inch-distance switching: The values set in parameter No. 26, "Jog speed" and parameter No. 48, "Inch distance" are used when this signal is OFF, and the values set in parameter No. 47, "Jog speed 2" and parameter No. 49, "Inch distance 2" are used when the signal is ON.
		b5	JISL	Jog/inch switching: Jog operation is performed when this signal is OFF, and inch operation is performed when the signal is ON.
		b4	SON	Servo ON Command: The servo turns ON when this signal turns ON.
		b3	RES	Reset: A reset is performed when this signal turns ON.
		b2	STP	Pause: A pause command is issued when this signal turns ON.
		b1	HOME	Home return: A home-return command is issued when this signal turns ON.
		b0	DSTR	Positioning Start: A move command is issued when this signal turns ON.

All but ‘b4’ is masked low level and available to the programmer. ‘b4’ is controlled by the ‘drive enable’ command, set to a 1 upon execution and defaulting to 0 at power up. The M3-41 will control RES, HOME, and DSTR as needed under program control. RES is used to clear an alarm condition. Control Signal 2 represents bits 16 to 31 in ‘IAI_control’.

Control Status Mapping

The ACON controller has one 16 bit Control Status Input block. These can be accessed directly referencing the ‘dins’ and ‘din#’ MSB variables. The variable is updated every scan cycle. From the IAI manual the status flag are defined as below and represent what appears in the PDO STATUS row of the EtherCAT Explorer.



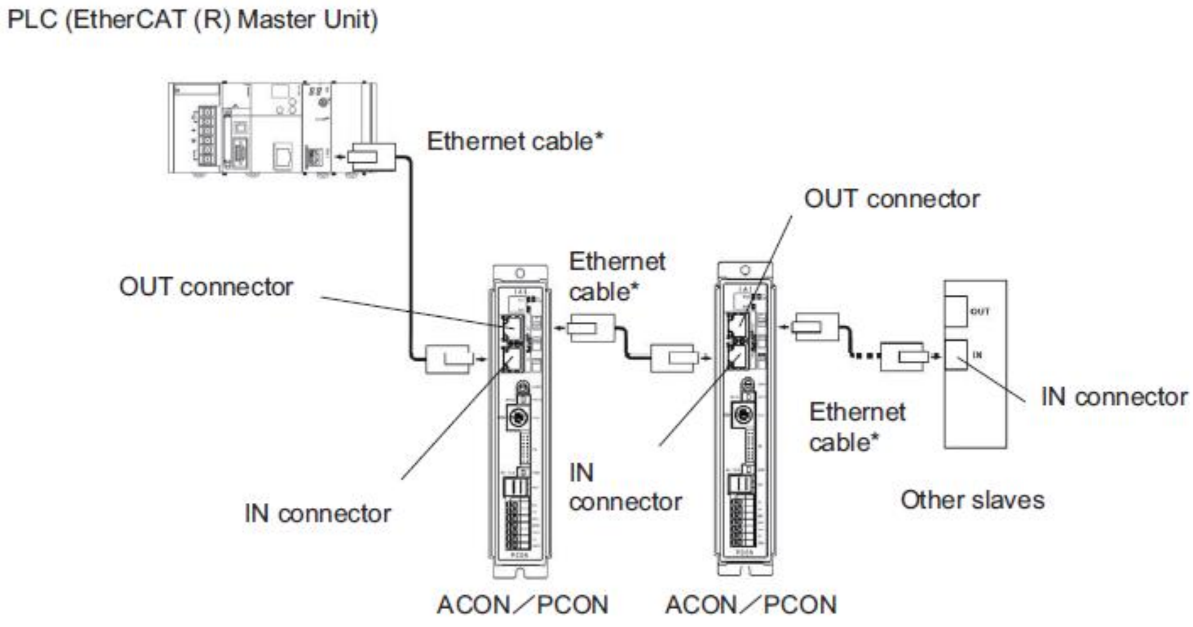
PLC Input	Status Signal	b15		EMGS	Emergency stop: An emergency stop is actuated when this signal turns ON.
		b14		PWR	Controller ready : This signal turns ON when the controller becomes ready.
		b13		ZONE2	Zone 2:"ON" for the current position within the zone set range
		b12		ZONE1	Zone 1:"ON" for the current position within the zone set range
		b11		PZONE	Position zone: This signal turns ON when the current position is inside the specified position zone.
		b10	ACON	-	Unavailable (ON/OFF is undefined)
			PCON	LOAD	Load output judgment: When this signal is ON, the specified load has been reached. When the signal is OFF, the load has not been reached yet. (Refer to Operation Manual for the controller main body for more information)
		b9	ACON	-	Unavailable (ON/OFF is undefined)
			PCON	TROS	Torque level: When this signal is ON, the specified load has been reached. When the signal is OFF, the load has not been reached yet. (Refer to Operation Manual for the controller main body for more information)
		b8		RMDS	Operation Mode Status: This signal is OFF when the current mode is AUTO, or ON when the current mode is MANU.
		b7		GHMS	Under Home return Operation: This signal remains ON while home return is in progress.
		b6		PUSHS	Push motion in progress: This signal remains ON while push-motion operation is in progress.
		b5		PSEL	Pressing and a Miss: This signal turns ON when the actuator missed the work part in push-motion operation.
		b4		SV	Operation preparation end: This signal turns ON when the servo turns ON.
		b3		ALM	Alarm: This signal turns ON when an alarm occurs.
		b2		MOVE	Moving Signal: This signal remains ON while the actuator is moving.
		b1		HEND	Home return completion: This signal turns ON when home return is completed.
		b0		PEND	Positioning completion signal: This signal turns ON when positioning is completed.

When referencing '*dins*' and '*din#*' variables the bits have been reordered to allow the use of the din1 to din10 bit referencing within the MSB language. Thus when referencing these variables the status flags are ordered as follows:

* b15 - LOAD (not used)
* b14 - TROS (not used)
* b13 - RMDS (Auto/manual)
* b12 - GHMS
* b11 - PUSHS
* b10 - PWR
* b09 - SV
* b08 - ALM
* b07 - MOVE
* b06 - HEND
* b05 - PEND
* b04 - EMGS
* b03 - PSEL
* b02 - ZONE2
* b01 - ZONE1
* b00 - PZONE

Operation

The ACON controller is initially configured using the ROBO Cylinder Software from IAI. This software allows for the setting of Parameters in non-volatile storage within the controller via a serial cable. As previously noted Fieldbus Operation Mode, parameter 84, should be set to a 3 for Profile Position mode type motions and Fieldbus Node Address, parameter 85, to 0 for automatic sequential axis assignment or an axis number from 1 to 16 (station alias). Once the parameters are saved the Manual/Auto switch on the front panel of the controller must be set to Auto for M3-41 EtherCAT control. The address switch should also be set to 0 and the BKLS switch to NORMAL.

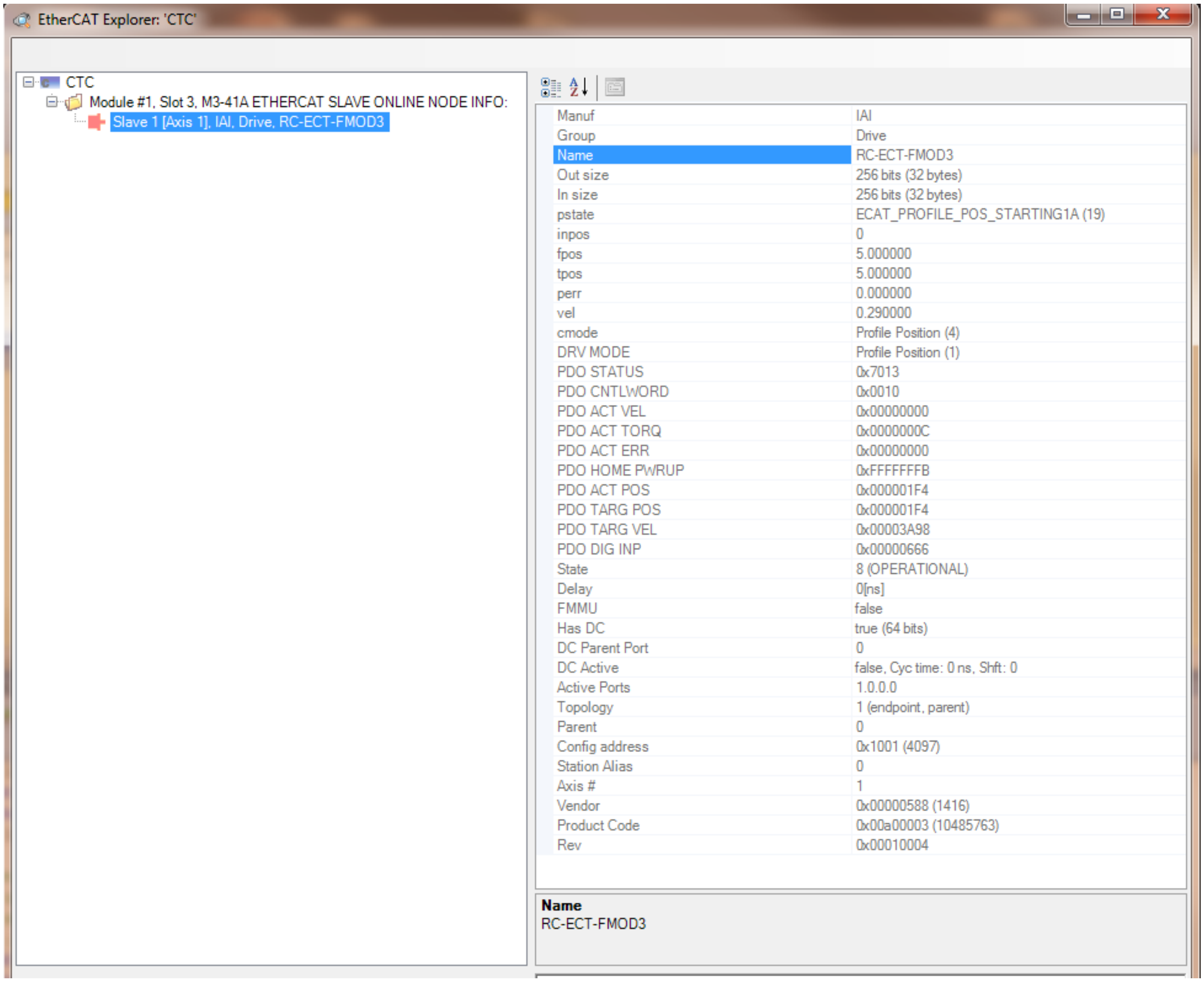


* Ethernet cable: Straight cable of category 5e or above, 100 m max
(Aluminum tape and braided double-shielded cable are recommended.)

The ACON Controller is supported only in Full Direct Value Mode (parameter 84 = 3). This allows for the most flexibility from an M3-41 perspective, controlling acceleration, velocity, torque, and position dynamically. The factory default mode uses a table stored in the ACON Controllers non-volatile memory. This mode, known as Remote I/O mode is not supported and is the least flexible.

Other parameters that may be of interest are 1/2 and 23/24 for Zone Position 1 and 2 respectively. This controls the ZONE1 and ZONE2 bits in *dins* when the position is within the assigned window. Variables *zone_limit_pos/zone_limit_neg* are used for the dynamic PZONE bit window control.

A properly configured IAI unit will appear online as is shown below within the EtherCAT Explorer, note the RC-ECT-FMOD3 drive type. The FMOD3 represent Full Direct Mode, anything else appearing here means that Parameter 84 is set wrong.


EtherCAT Explorer: 'CTC'
CTC
Module #1, Slot 3, M3-41A ETHERCAT SLAVE ONLINE NODE INFO:
Slave 1 [Axis 1], IAI, Drive, RC-ECT-FMOD3

Manuf	IAI
Group	Drive
Name	RC-ECT-FMOD3
Out size	256 bits (32 bytes)
In size	256 bits (32 bytes)
pstate	ECAT_PROFILE_POS_STARTING1A (19)
inpos	0
fpos	5.000000
tpos	5.000000
perr	0.000000
vel	0.290000
cmode	Profile Position (4)
DRV MODE	Profile Position (1)
PDO STATUS	0x7013
PDO CNTLWORD	0x0010
PDO ACT VEL	0x00000000
PDO ACT TORQ	0x0000000C
PDO ACT ERR	0x00000000
PDO HOME PWRUP	0xFFFFFFFB
PDO ACT POS	0x000001F4
PDO TARG POS	0x000001F4
PDO TARG VEL	0x00003A98
PDO DIG INP	0x00000666
State	8 (OPERATIONAL)
Delay	0[ns]
FMMU	false
Has DC	true (64 bits)
DC Parent Port	0
DC Active	false, Cyc time: 0 ns, Shft: 0
Active Ports	1.0.0.0
Topology	1 (endpoint, parent)
Parent	0
Config address	0x1001 (4097)
Station Alias	0
Axis #	1
Vendor	0x00000588 (1416)
Product Code	0x00a00003 (10485763)
Rev	0x00010004

Name
RC-ECT-FMOD3

All programming units are in mm by default and the '*ppr*' should be set to 100 since the IAI controller is using .01 mm per unit for default positioning, thus 100 * .01 = 1 mm/unit. Normal MSB move commands may be used, specifying position (mm), velocity (mm/s), and acceleration (G's).

```
inposw = .01;      // Positioning Band, .01mm
// Zone boundary can optionally be used
zone_limit_pos = 24.0;
zone_limit_neg = 16.0;  // PZONE bit in dins will go active from 16 to 24 mm
tmax = 70;         // 70% Pressing Current Limit
tlim = 0;          // 0 % Load Current (threshold judgement)
drive enable;

// Let the drive home
[home]
homing_speed1 = 20;     // homing_speed1 is used, 20 mm/sec

cmode = $HOMING_MODE;   // Homing mode for drive
move to 0 using .3,.3;  // Tell the drive to initiate the Home.
wait for in position;
```

```
delay 100 ms;

// IAI only support a full profile position type mode but we can setup the
profile.
cmode = $PROFILE_POSITION_MODE;

[top]
// Move at a velocity of up to 150 mm/sec for 5 mm
// at an acceleration of .3 G and deceleration
// of .3G incrementally
move at 150 for 5 using 0.3,0.3;
wait for in position;
delay 100 ms;
move at 150 for 5 using 0.3,0.3;
wait for in position;
delay 100 ms;
move at 150 for 5 using 0.3,0.3;
wait for in position;
delay 100 ms;
move at 150 for 5 using 0.3,0.3;
wait for in position;
delay 100 ms;
move at 150 for 5 using 0.3,0.3;
wait for in position;
delay 100 ms;
move at 150 for 5 using 0.3,0.3;
wait for in position;
delay 100 ms;

// Move all the way back at a velocity
// of up to 200 mm/sec for -30 mm
// at an acceleration of .3 G and deceleration
// of .3G incrementally.
move at 200 for -30 using 0.3,0.3;
wait for in position;
delay 100 ms;
goto top;         // repeat
```

By default POSITION mode is used. IAI supports a PUSH mode as well and that can be enabled by setting bits b2 and b1 of Control Signal 1 to their proper value, with b1 ON enabling PUSH mode. When using PUSH mode a 'Pressing and a Miss' flag, PSEL b4, must be monitored since this is not considered an alarm condition and is recoverable by the program. When a miss occurs *tpos* will be set equal to *fpos* to allow exiting from the 'wait for in position' instruction. The variables *ztpos* and *zfpos* will contain the values prior to the change should they need to be referenced. Below is a code sample using the PUSH capability:

```
inposw = .01;     // Positioning Band, .01mm
// Zone boundary can optionally be used
zone_limit_pos = 24.0;
zone_limit_neg = 16.0;   // PZONE bit in dins will go active from 16 to 24 mm
tmax = 70;        // 70% Pressing Current Limit
tlim = 0;         // 0 % Load Current (threshold judgement)
drive enable;

// Let the drive home
[home]
```

```
homing_speed1 = 20;     // homing_speed1 is used, 20 mm/sec

cmode = $HOMING_MODE;                 // Homing mode for drive
move to 0 using .3,.3;  // Tell the drive to initiate the Home.
wait for in position;

delay 100 ms;

// IAI only support a full profile position type mode but we can setup the
profile.
cmode = $PROFILE_POSITION_MODE;

// enable push motion with positive direction
IAI_control = 0x00000006;
inposw = 1; // 1 mm push

[top]
// Move at a velocity of up to 150 mm/sec for 5 mm
// at an acceleration of .3 G and deceleration
// of .3G incrementally.  Check for Push/Miss after move.
move at 150 for 5 using 0.3,0.3;
wait for in position;
if (dins & 0x00000008) goto PSEL_Error;
delay 100 ms;
move at 150 for 5 using 0.3,0.3;
wait for in position;
if (dins & 0x00000008) goto PSEL_Error;
delay 100 ms;
move at 150 for 5 using 0.3,0.3;
wait for in position;
if (dins & 0x00000008) goto PSEL_Error;
delay 100 ms;
move at 150 for 5 using 0.3,0.3;
wait for in position;
if (dins & 0x00000008) goto PSEL_Error;
delay 100 ms;
move at 150 for 5 using 0.3,0.3;
wait for in position;
if (dins & 0x00000008) goto PSEL_Error;
delay 100 ms;
move at 150 for 5 using 0.3,0.3;
wait for in position;
if (dins & 0x00000008) goto PSEL_Error;
delay 100 ms;

// Move all the way back at a velocity
// of up to 200 mm/sec for -30 mm
// at an acceleration of .3 G and deceleration
// of .3G incrementally.
move at 200 for -30 using 0.3,0.3;
wait for in position;
// Check for Pressing and Miss error
if (dins & 0x00000008) goto PSEL_Error;
delay 100 ms;
goto top;

[PSEL_Error]
```

```
i = i+1;                    // Press and miss error occurred
delay 1000 ms;
zero following error;
delay 1000 ms;
goto home;
```

Errors and Alarms

The ACON controller will generate cause an MSB fault and EtherCAT log entry under certain error conditions, some are recoverable, and some require power cycling of the IAI Controller. Reference the IAI Controller manual for details. In general it has been found that the Overload error, alarm code 0x00E0 requires power cycling.

The most common error is the Position Command Error, 0x00a3. This error typically occurs when a profile parameter has been exceeded, such as velocity, acceleration, position out of range, or pressing/load current %. Sometimes referencing the Parameter table stored within the controller, via the ROBO Cylinder Software, is helpful in discovering what profile parameter exceeded a limit. For example for the test cylinder used during EtherCAT integration 0.30 G’s was the maximum acceleration/deceleration allowable by the model RCA2-TF4N-I-20-4-30-A3-P actuator. Setting this to 0.31 G’s would cause an ALM condition and 0x00a3 error to be logged.

Blank

Appendix

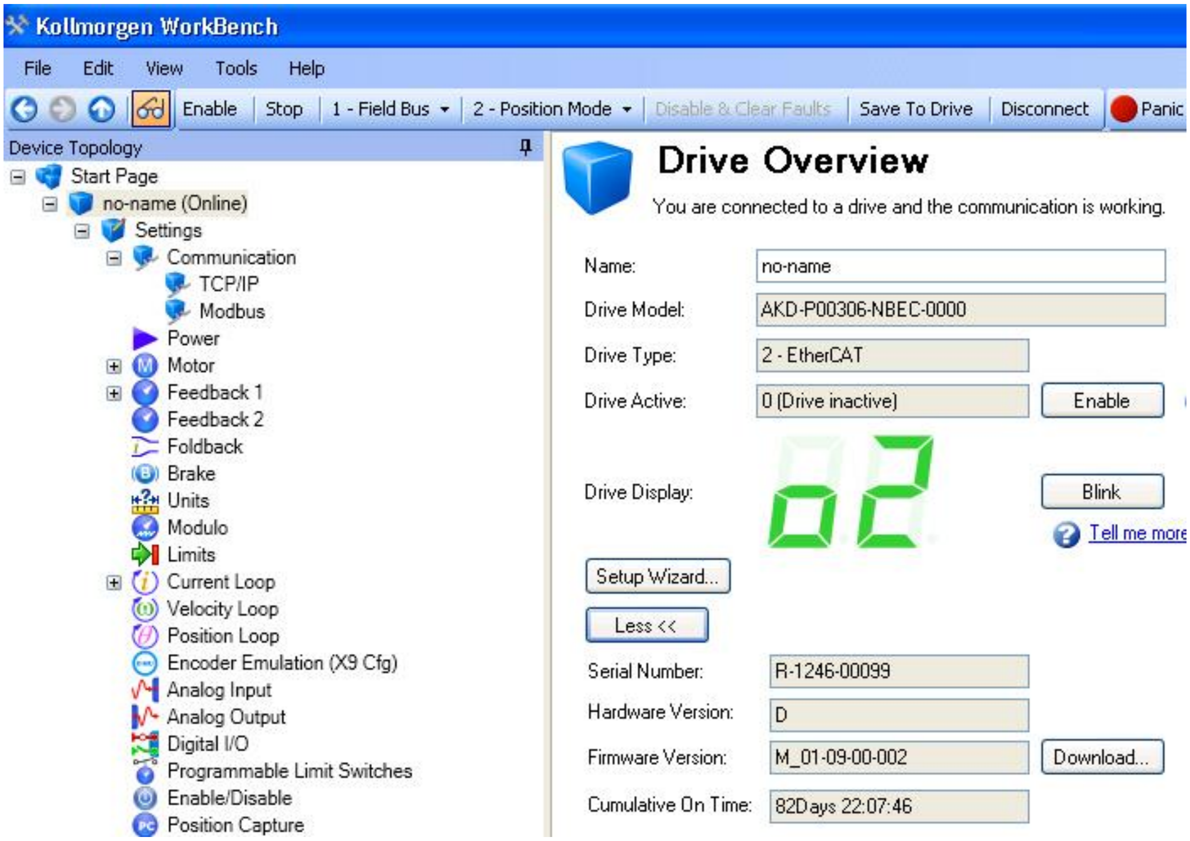
G

[G] Kollmorgen

Kollmorgen manufactures a number of drives. The drive currently supported is the single axis AKD servo drive. This section provides information that may be specific to this manufacturer.

eCAT_driveType – 5

Drive Information & Firmware



Drive Overview

You are connected to a drive and the communication is working.

Name: no-name

Drive Model: AKD-P00306-NBEC-0000

Drive Type: 2 - EtherCAT

Drive Active: 0 (Drive inactive)

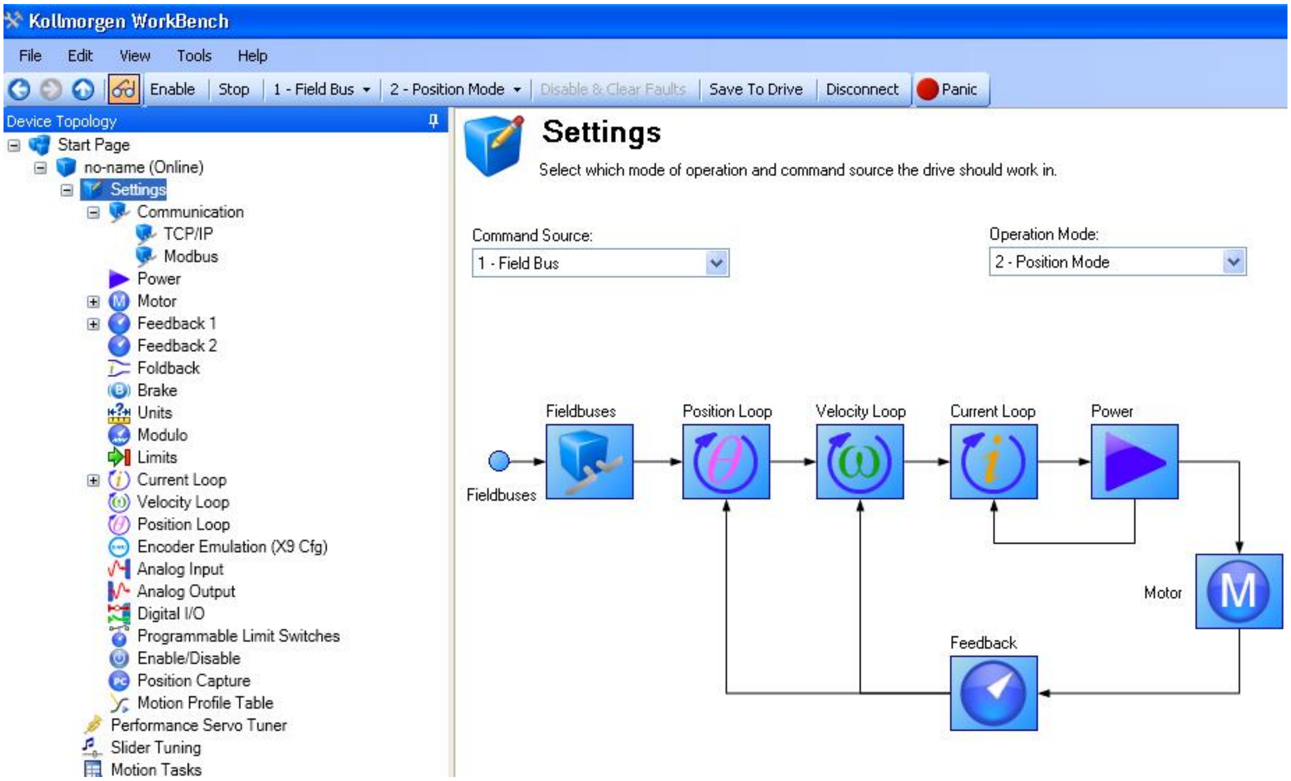
Drive Display: o2

Serial Number: R-1246-00099

Hardware Version: D

Firmware Version: M_01-09-00-002

Cumulative On Time: 82Days 22:07:46



Settings

Select which mode of operation and command source the drive should work in.

Command Source: 1 - Field Bus

Operation Mode: 2 - Position Mode

Positioning Mode

Kollmorgen AKD drives do not support Profile Position mode; Cyclic Sync Position must be used. With this drive, the EtherCAT Master automatically switches to Cyclic Sync Position mode even if Profile Position mode is selected. Interpolated Position mode will lag one scan cycle behind in the move operation from that commanded.

Powerup Delay

Unlike most drives, Kollmorgen can take up to 20 seconds to power up and be recognized on the network. Make sure all drives are powered on and ready before powering up the controller. Otherwise the drive(s) will not be seen. If a configuration file is loaded into the EtherCAT Master, it will retry for the 20-second period prior to aborting, allowing both the controller and the drive to be power cycled at the same time. Combining this with Option Switch #1 (retry override, which continuously scans the network for proper configuration when enabled) will ensure successful power up.

Appendix

H

[H] LinMot

LinMot manufactures a number of linear motor drives. Those currently supported are the C1250-DS-XC-0S and the E1450-DS-QN-0S. This section provides information that may be specific to this manufacturer.

eCAT_driveType – 15

Overview

Only CSP mode is supported with the LinMot series of drives using their CiA402 interface. Homing is done with a series of move commands, executing a 'zero feedback position' to zero out the position. Position capture is not supported, only simple moves. Default units are as follows:

Velocity – mm/s
Acceleration – mm/s^2
Position – millimeters (mm)
ppr – 10000 (.0001 mm per position count)
inposw – typically .01 mm, E1450 may require a greater value depending upon load and tuning.

For proper operation you must verify the firmware revision levels installed in the drive using their LinMot-Talk PC software. During testing a number of anomalies were discovered that required firmware updates that at the time were not the default shipped with their LinMot-Talk software. The tested revisions are as follows:

C1250:
Firmware Release – 6.4 Build 20151105
OS – OSSW_C1200_V6S4_b01
MC – MCSW1200_S21_V6S4_b01
INTF – IntfSWEC_SG6_V6S5_a01 (contains fixes supplied by LinMot, .HX3 file)

E1450:
Firmware Release – 6.4 Build 20151105
OS – OSSW_E1400V2_V6S4_b01
MC – MCSW1400_S32_V6S4_b01
INTF – IntfSWEC_SG6_V6S5_a01 (contains fixes supplied by LinMot, .HX3 file)

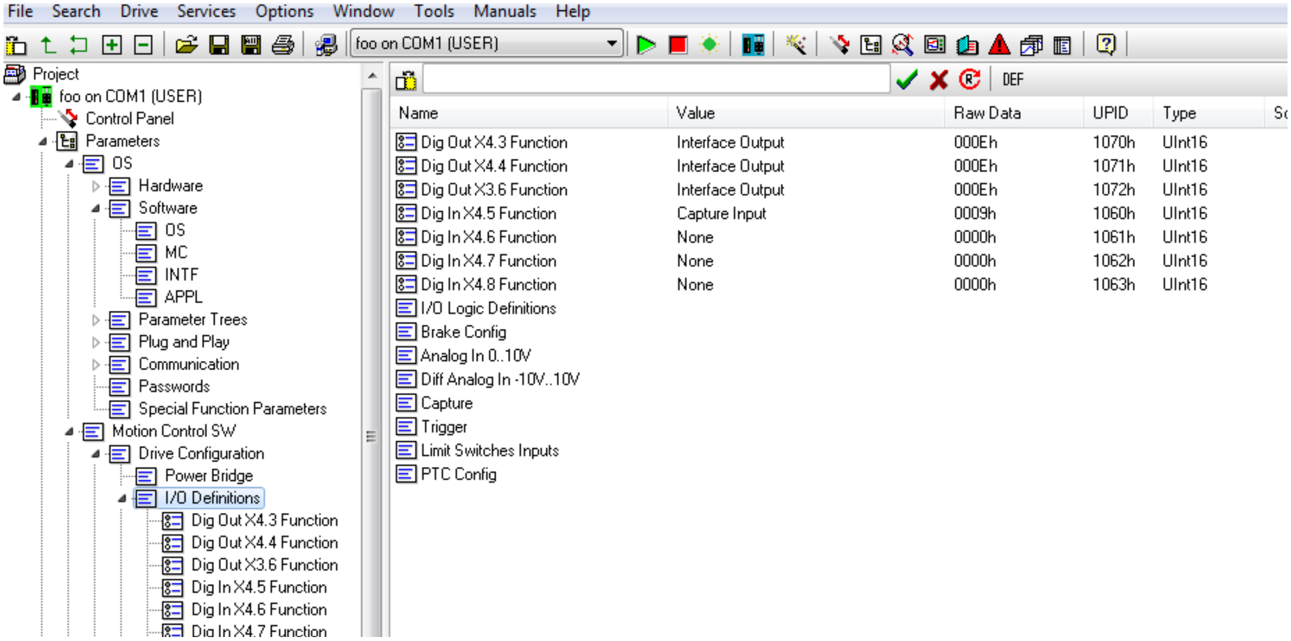
EtherCAT Connectors:

C1250: X17 IN, X18 Out

E1450: X17 IN, X18 Out

I/O

For I/O to be functional its interface must first be enabled using LinMot-Talk:

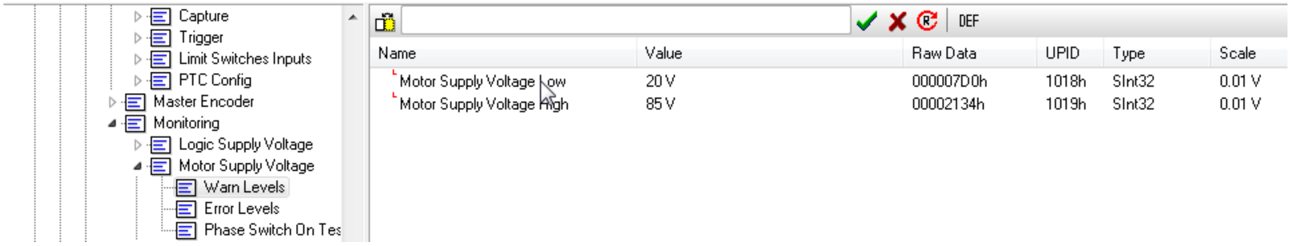


Name	Value	Raw Data	UPID	Type	Sc
Dig Out X4.3 Function	Interface Output	000Eh	1070h	UInt16	
Dig Out X4.4 Function	Interface Output	000Eh	1071h	UInt16	
Dig Out X3.6 Function	Interface Output	000Eh	1072h	UInt16	
Dig In X4.5 Function	Capture Input	0009h	1060h	UInt16	
Dig In X4.6 Function	None	0000h	1061h	UInt16	
Dig In X4.7 Function	None	0000h	1062h	UInt16	
Dig In X4.8 Function	None	0000h	1063h	UInt16	
I/O Logic Definitions					
Brake Config					
Analog In 0..10V					
Diff Analog In -10V..10V					
Capture					
Trigger					
Limit Switches Inputs					
PTC Config					

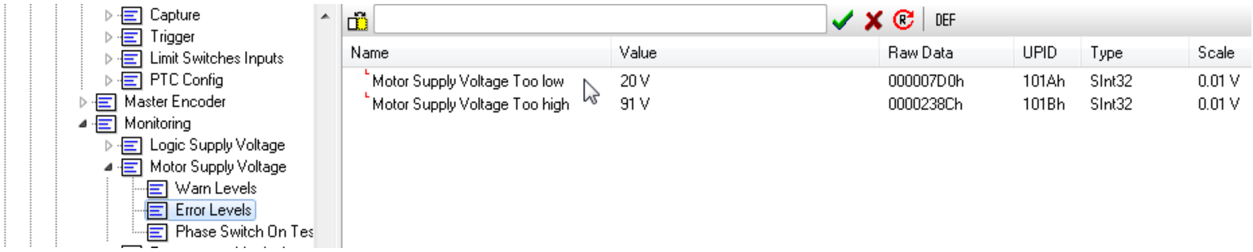
IO maps direction to the X4 interface where X4.3 is bit 3 of the output and X4.5 is bit 5 of the input, as referenced by LinMot-Talk.

Motor Voltage Levels

If low motor voltage is being used, for example 24V on a higher voltage motor, the warning and error detection levels must be adjusted accordingly:



Name	Value	Raw Data	UPID	Type	Scale
Motor Supply Voltage Low	20 V	000007D0h	1018h	SInt32	0.01 V
Motor Supply Voltage High	85 V	00002134h	1019h	SInt32	0.01 V

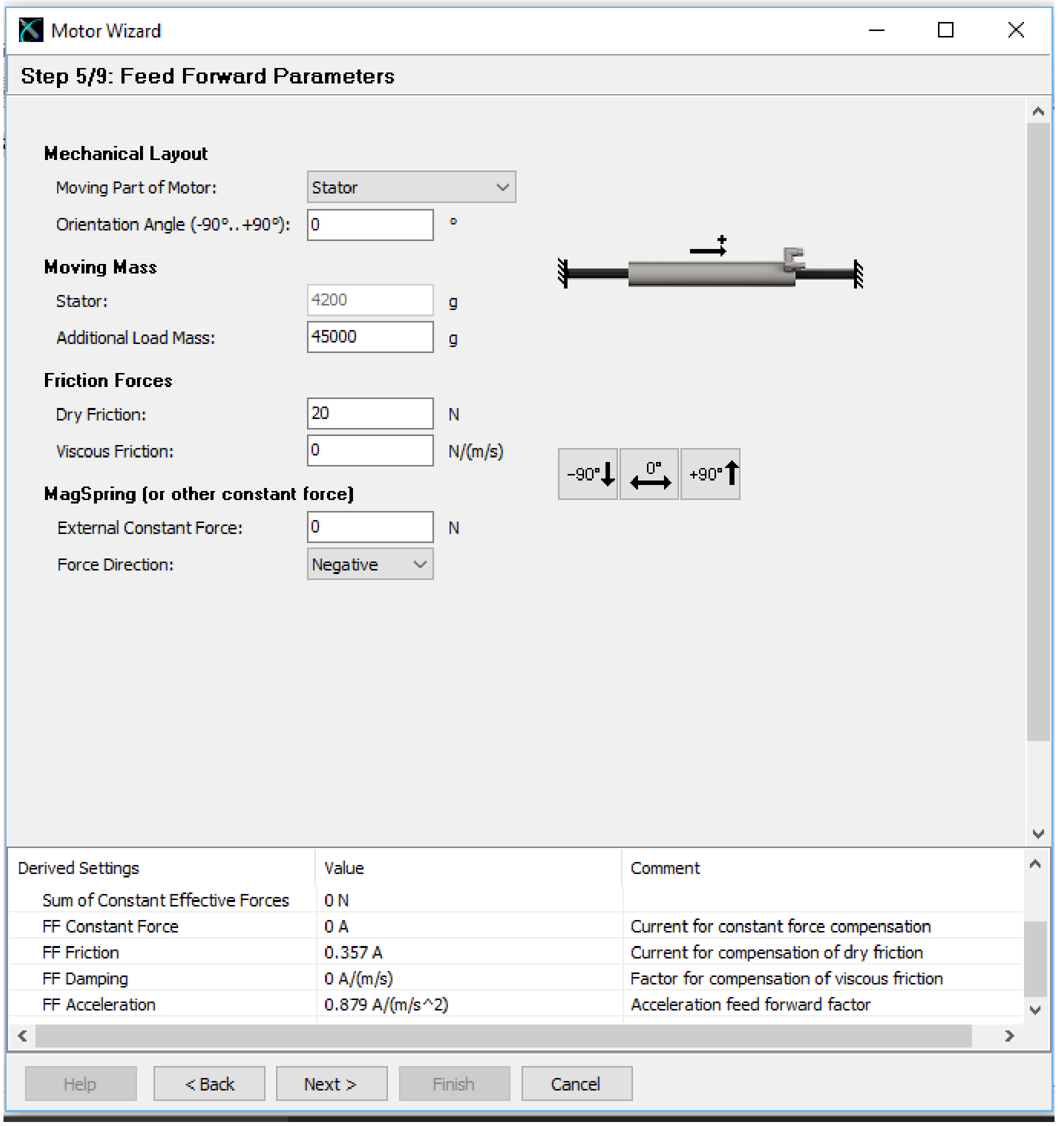


Capture
Trigger
Limit Switches Inputs
PTC Config
Master Encoder
Monitoring
Logic Supply Voltage
Motor Supply Voltage
Warn Levels
Error Levels
Phase Switch On Tes

Name	Value	Raw Data	UPID	Type	Scale
Motor Supply Voltage Too low	20 V	000007D0h	101Ah	SInt32	0.01 V
Motor Supply Voltage Too high	91 V	0000238Ch	101Bh	SInt32	0.01 V

Motor Wizard

When using the Motor Wizard it is recommended that the “Additional Load Mass” and “Dry Friction” be set to 0. Failure to do so can cause the LinMot drive not to track closely with the EtherCAT commanded position.



Motor Wizard

Step 5/9: Feed Forward Parameters

Mechanical Layout

Moving Part of Motor: Stator

Orientation Angle (-90°..+90°): 0 °

Moving Mass

Stator: 4200 g

Additional Load Mass: 45000 g

Friction Forces

Dry Friction: 20 N

Viscous Friction: 0 N/(m/s)

MagSpring (or other constant force)

External Constant Force: 0 N

Force Direction: Negative

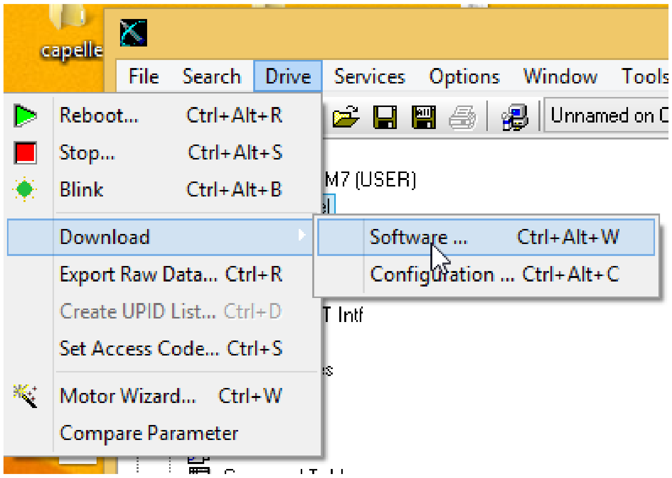
Derived Settings	Value	Comment
Sum of Constant Effective Forces	0 N	
FF Constant Force	0 A	Current for constant force compensation
FF Friction	0.357 A	Current for compensation of dry friction
FF Damping	0 A/(m/s)	Factor for compensation of viscous friction
FF Acceleration	0.879 A/(m/s^2)	Acceleration feed forward factor

Control Parameter A/B

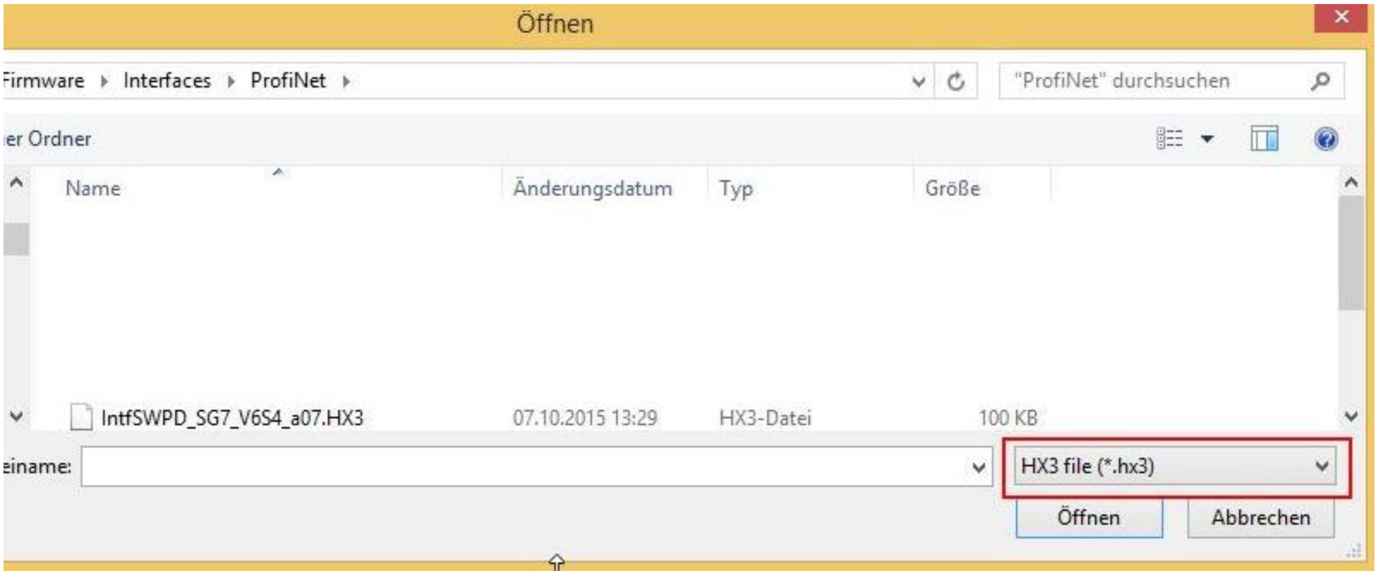
LinMot provides two sets of tuning parameters. By default Control Parameter A is used. The variable 'tmax' is used to set the Maximum Current, UPID 0x13A6 & 0x13BA. You may select Control Parameter B by writing a 1 to the msb variable 'IAI_control'.

INTF file upgrade (.HX3)

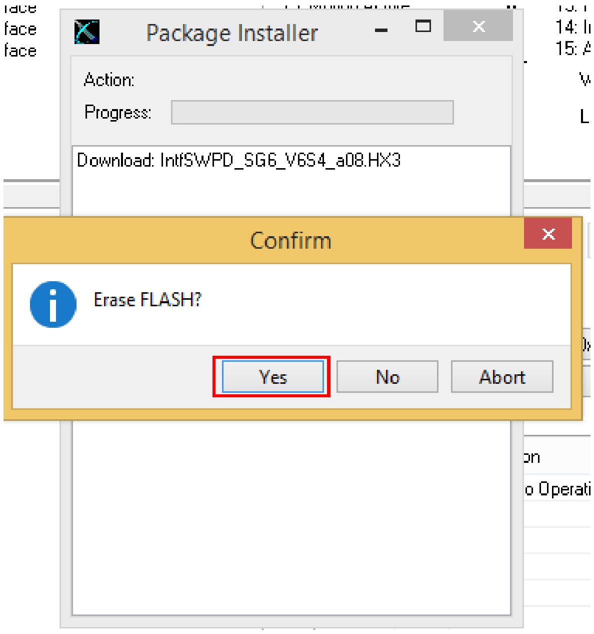
Log into the individual drive with LinMot talk then go to the Drive Toolbar and select download/Software.



A dialogue box will open, and in this window you will need to locate the file named "IntfSWEC_SG6_V6S5_a01.HX3". Make sure to select HX3 files from the dropdown menu outlined below:



The next popup will prompt you to erase the FLASH program, and you will select Yes.



Download: IntfSWPD_SG6_V6S4_a08.HX3

Confirm

Erase FLASH?

After installation is complete cycle power on the LinMot Drive.

Test MSB

```
// This is a background MSB.  Make sure inposw is set for the drive, typically
// .001 for 1048576 ppr.  Also set the ppr and mppr.  This program will set
// the ppr and mppr to a value commonly used, it may have to be changed based
// on the user setup.

// Enable the drive, turning power on to the amplifier.  The current position
// will be constantly updated so the drive does not move.

// Activate DC Sync0 each cycle time.
// dcsync <slave node or -1 for current>,
//      <Sync0 Cycle Time in nanoseconds, ns>,
//      <Sync1 shift from Sync0, ns>,
//      <Sync0 shift from Cycle Time, ns>,
//      <Sync start delay in ns>
// Set all parameters to 0 except the slave node to deactivate.
// Below is a 1ms Sync0 cycle time with no Sync1.
// Sync0 starts at cycle time and is not shifted and there is a
// 100ms delay before it all starts the first sequence.

// Set dec, acc, and max velocity.
dec = 100;
acc = 100;
vmax = 50;
driveenable = 0;  // disable output control since LINMOT does not need it

drive_type = eCAT_driveType;
// Adjust the ppr and mppr based on the drive/encoder we have installed.
// This overrides that of the property sheet.
if eCAT_driveType == 2 goto Copley;
if eCAT_driveType == 3 goto Yaskawa;
if eCAT_driveType == 4 goto Elmo;
if eCAT_driveType == 5 goto Kollmorgen;
if eCAT_driveType == 6 goto Sanyo_Denki;
if eCAT_driveType == 7 goto Emerson;
if eCAT_driveType == 8 goto AMC;
if eCAT_driveType == 9 goto Virtual;
if eCAT_driveType == 12 goto ABB;
if eCAT_driveType == 15 goto LinMot;
goto AMC;
// by default assume 1048576
[Yaskawa]
[Kollmorgen]
mppr = 1048576;
ppr = 1048576;
goto beginTest;

[ABB]
mppr = 524288;
ppr = 524288;
goto beginTest;

[Virtual]
mppr = 65536;
ppr = 65536;
// After each control loop fposc = tposc so it appears as though the axis moved during
// a move command
//set simulated feedback on;
goto beginTest;

[Copley]
[Elmo]
mppr = 8000;
ppr = 8000;
goto beginTest;

[Emerson]
mppr = 65536;
ppr = 65536;
goto beginTest;

[Sanyo_Denki]
mppr = 131072;
ppr = 131072;
goto beginTest;

[AMC]
mppr = 12000;
ppr = 12000;
goto beginTest;

[LinMot]
mppr = 10000;
ppr = 10000;
inposw = .1;    // This can be tightened up after tuning, C1250 about .01, E1450 was not tuned so .1
goto beginTest;

[beginTest]
cmode = $CYCLIC_SYNC_POSITION_MODE;
//inposw=.001;

// Initialize distributed clocks
delay 3000 ms;
dcsync -1, 1000000, 0, 0, 100000000;
delay 200 ms;  // starts 100 milliseconds into the future
drive enable;
delay 1000 ms;
zero feedback position;  // assume current position is home and zero our position
[run]
// Begin the move, 20 mm/second for 30mm
move at 20 for 30;
wait for in position;
//setout 1,2,3,4,5,6,7,8;
// Delay 100ms once in position
delay 100 ms;
// Do a relative move back 30mm  at 20 mm/second
move at 20 for -30;
wait for in position;
// Delay 100ms second once in position
```

```
//clrout 1,2,3,4,5,6,7,8;
delay 100 ms;
// Do it again, forever...

goto run;
```

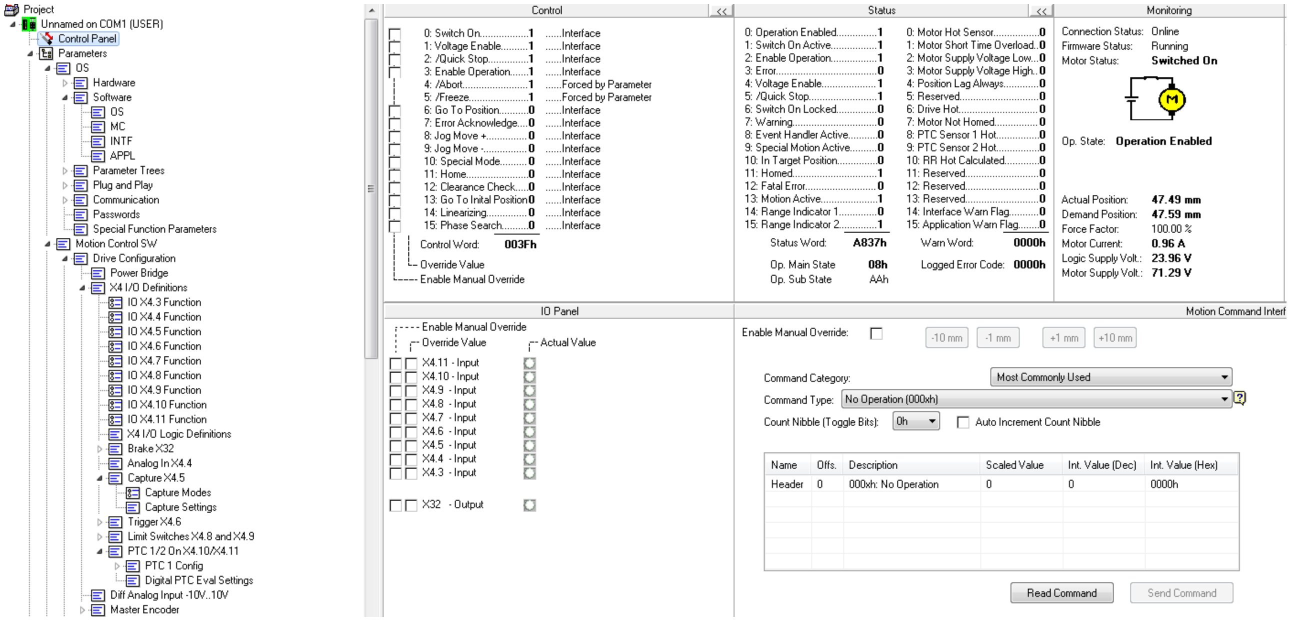
Homing MSB

When homing based on torque a position error will build up. This error is generated by the LinMot drive as the slider is stalled. Proper homing consists of moving in the direction of home while monitoring torque. Once the desired torque level is reached 'stop' the motor and then move in the opposite direction the amount of 'perr'. Since the error was accumulated by the drive while it was stalled the move of the amount of 'perr' simply syncs the position reported by the drive with that given by the program as the target postion, removing the error. Once at that position you can clear the feedback position and begin normal moves.

```
//**********************************HomeRoutine********************************
[HomeRoutine]
// A setting to define which direction we're traveling in to home the axis.
Direction = -1;

// Move the axis based on commands fed from main program for homing.
// We move some very large amount based on direction until torque is large.
move at Maxspeed to (1000000 * Direction) using Accel,Decel;
delay 10 ms;

/*** Scan for excessive torque indicating we hit something
*     ...hopefully it's the end-stop
*/
[HomeMoveLoop]
// Check for user to end the move early.
if command == 0 goto StopMove;
// Check for the end-stop using accumulated torque (rmstrq).
// This value may have to be adjusted based upon the application.
if rmstrq >= .9 goto HomeStopAndZero;
if rmstrq <= -.9 goto HomeStopAndZero;
goto HomeMoveLoop; // loop until something happens

[HomeStopAndZero]
stoprate = Decel;// Set stopping decel rate
delay 10 ms;
//stop the axis because we've reached our defined torque level from above.
stop;
delay 500 ms;

// We've theoretically gone beyond the end-stop causing following-error
// to accumulated during the decel transition.
// Move back to the point where we began accumuating following-error along
// with an additional short 1mm move to us get off the end-stop.
// Direction is defined previously.
move at Maxspeed for ((perr * Direction) + (1 * -Direction));
wait for in position;
delay 500 ms;
zero feedback position; //Clear the feedback position
delay 10 ms;
goto MoveToOffsetPosition;  // All set we are home, do any move needed...
```

LinMot-Talk Control Panel

The LinMot-Talk Control Panel can be used during drive operation to monitor operation by attaching an RS232 port from the PC to X19, Config RS232 port. You must then use their Login function prior to communications.



Project
- Unnamed on COM1 (USER)
  - Control Panel
  - Parameters
    - OS
      - Hardware
      - Software
        - OS
        - MC
        - INTF
        - APPL
      - Parameter Trees
      - Plug and Play
      - Communication
      - Passwords
      - Special Function Parameters
    - Motion Control SW
      - Drive Configuration
        - Power Bridge
        - X4 I/O Definitions
          - IO X4.3 Function
          - IO X4.4 Function
          - IO X4.5 Function
          - IO X4.6 Function
          - IO X4.7 Function
          - IO X4.8 Function
          - IO X4.9 Function
          - IO X4.10 Function
          - IO X4.11 Function
          - X4 I/O Logic Definitions
          - Brake X32
          - Analog In X4.4
          - Capture X4.5
            - Capture Modes
            - Capture Settings
          - Trigger X4.6
          - Limit Switches X4.8 and X4.9
          - PTC 1/2 On X4.10/X4.11
            - PTC 1 Config
            - Digital PTC Eval Settings
        - Diff Analog Input -10V..10V
        - Master Encoder

Control

	Bit	Value	Source
□	0: Switch On	1	Interface
□	1: Voltage Enable	1	Interface
□	2: /Quick Stop	1	Interface
□	3: Enable Operation	1	Interface
	4: /Abort	1	Forced by Parameter
	5: /Freeze	1	Forced by Parameter
□	6: Go To Position	0	Interface
□	7: Error Acknowledge	0	Interface
□	8: Jog Move +	0	Interface
□	9: Jog Move -	0	Interface
□	10: Special Mode	0	Interface
□	11: Home	0	Interface
□	12: Clearance Check	0	Interface
□	13: Go To Inital Position	0	Interface
□	14: Linearizing	0	Interface
□	15: Phase Search	0	Interface

Control Word: **003Fh**

Override Value

Enable Manual Override

Status

Bit	Value	Bit	Value
0: Operation Enabled	1	0: Motor Hot Sensor	0
1: Switch On Active	1	1: Motor Short Time Overload	0
2: Enable Operation	1	2: Motor Supply Voltage Low	0
3: Error	0	3: Motor Supply Voltage High	0
4: Voltage Enable	1	4: Position Lag Always	0
5: /Quick Stop	1	5: Reserved	0
6: Switch On Locked	0	6: Drive Hot	0
7: Warning	0	7: Motor Not Homed	0
8: Event Handler Active	0	8: PTC Sensor 1 Hot	0
9: Special Motion Active	0	9: PTC Sensor 2 Hot	0
10: In Target Position	0	10: RR Hot Calculated	0
11: Homed	1	11: Reserved	0
12: Fatal Error	0	12: Reserved	0
13: Motion Active	1	13: Reserved	0
14: Range Indicator 1	0	14: Interface Warn Flag	0
15: Range Indicator 2	1	15: Application Warn Flag	0
Status Word:	**A837h**	Warn Word:	**0000h**
Op. Main State	**08h**	Logged Error Code:	**0000h**
Op. Sub State	AAh		

Monitoring

Connection Status: Online
Firmware Status: Running
Motor Status: **Switched On**

Op. State: **Operation Enabled**

Actual Position: **47.49 mm**
Demand Position: **47.59 mm**
Force Factor: 100.00 %
Motor Current: **0.96 A**
Logic Supply Volt.: **23.96 V**
Motor Supply Volt.: **71.29 V**

IO Panel

Enable Manual Override
Override Value
Actual Value

Enable Manual Override	Override Value		Actual Value
□	□	X4.11 - Input	○
□	□	X4.10 - Input	○
□	□	X4.9 - Input	○
□	□	X4.8 - Input	○
□	□	X4.7 - Input	○
□	□	X4.6 - Input	○
□	□	X4.5 - Input	○
□	□	X4.4 - Input	○
□	□	X4.3 - Input	○
□	□	X32 - Output	○

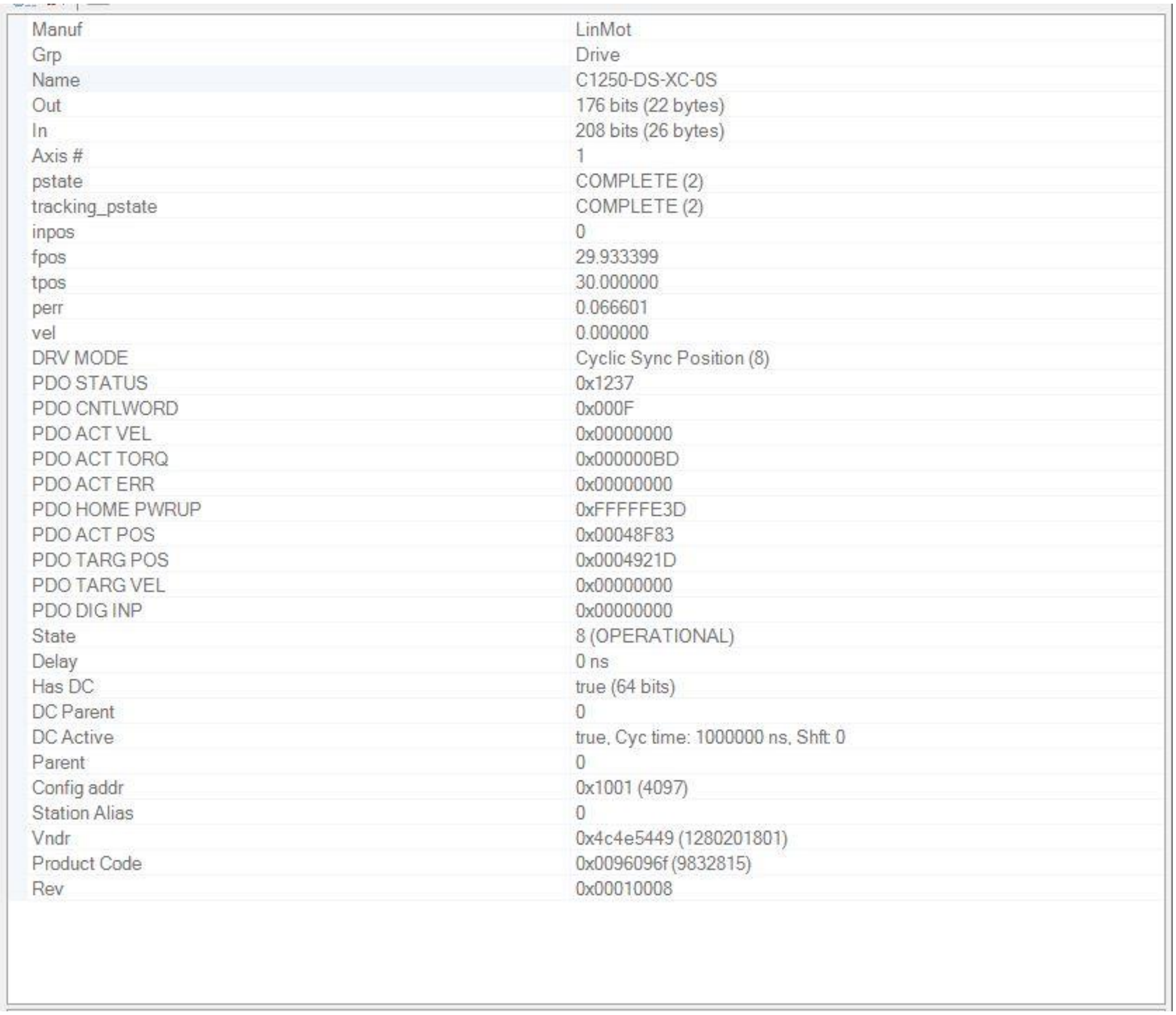
Motion Command Interf

Enable Manual Override: □

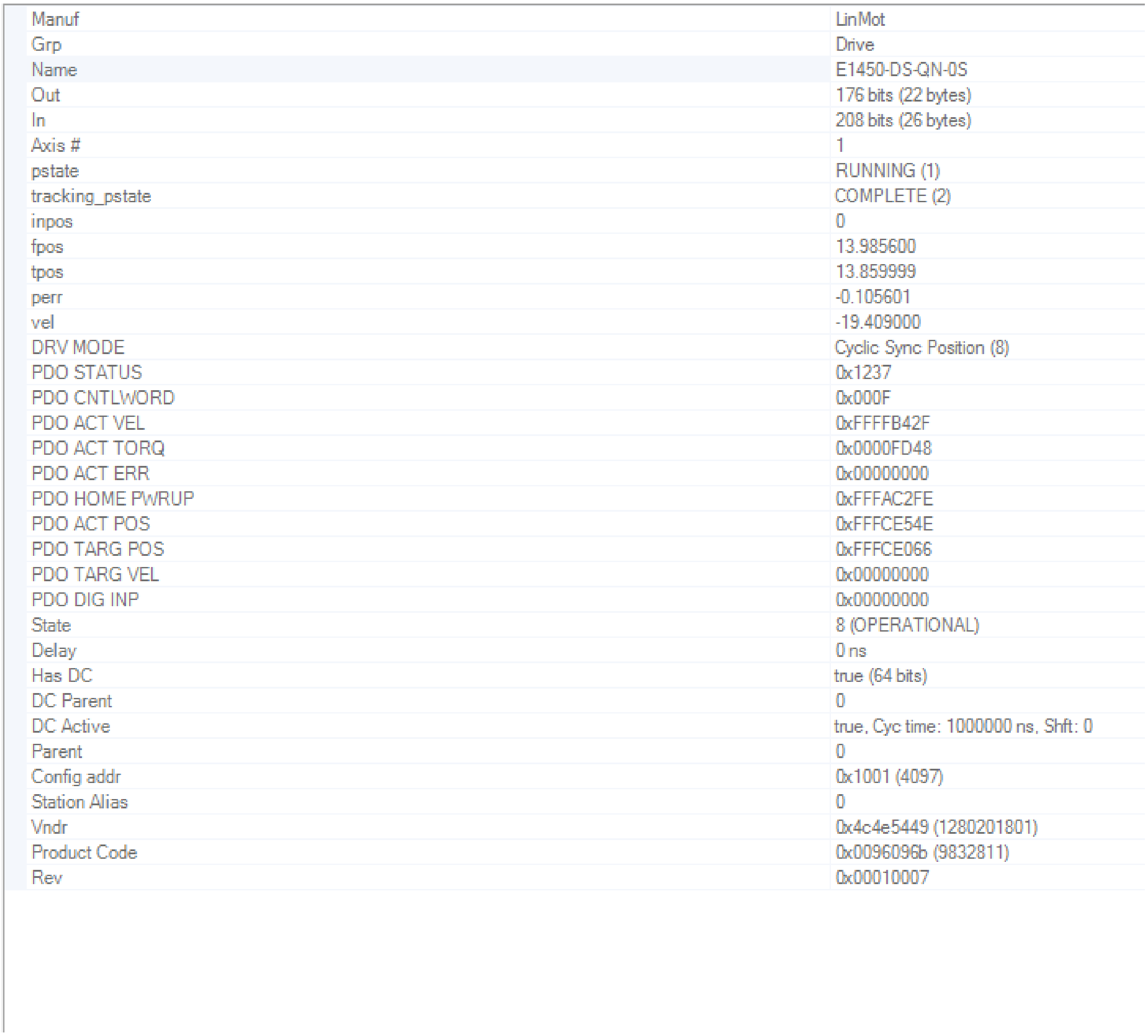
Command Category: Most Commonly Used
Command Type: No Operation (000xh)
Count Nibble (Toggle Bits): 0h
□ Auto Increment Count Nibble

Name	Offs.	Description	Scaled Value	Int. Value (Dec)	Int. Value (Hex)
Header	0	000xh: No Operation	0	0	0000h

EtherCAT Explorer View



Manuf	LinMot
Grp	Drive
Name	C1250-DS-XC-0S
Out	176 bits (22 bytes)
In	208 bits (26 bytes)
Axis #	1
pstate	COMPLETE (2)
tracking_pstate	COMPLETE (2)
inpos	0
fpos	29.933399
tpos	30.000000
perr	0.066601
vel	0.000000
DRV MODE	Cyclic Sync Position (8)
PDO STATUS	0x1237
PDO CNTLWORD	0x000F
PDO ACT VEL	0x00000000
PDO ACT TORQ	0x000000BD
PDO ACT ERR	0x00000000
PDO HOME PWRUP	0xFFFFFE3D
PDO ACT POS	0x00048F83
PDO TARG POS	0x0004921D
PDO TARG VEL	0x00000000
PDO DIG INP	0x00000000
State	8 (OPERATIONAL)
Delay	0 ns
Has DC	true (64 bits)
DC Parent	0
DC Active	true, Cyc time: 1000000 ns, Shft: 0
Parent	0
Config addr	0x1001 (4097)
Station Alias	0
Vndr	0x4c4e5449 (1280201801)
Product Code	0x0096096f (9832815)
Rev	0x00010008



Manuf	LinMot
Grp	Drive
Name	E1450-DS-QN-0S
Out	176 bits (22 bytes)
In	208 bits (26 bytes)
Axis #	1
pstate	RUNNING (1)
tracking_pstate	COMPLETE (2)
inpos	0
fpos	13.985600
tpos	13.859999
perr	-0.105601
vel	-19.409000
DRV MODE	Cyclic Sync Position (8)
PDO STATUS	0x1237
PDO CNTLWORD	0x000F
PDO ACT VEL	0xFFFFB42F
PDO ACT TORQ	0x0000FD48
PDO ACT ERR	0x00000000
PDO HOME PWRUP	0xFFFAC2FE
PDO ACT POS	0xFFFCE54E
PDO TARG POS	0xFFFCE066
PDO TARG VEL	0x00000000
PDO DIG INP	0x00000000
State	8 (OPERATIONAL)
Delay	0 ns
Has DC	true (64 bits)
DC Parent	0
DC Active	true, Cyc time: 1000000 ns, Shft: 0
Parent	0
Config addr	0x1001 (4097)
Station Alias	0
Vndr	0x4c4e5449 (1280201801)
Product Code	0x0096096b (9832811)
Rev	0x00010007

Appendix I

[I] Mitsubishi

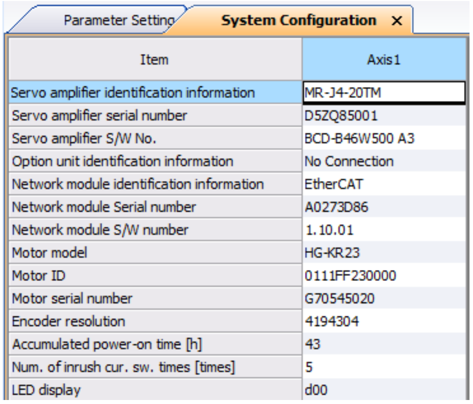
Mitsubishi manufactures a number of drives. That currently supported by the M3-41A is the J4. For test purposes two drives in particular were tested: MR-J4-20TM-ECT and the MR-J4-40TM1-ETP. This section provides information that may be specific to this manufacturer.

eCAT_driveType – 13

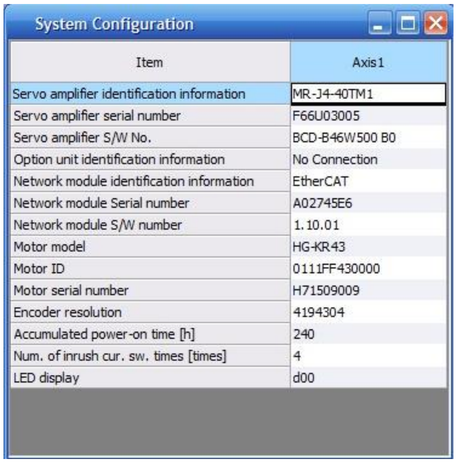
Overview

Only CSP and Homing mode is supported with the Mitsubishi drives. Additionally the DC SYNC motion control instruction is ignored since it is automatically enabled prior to network operation. Thus it is automatically enabled by the EtherCAT Master with the SYNC0 always having an offset from the network cycle time, typically 1 mS cycle time, with a shift of ‘cycle time/4’ or 250uS for 1 mS, whichever is smaller. SYNC1 is not supported by the drive. Additionally, early versions of Mitsubishi firmware had jitter in their clock when used as the master slave device. If a synchronization problem occurs it is recommended to replace the first slave device with a different device, such as Wago, or Omron switch, otherwise contact Mitsubishi to ensure you have the latest firmware.

For proper operation you must verify the firmware revision levels installed in the drive using their MR Configurator2 PC software. The ‘Control mode selection’ of the drive must also be set to ‘Cyclic synchronous mode’. Below is information on the drives used for testing, note the servo amplifier software revision of ‘BCD-B46W500 A3’, this firmware contains clock jitter. The later revision ‘B46W500 B0’ has corrected the problem:



Item	Axis1
Servo amplifier identification information	MR-J4-20TM
Servo amplifier serial number	D5ZQ85001
Servo amplifier S/W No.	BCD-B46W500 A3
Option unit identification information	No Connection
Network module identification information	EtherCAT
Network module Serial number	A0273D86
Network module S/W number	1.10.01
Motor model	HG-KR23
Motor ID	0111FF230000
Motor serial number	G70545020
Encoder resolution	4194304
Accumulated power-on time [h]	43
Num. of inrush cur. sw. times [times]	5
LED display	d00

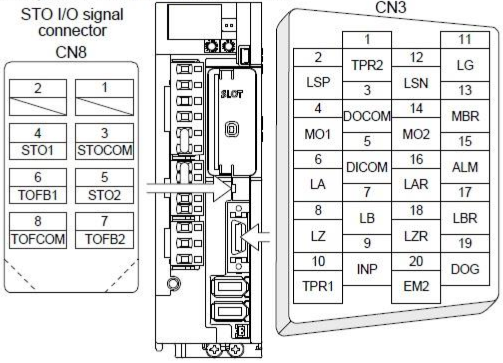


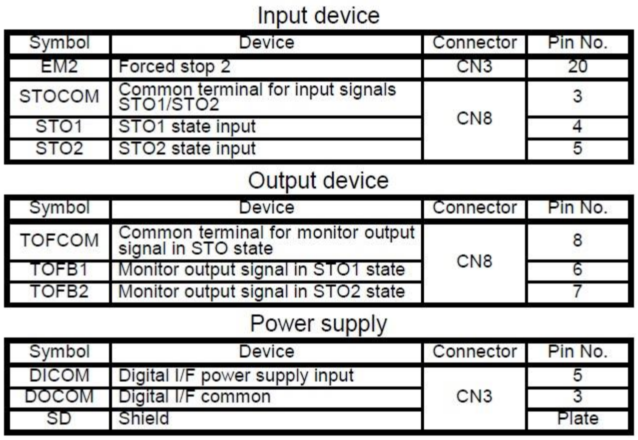
Item	Axis1
Servo amplifier identification information	MR-J4-40TM1
Servo amplifier serial number	F66U03005
Servo amplifier S/W No.	BCD-B46W500 B0
Option unit identification information	No Connection
Network module identification information	EtherCAT
Network module Serial number	A02745E6
Network module S/W number	1.10.01
Motor model	HG-KR43
Motor ID	0111FF430000
Motor serial number	H71509009
Encoder resolution	4194304
Accumulated power-on time [h]	240
Num. of inrush cur. sw. times [times]	4
LED display	d00

CN3 Connector

For test purposes there are three signals that must be enabled either internally using the MR Configurator2 PC based software or pins jumpered on the CN3 Connector. This consists of LSP (Limit Positive), LSN (Limit Negative), EM2 (Enable), and DICOM. DICOM must be connected to +24V, the other signals to 24 Volt Return or proper controlling switch.

The following shows MR-J4-10TM signals as a typical example. For other servo amplifiers, refer to each servo amplifier instruction manual.




Input device

Symbol	Device	Connector	Pin No.
EM2	Forced stop 2	CN3	20
STOCOM	Common terminal for input signals STO1/STO2	CN8	3
STO1	STO1 state input		4
STO2	STO2 state input		5

Output device

Symbol	Device	Connector	Pin No.
TOFCOM	Common terminal for monitor output signal in STO state	CN8	8
TOFB1	Monitor output signal in STO1 state		6
TOFB2	Monitor output signal in STO2 state		7

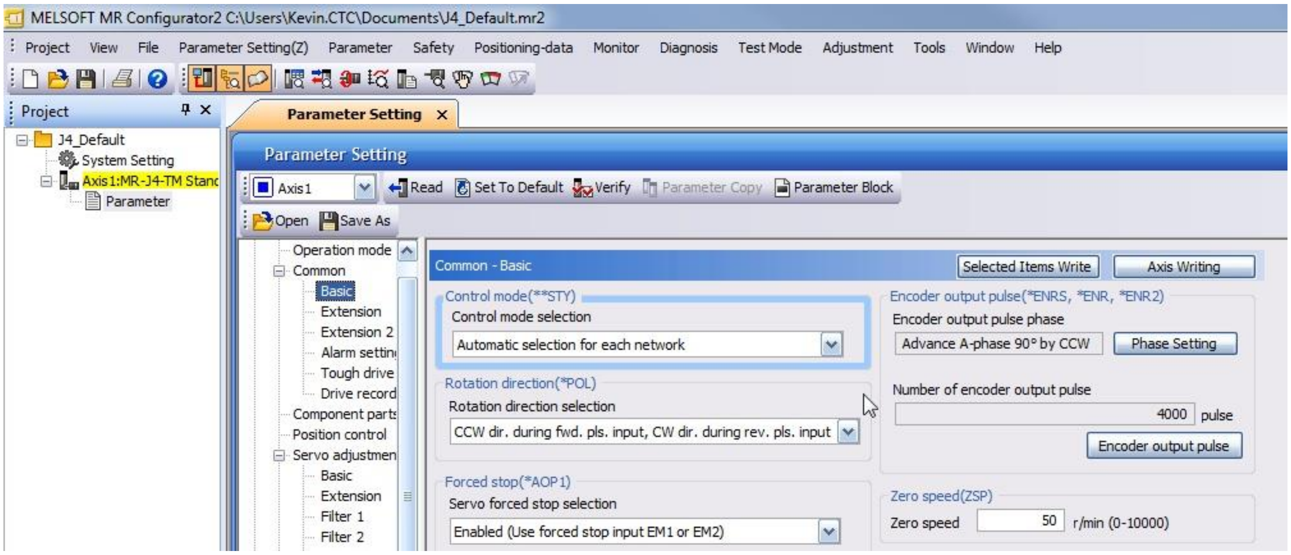
Power supply

Symbol	Device	Connector	Pin No.
DICOM	Digital I/F power supply input	CN3	5
DOCOM	Digital I/F common		3
SD	Shield		Plate

MR Configurator2

Connect CN5 to a USB port on your PC and then connect to the drive using the Mitsubishi MR Configurator2 (MR2) software. For the purposes of this manual version V1.51D of MR2 was used.

Make sure 'Automatic selection for each network' is selected.



Control mode(**STY)
Control mode selection: Automatic selection for each network

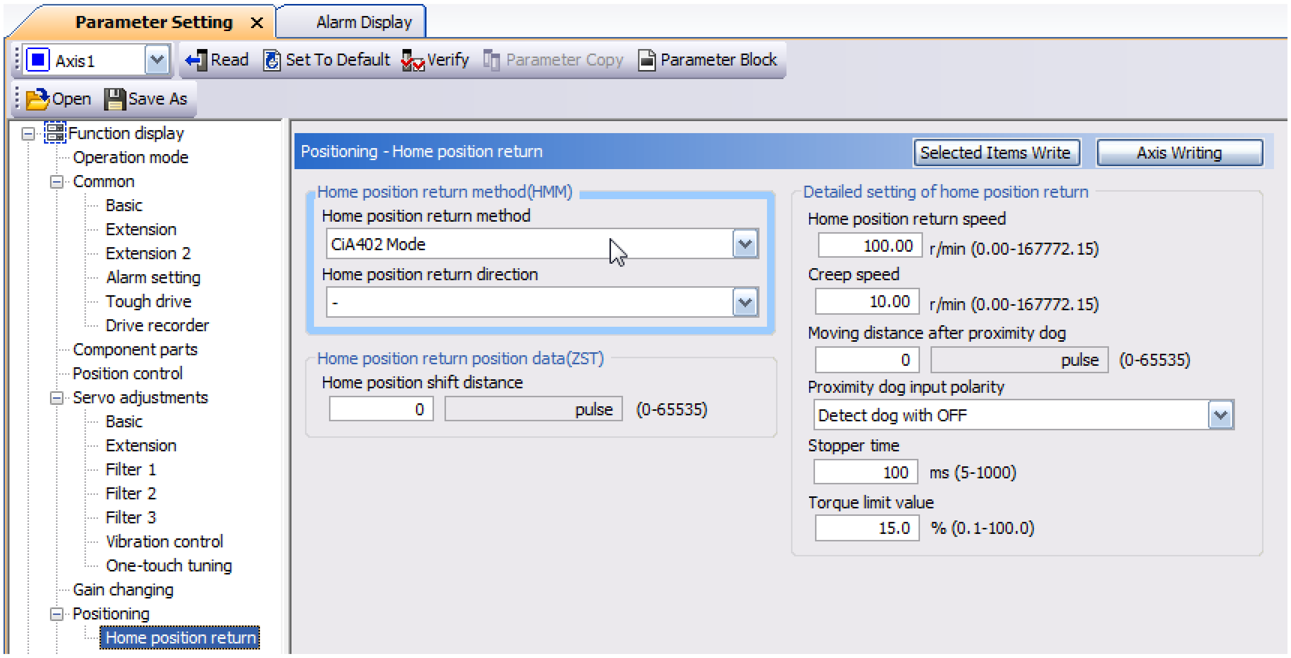
Rotation direction(*POL)
Rotation direction selection: CCW dir. during fwd. pls. input, CW dir. during rev. pls. input

Forced stop(*AOP1)
Servo forced stop selection: Enabled (Use forced stop input EM1 or EM2)

Encoder output pulse(*ENRS, *ENR, *ENR2)
Encoder output pulse phase: Advance A-phase 90° by CCW
Number of encoder output pulse: 4000 pulse

Zero speed(ZSP)
Zero speed: 50 r/min (0-10000)

Homing mode by the drive using Cia402 standard is supported. You may also use CSP mode and zero the feedback when homing manually using MSB instructions. The drive should still be configured as below.

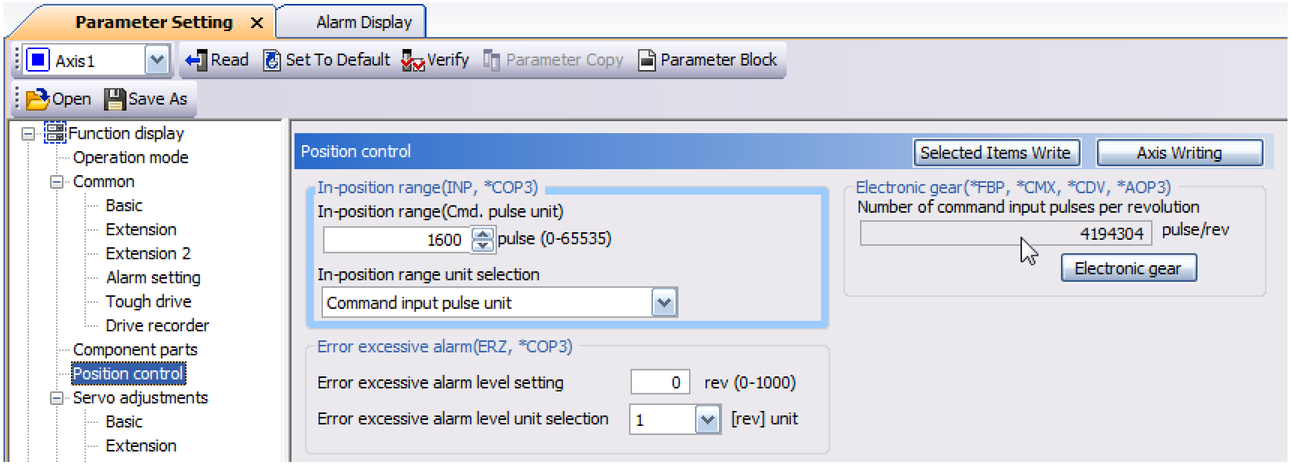


Home position return method(HMM)
Home position return method: CiA402 Mode
Home position return direction: -

Home position return position data(ZST)
Home position shift distance: 0 pulse (0-65535)

Detailed setting of home position return
Home position return speed: 100.00 r/min (0.00-167772.15)
Creep speed: 10.00 r/min (0.00-167772.15)
Moving distance after proximity dog: 0 pulse (0-65535)
Proximity dog input polarity: Detect dog with OFF
Stopper time: 100 ms (5-1000)
Torque limit value: 15.0 % (0.1-100.0)

Check your ppr for MSB programming, this must be set in the properties section of the drive when using QuickBuilder. Below shows a ppr of 419430 pulse/rev. Also look at the "Error excessive alarm level setting". This typically needs to be increased to prevent 35.1 and 52.03 alarm errors. The more the drive lags the commanded position the larger the value would need to be (2 to 3 revs would not be unreasonable).



Position control

In-position range(INP, *COP3)

In-position range(Cmd. pulse unit) 1600 pulse (0-65535)

In-position range unit selection: Command input pulse unit

Error excessive alarm(ERZ, *COP3)

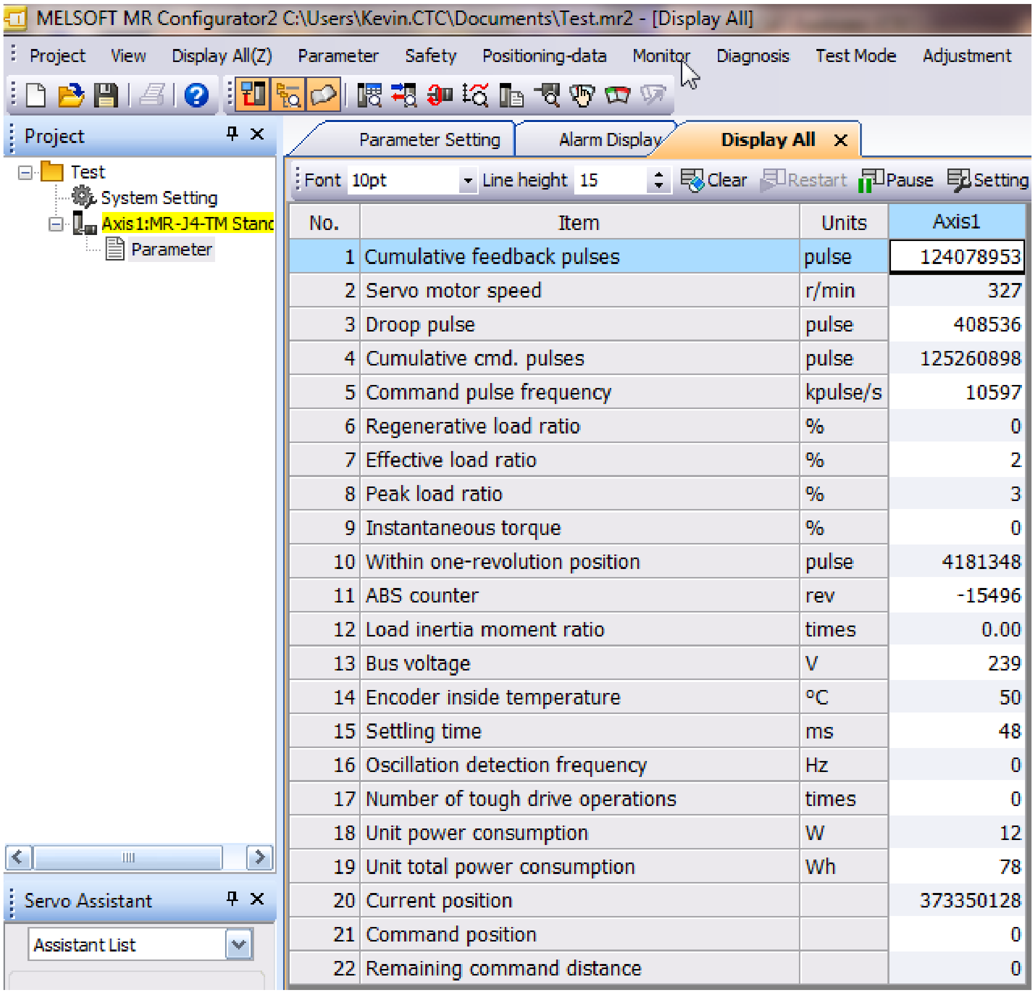
Error excessive alarm level setting 0 rev (0-1000)

Error excessive alarm level unit selection 1 [rev] unit

Electronic gear(*FBP, *CMX, *CDV, *AOP3)

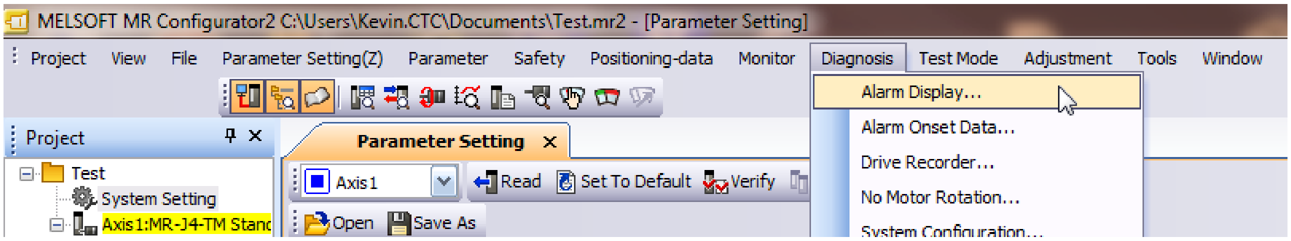
Number of command input pulses per revolution 4194304 pulse/rev

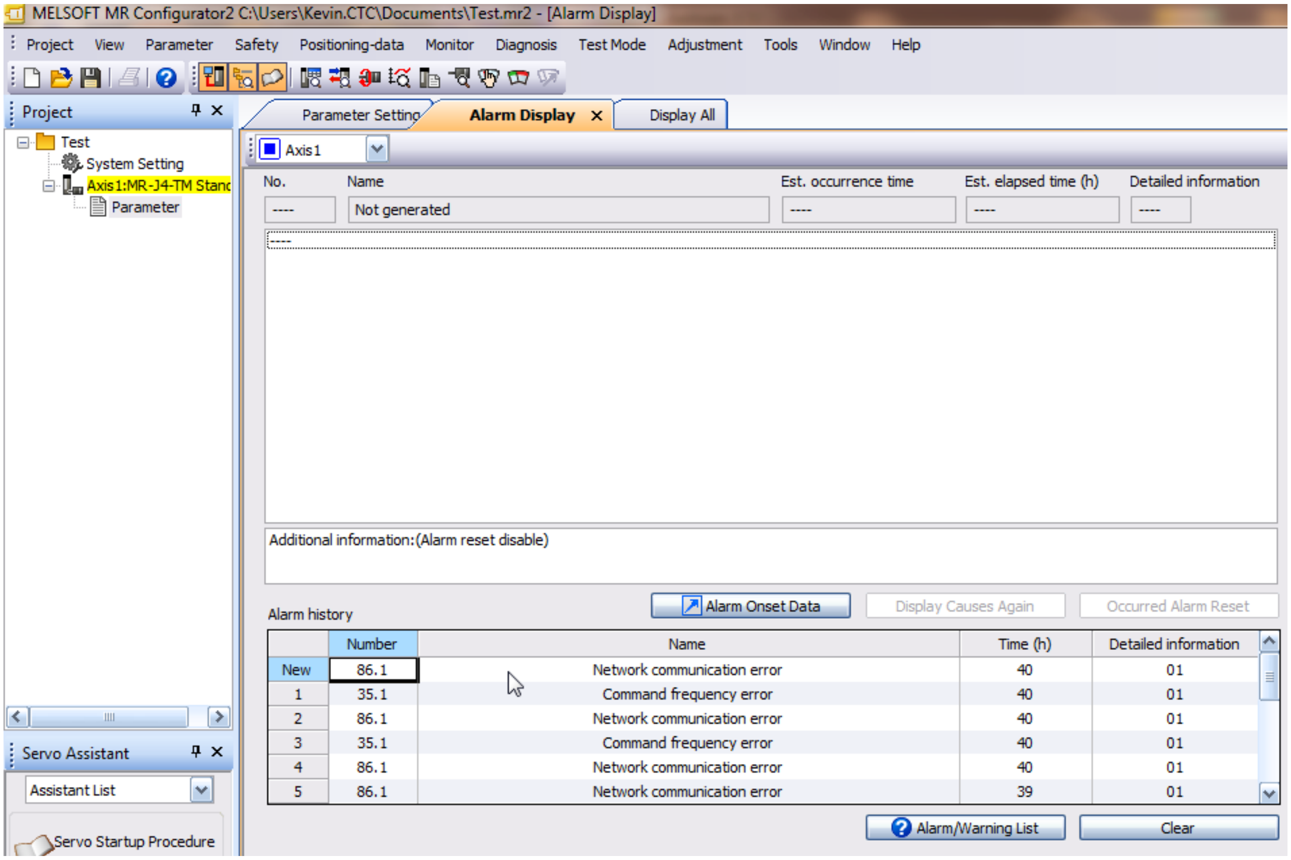
You may monitor operation using the Monitor->Display All selection:



No.	Item	Units	Axis1
1	Cumulative feedback pulses	pulse	124078953
2	Servo motor speed	r/min	327
3	Droop pulse	pulse	408536
4	Cumulative cmd. pulses	pulse	125260898
5	Command pulse frequency	kpulse/s	10597
6	Regenerative load ratio	%	0
7	Effective load ratio	%	2
8	Peak load ratio	%	3
9	Instantaneous torque	%	0
10	Within one-revolution position	pulse	4181348
11	ABS counter	rev	-15496
12	Load inertia moment ratio	times	0.00
13	Bus voltage	V	239
14	Encoder inside temperature	°C	50
15	Settling time	ms	48
16	Oscillation detection frequency	Hz	0
17	Number of tough drive operations	times	0
18	Unit power consumption	W	12
19	Unit total power consumption	Wh	78
20	Current position		373350128
21	Command position		0
22	Remaining command distance		0

Alarm display is also useful. Typically an 86.1 will appear, this is normal as the EtherCAT network is restarted and is automatically cleared by the EtherCAT Master during initialization.





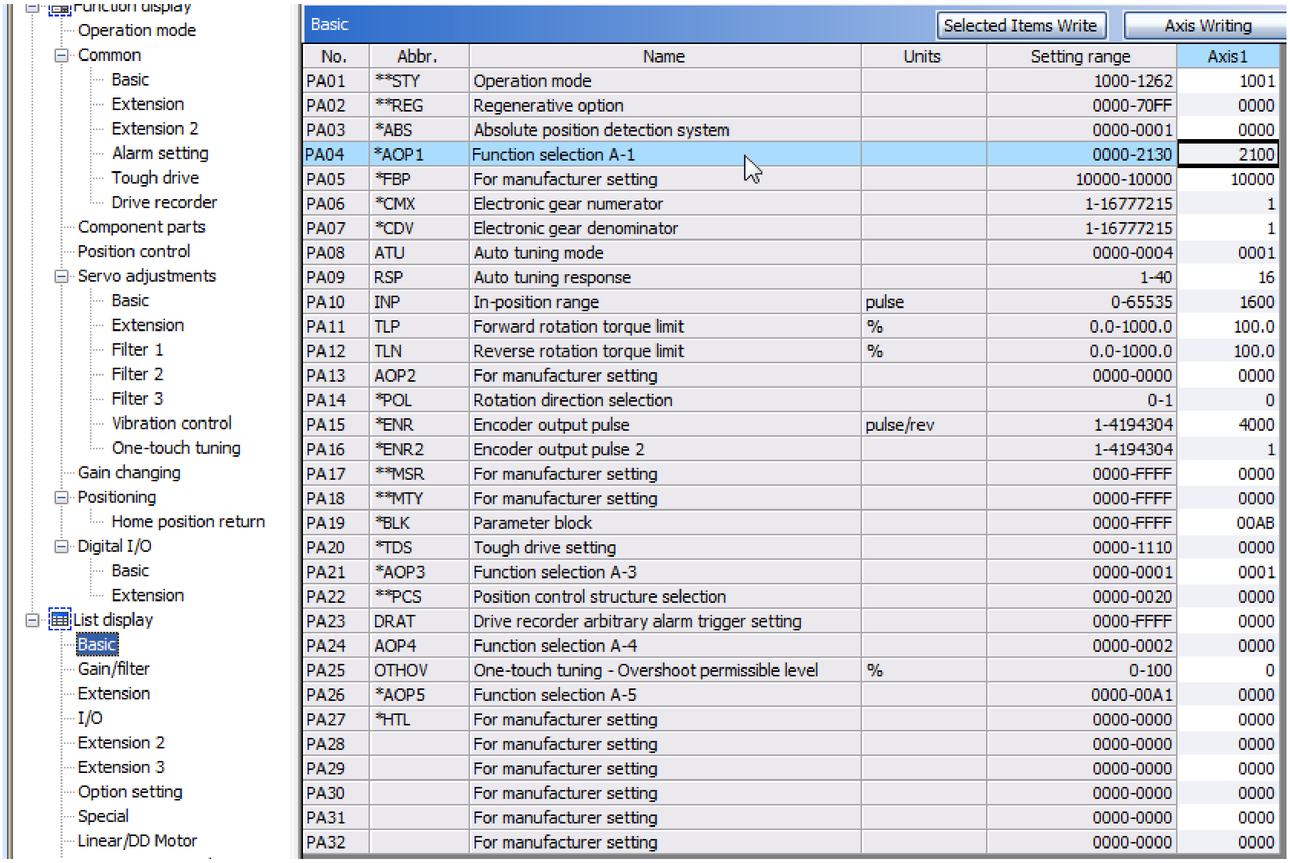
No.	Name	Est. occurrence time	Est. elapsed time (h)	Detailed information
----	Not generated	----	----	----

Additional information:(Alarm reset disable)

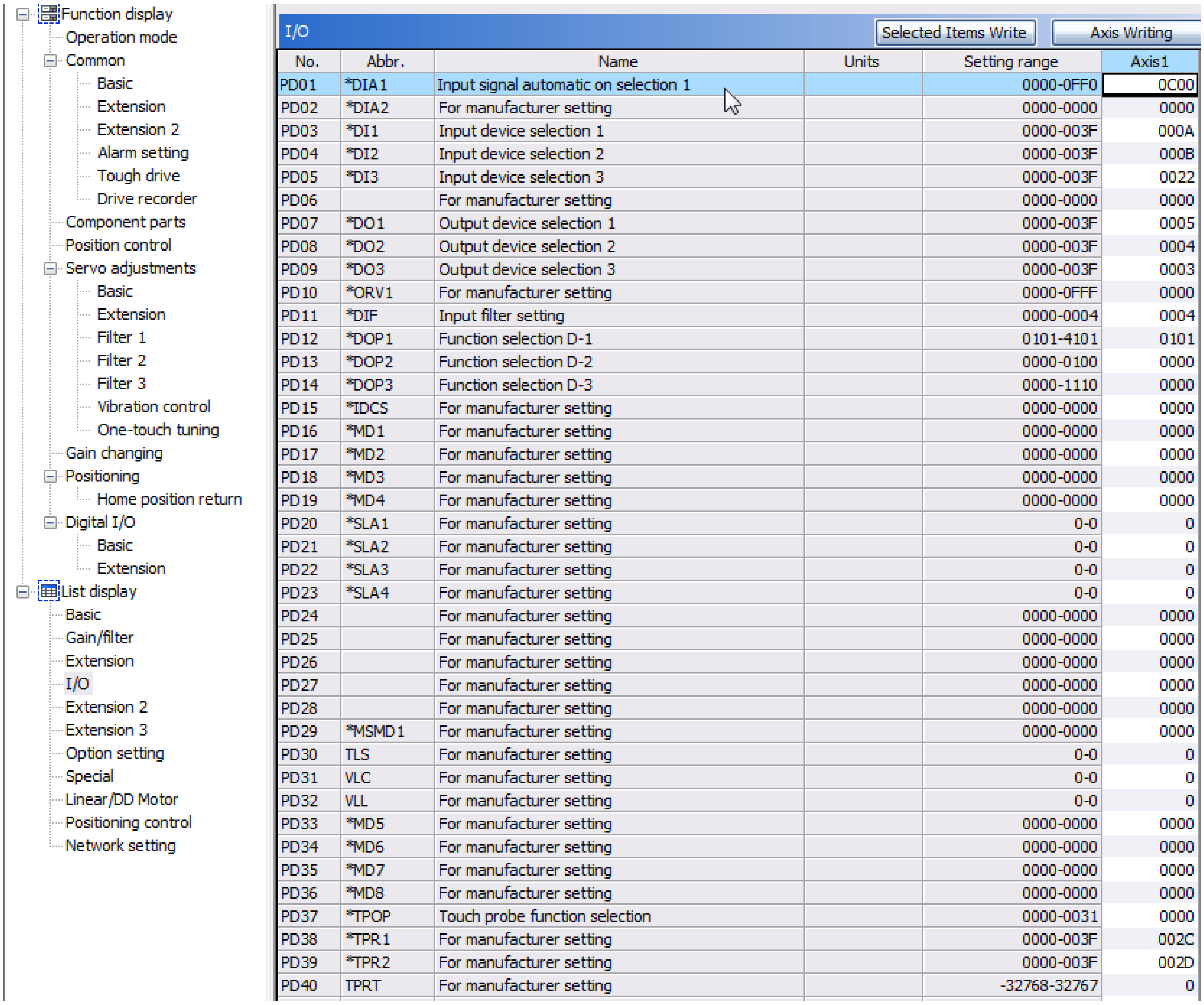
Alarm history

	Number	Name	Time (h)	Detailed information
New	86.1	Network communication error	40	01
1	35.1	Command frequency error	40	01
2	86.1	Network communication error	40	01
3	35.1	Command frequency error	40	01
4	86.1	Network communication error	40	01
5	86.1	Network communication error	39	01

If you need to override the CN3 inputs, servo parameter PA04 can be set to 2100H to disable forced stop input and PD01 to 0C00 to disable the external limit switches.



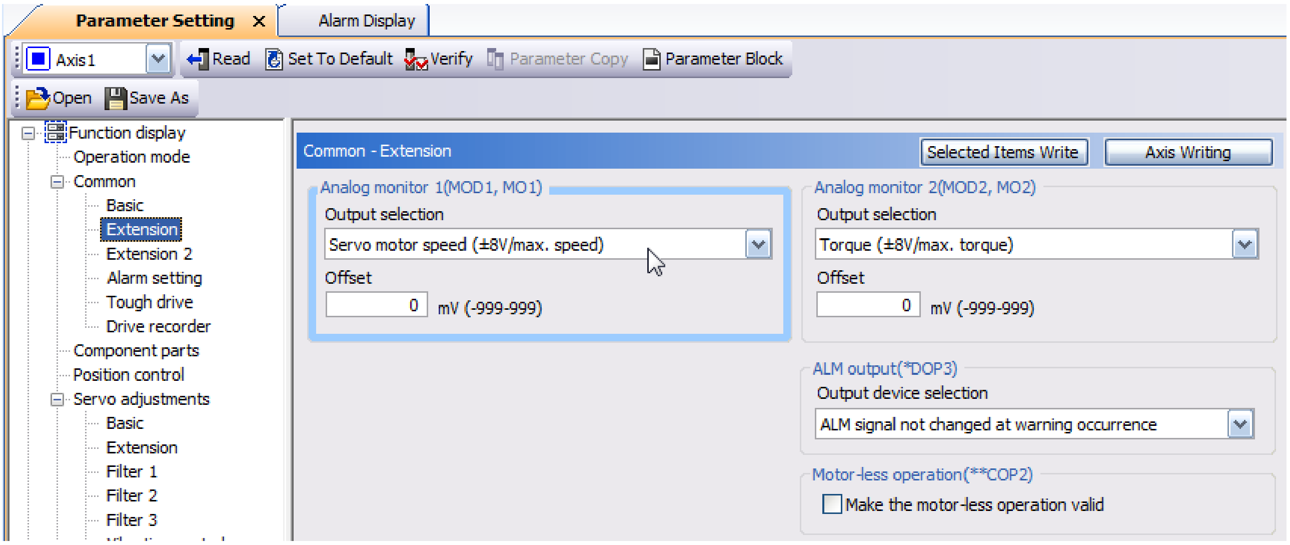
No.	Abbr.	Name	Units	Setting range	Axis1
PA01	**STY	Operation mode		1000-1262	1001
PA02	**REG	Regenerative option		0000-70FF	0000
PA03	*ABS	Absolute position detection system		0000-0001	0000
PA04	*AOP1	Function selection A-1		0000-2130	2100
PA05	*FBP	For manufacturer setting		10000-10000	10000
PA06	*CMX	Electronic gear numerator		1-16777215	1
PA07	*CDV	Electronic gear denominator		1-16777215	1
PA08	ATU	Auto tuning mode		0000-0004	0001
PA09	RSP	Auto tuning response		1-40	16
PA10	INP	In-position range	pulse	0-65535	1600
PA11	TLP	Forward rotation torque limit	%	0.0-1000.0	100.0
PA12	TLN	Reverse rotation torque limit	%	0.0-1000.0	100.0
PA13	AOP2	For manufacturer setting		0000-0000	0000
PA14	*POL	Rotation direction selection		0-1	0
PA15	*ENR	Encoder output pulse	pulse/rev	1-4194304	4000
PA16	*ENR2	Encoder output pulse 2		1-4194304	1
PA17	**MSR	For manufacturer setting		0000-FFFF	0000
PA18	**MTY	For manufacturer setting		0000-FFFF	0000
PA19	*BLK	Parameter block		0000-FFFF	00AB
PA20	*TDS	Tough drive setting		0000-1110	0000
PA21	*AOP3	Function selection A-3		0000-0001	0001
PA22	**PCS	Position control structure selection		0000-0020	0000
PA23	DRAT	Drive recorder arbitrary alarm trigger setting		0000-FFFF	0000
PA24	AOP4	Function selection A-4		0000-0002	0000
PA25	OTHOV	One-touch tuning - Overshoot permissible level	%	0-100	0
PA26	*AOP5	Function selection A-5		0000-00A1	0000
PA27	*HTL	For manufacturer setting		0000-0000	0000
PA28		For manufacturer setting		0000-0000	0000
PA29		For manufacturer setting		0000-0000	0000
PA30		For manufacturer setting		0000-0000	0000
PA31		For manufacturer setting		0000-0000	0000
PA32		For manufacturer setting		0000-0000	0000



No.	Abbr.	Name	Units	Setting range	Axis1
PD01	*DIA1	Input signal automatic on selection 1		0000-0FF0	0C00
PD02	*DIA2	For manufacturer setting		0000-0000	0000
PD03	*DI1	Input device selection 1		0000-003F	000A
PD04	*DI2	Input device selection 2		0000-003F	000B
PD05	*DI3	Input device selection 3		0000-003F	0022
PD06		For manufacturer setting		0000-0000	0000
PD07	*DO1	Output device selection 1		0000-003F	0005
PD08	*DO2	Output device selection 2		0000-003F	0004
PD09	*DO3	Output device selection 3		0000-003F	0003
PD10	*ORV1	For manufacturer setting		0000-0FFF	0000
PD11	*DIF	Input filter setting		0000-0004	0004
PD12	*DOP1	Function selection D-1		0101-4101	0101
PD13	*DOP2	Function selection D-2		0000-0100	0000
PD14	*DOP3	Function selection D-3		0000-1110	0000
PD15	*IDCS	For manufacturer setting		0000-0000	0000
PD16	*MD1	For manufacturer setting		0000-0000	0000
PD17	*MD2	For manufacturer setting		0000-0000	0000
PD18	*MD3	For manufacturer setting		0000-0000	0000
PD19	*MD4	For manufacturer setting		0000-0000	0000
PD20	*SLA1	For manufacturer setting		0-0	0
PD21	*SLA2	For manufacturer setting		0-0	0
PD22	*SLA3	For manufacturer setting		0-0	0
PD23	*SLA4	For manufacturer setting		0-0	0
PD24		For manufacturer setting		0000-0000	0000
PD25		For manufacturer setting		0000-0000	0000
PD26		For manufacturer setting		0000-0000	0000
PD27		For manufacturer setting		0000-0000	0000
PD28		For manufacturer setting		0000-0000	0000
PD29	*MSMD1	For manufacturer setting		0000-0000	0000
PD30	TLS	For manufacturer setting		0-0	0
PD31	VLC	For manufacturer setting		0-0	0
PD32	VLL	For manufacturer setting		0-0	0
PD33	*MD5	For manufacturer setting		0000-0000	0000
PD34	*MD6	For manufacturer setting		0000-0000	0000
PD35	*MD7	For manufacturer setting		0000-0000	0000
PD36	*MD8	For manufacturer setting		0000-0000	0000
PD37	*TPOP	Touch probe function selection		0000-0031	0000
PD38	*TPR1	For manufacturer setting		0000-003F	002C
PD39	*TPR2	For manufacturer setting		0000-003F	002D
PD40	TPRT	For manufacturer setting		-32768-32767	0

MR Configurator2 Miscellaneous

Other screens of interest and their settings during testing:



Common - Extension

Analog monitor 1(MOD1, MO1)

Output selection: Servo motor speed (±8V/max. speed)

Offset: 0 mV (-999-999)

Analog monitor 2(MOD2, MO2)

Output selection: Torque (±8V/max. torque)

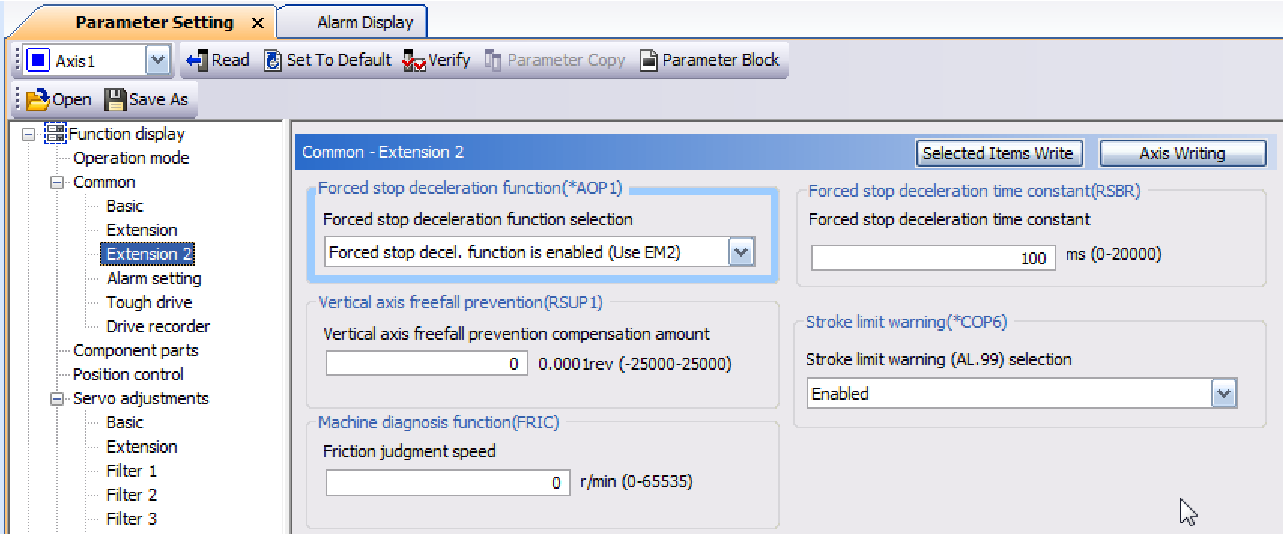
Offset: 0 mV (-999-999)

ALM output(*DOP3)

Output device selection: ALM signal not changed at warning occurrence

Motor-less operation(**COP2)

□ Make the motor-less operation valid



Common - Extension 2

Forced stop deceleration function(*AOP1)

Forced stop deceleration function selection: Forced stop decel. function is enabled (Use EM2)

Forced stop deceleration time constant(RSBR)

Forced stop deceleration time constant: 100 ms (0-20000)

Vertical axis freefall prevention(RSUP1)

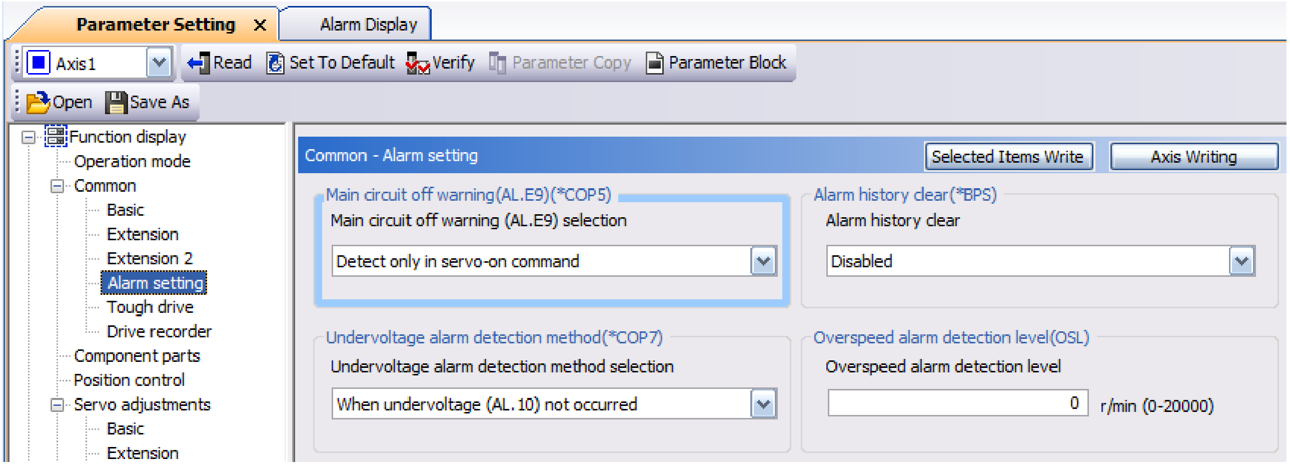
Vertical axis freefall prevention compensation amount: 0 0.0001rev (-25000-25000)

Stroke limit warning(*COP6)

Stroke limit warning (AL.99) selection: Enabled

Machine diagnosis function(FRIC)

Friction judgment speed: 0 r/min (0-65535)



Common - Alarm setting

Main circuit off warning(AL.E9)(*COP5)

Main circuit off warning (AL.E9) selection: Detect only in servo-on command

Alarm history clear(*BPS)

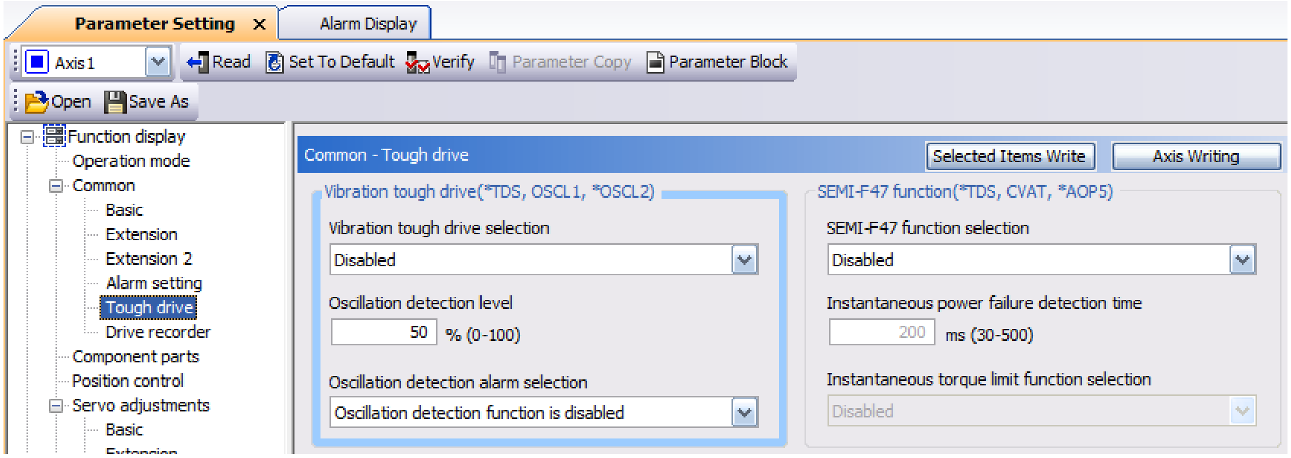
Alarm history clear: Disabled

Undervoltage alarm detection method(*COP7)

Undervoltage alarm detection method selection: When undervoltage (AL.10) not occurred

Overspeed alarm detection level(OSL)

Overspeed alarm detection level: 0 r/min (0-20000)



Common - Tough drive

Vibration tough drive(*TDS, OSCL1, *OSCL2)

Vibration tough drive selection: Disabled

Oscillation detection level: 50 % (0-100)

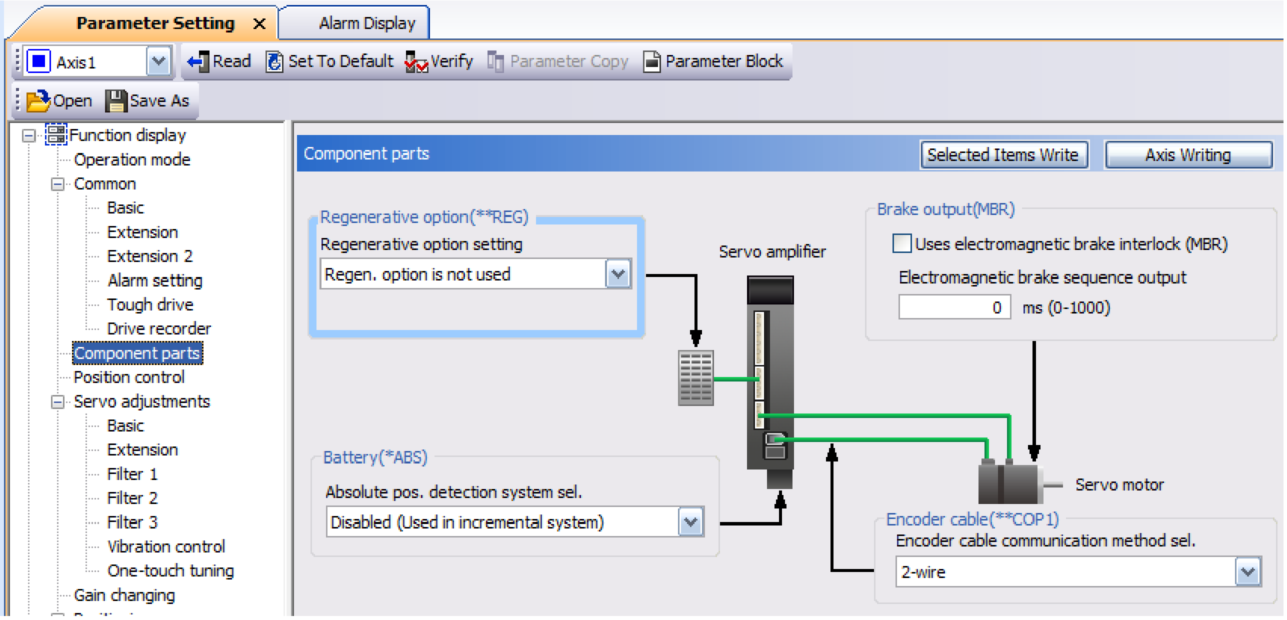
Oscillation detection alarm selection: Oscillation detection function is disabled

SEMI-F47 function(*TDS, CVAT, *AOP5)

SEMI-F47 function selection: Disabled

Instantaneous power failure detection time: 200 ms (30-500)

Instantaneous torque limit function selection: Disabled



Component parts

Regenerative option(**REG)

Regenerative option setting: Regen. option is not used

Brake output(MBR)

☐ Uses electromagnetic brake interlock (MBR)

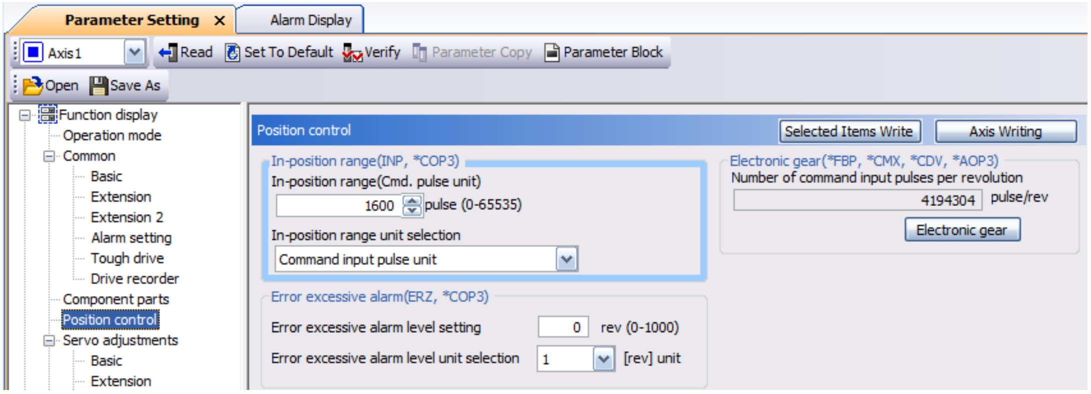
Electromagnetic brake sequence output: 0 ms (0-1000)

Battery(*ABS)

Absolute pos. detection system sel.: Disabled (Used in incremental system)

Encoder cable(**COP1)

Encoder cable communication method sel.: 2-wire



Position control

In-position range(INP, *COP3)

In-position range(Cmd. pulse unit) 1600 pulse (0-65535)

In-position range unit selection: Command input pulse unit

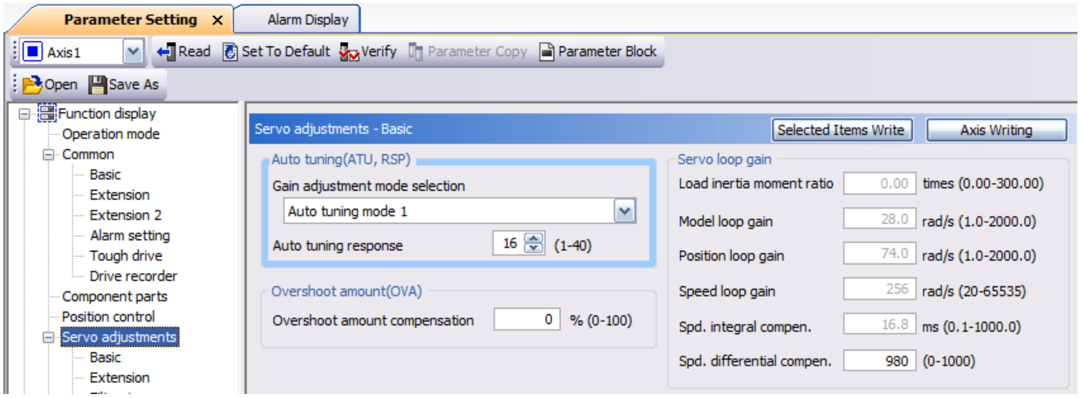
Error excessive alarm(ERZ, *COP3)

Error excessive alarm level setting 0 rev (0-1000)

Error excessive alarm level unit selection 1 [rev] unit

Electronic gear(*FBP, *CMX, *CDV, *AOP3)

Number of command input pulses per revolution 4194304 pulse/rev



Servo adjustments - Basic

Auto tuning(ATU, RSP)

Gain adjustment mode selection: Auto tuning mode 1

Auto tuning response 16 (1-40)

Overshoot amount(OVA)

Overshoot amount compensation 0 % (0-100)

Servo loop gain

Load inertia moment ratio 0.00 times (0.00-300.00)

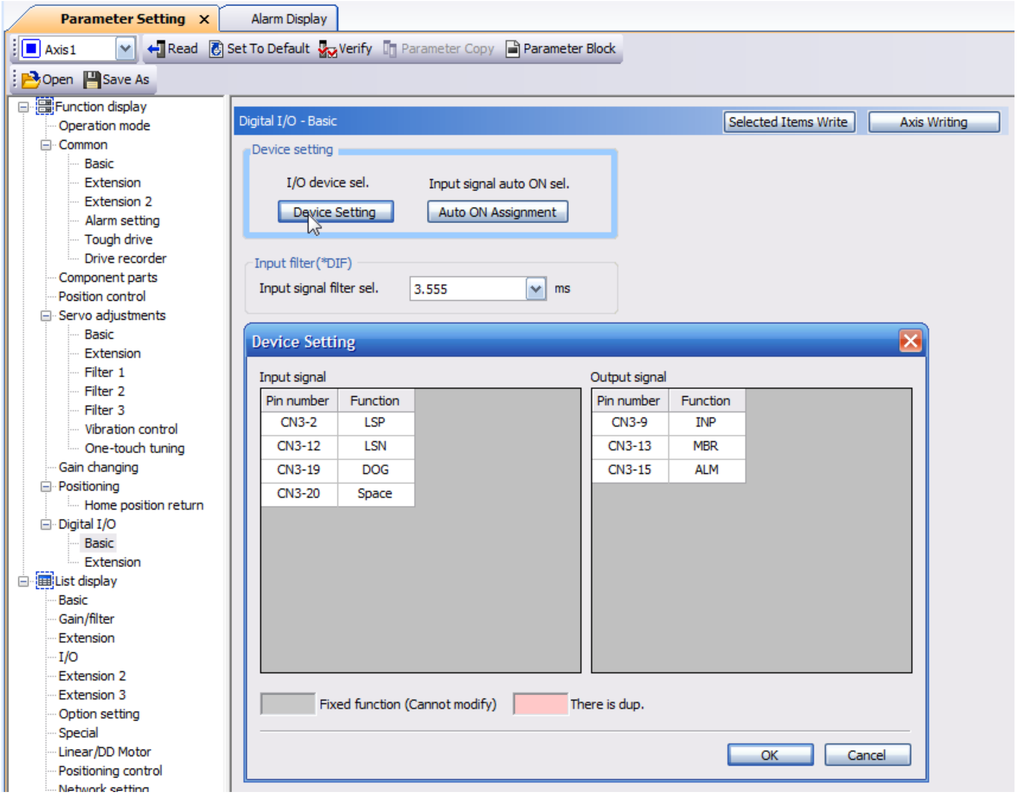
Model loop gain 28.0 rad/s (1.0-2000.0)

Position loop gain 74.0 rad/s (1.0-2000.0)

Speed loop gain 256 rad/s (20-65535)

Spd. integral compen. 16.8 ms (0.1-1000.0)

Spd. differential compen. 980 (0-1000)



Digital I/O - Basic

Device setting

I/O device sel.

Input signal auto ON sel.

Input filter(*DIF)

Input signal filter sel. 3.555 ms

Device Setting

Input signal

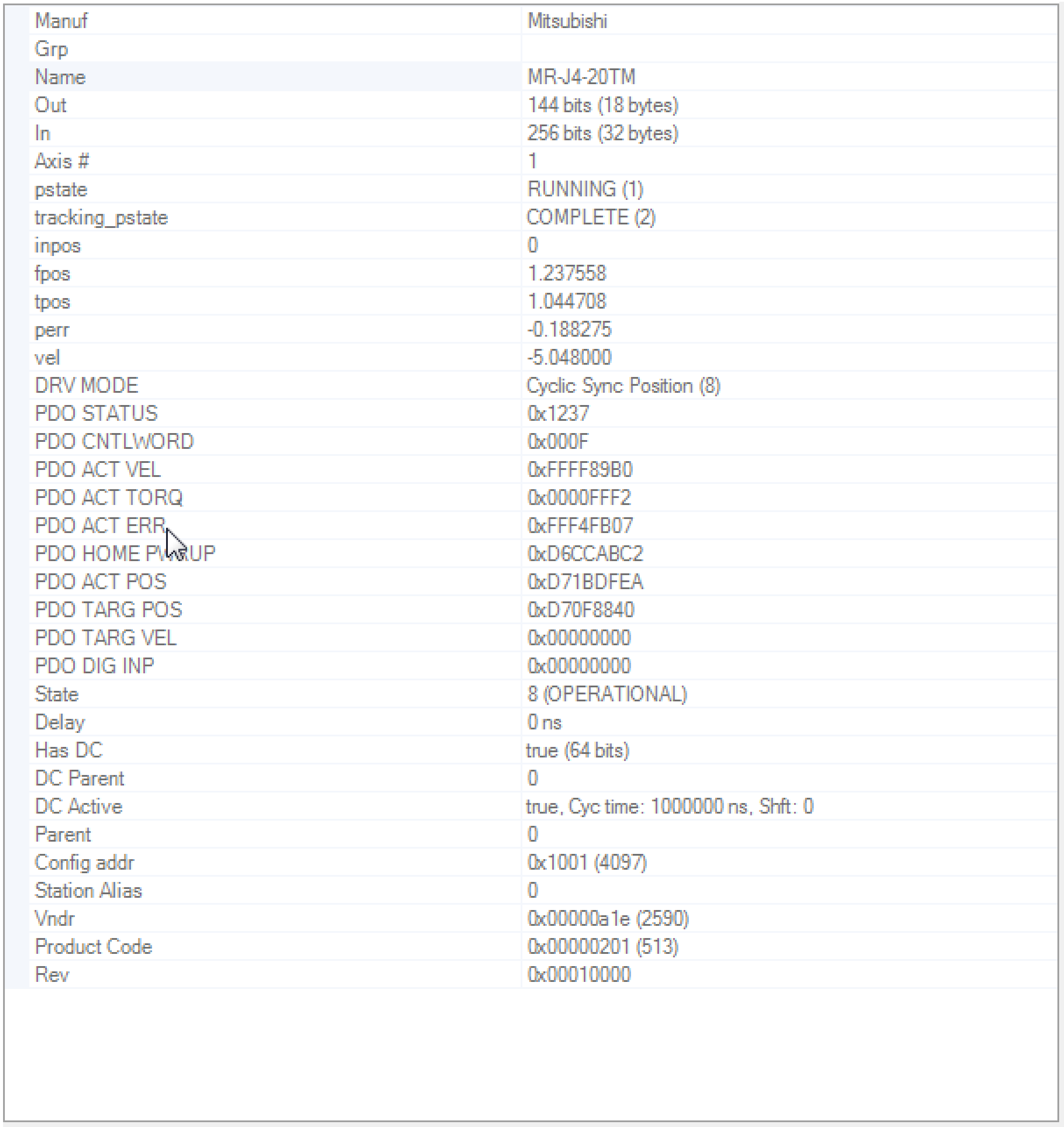
Pin number	Function
CN3-2	LSP
CN3-12	LSN
CN3-19	DOG
CN3-20	Space

Output signal

Pin number	Function
CN3-9	INP
CN3-13	MBR
CN3-15	ALM

▨ Fixed function (Cannot modify) 🟥 There is dup.

EtherCAT Explorer View



Manuf	Mitsubishi
Grp	
Name	MR-J4-20TM
Out	144 bits (18 bytes)
In	256 bits (32 bytes)
Axis #	1
pstate	RUNNING (1)
tracking_pstate	COMPLETE (2)
inpos	0
fpos	1.237558
tpos	1.044708
perr	-0.188275
vel	-5.048000
DRV MODE	Cyclic Sync Position (8)
PDO STATUS	0x1237
PDO CNTLWORD	0x000F
PDO ACT VEL	0xFFFF89B0
PDO ACT TORQ	0x0000FFF2
PDO ACT ERR	0xFFF4FB07
PDO HOME PWRUP	0xD6CCABC2
PDO ACT POS	0xD71BDFEA
PDO TARG POS	0xD70F8840
PDO TARG VEL	0x00000000
PDO DIG INP	0x00000000
State	8 (OPERATIONAL)
Delay	0 ns
Has DC	true (64 bits)
DC Parent	0
DC Active	true, Cyc time: 1000000 ns, Shft: 0
Parent	0
Config addr	0x1001 (4097)
Station Alias	0
Vndr	0x00000a1e (2590)
Product Code	0x00000201 (513)
Rev	0x00010000

Note: Rev shows 0x00020001 on the later model MR-J4-40TM1 and should be the firmware of choice.

Blank

Appendix

J

[J] Numatics (Emerson)

Numatics, a division of Emerson, manufactures a number solenoid valves and manifolds as well as I/O modules. That currently supported is the 501 & G3 Series. This section provides information that may be specific to this manufacturer.

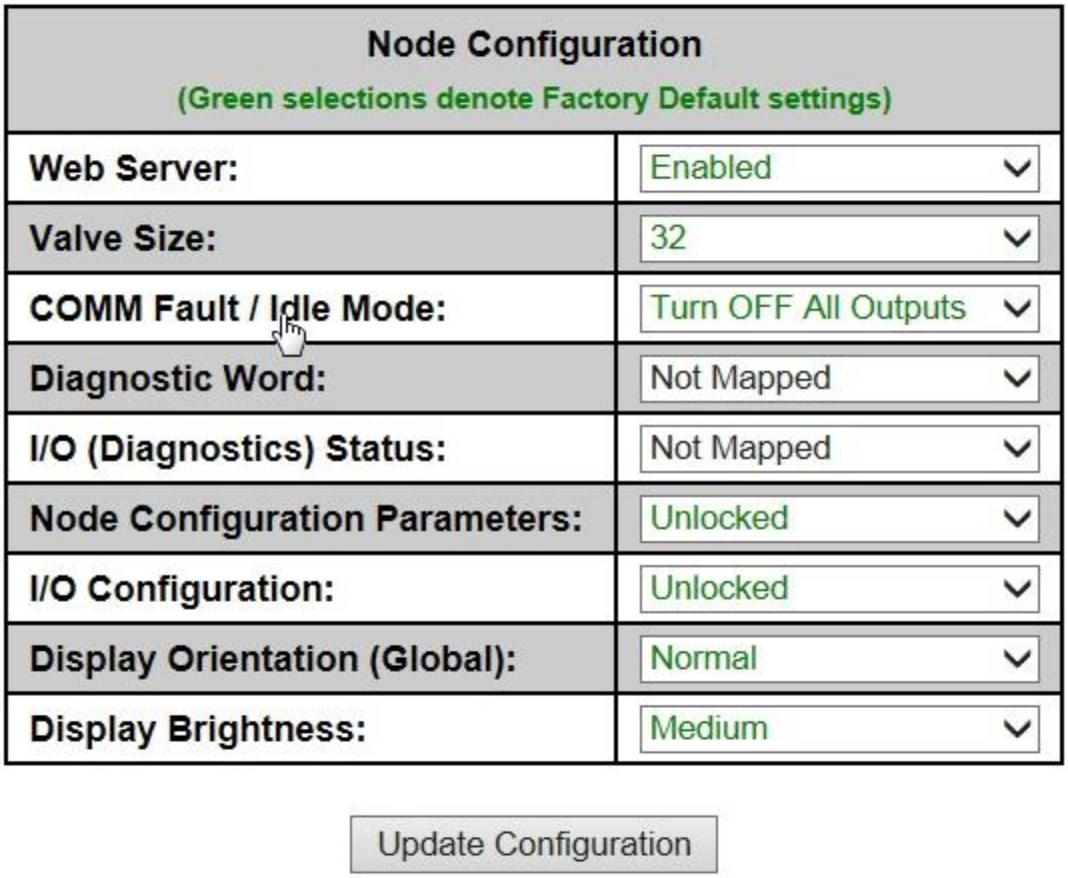
Valve Modules

The Valve Driver Module uses the first 32 outputs regardless of how many actual valve drivers are installed. This IO space is reserved by the Numatics controller and must be considered when mapping the 5300 IO. Ensure that the firmware is at the proper level, 1.1.42194. This firmware has special features and firmware enhancements for proper EtherCAT operation. Older firmware will not work properly.

Input Mapping (Required for New Controllers)

Prior to operation all Diagnostic Word & Status input bytes must be disabled or network inputs will not be aligned. This is done using the built-in web server on the Numatics unit. Numatics has documentation for this in chapter 10 of the “G3 Series EtherCAT Technical Manual” but in summary:

1. Make sure the Numatics controller EtherCAT Input port is connected to a network switch, not directly to a PC, unless a crossover cable is used.
2. Set a dedicated PC Ethernet adapter port with a static IP address of 192.168.3.100 and subnet of 255.255.255.0. Nothing is listed for the DNS server.
3. On the Numatics controller you must disable EtherCAT and enable the Web Server. This will enable TCP only on the controller and allow communications to a PC browser. This is done using the SET/NEXT buttons on the EtherCAT controller. Cycle power after the changes.
4. On the PC open a web browser and enter the IP address of the controller, 192.168.3.200. A web page should appear.
5. Set the Diagnostic Word and I/O (Diagnostics) Status to “Not Mapped” and select “Update Configuration”.



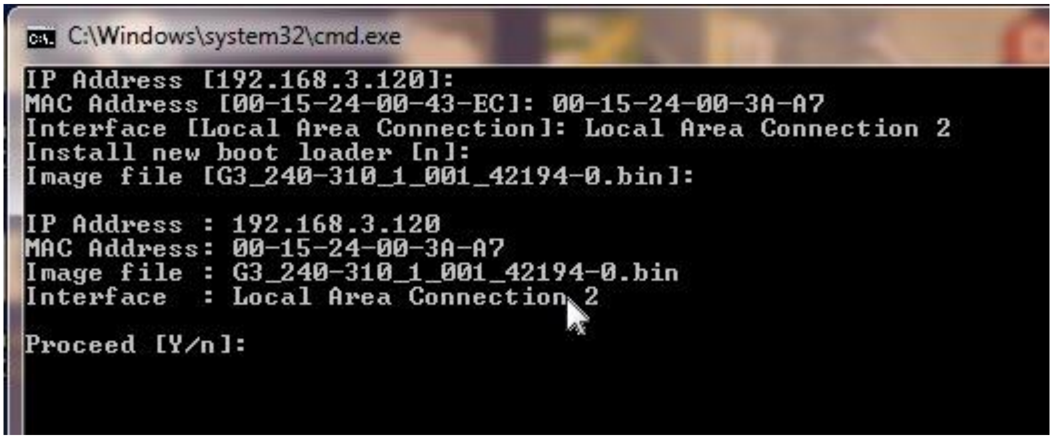
Node Configuration (Green selections denote Factory Default settings)	
Web Server:	Enabled
Valve Size:	32
COMM Fault / Idle Mode:	Turn OFF All Outputs
Diagnostic Word:	Not Mapped
I/O (Diagnostics) Status:	Not Mapped
Node Configuration Parameters:	Unlocked
I/O Configuration:	Unlocked
Display Orientation (Global):	Normal
Display Brightness:	Medium

6. When complete set the controller back to EtherCAT Enabled and Web Server Disabled, cycle power.

Firmware Updates

To update firmware the tftp protocol is used. A PC is required with a dedicated Ethernet port. That port should be connected to a network switch, the Numatics controller should also be connected to the same switch from the EtherCAT module. EtherCAT must be disabled and the Web Server enabled using the SET/NEXT buttons. Numatics provides documentation on the proper update of their firmware and it can also be provided by CTC if needed. In summary:

7. Set a dedicated PC Ethernet adapter port with a static IP address of 192.168.3.100 and subnet of 255.255.255.0. Nothing is listed for the DNS server.
8. The tftp client protocol must be enabled as a Windows Feature.
9. Cycle power on the Numatics controller with the previously described option changes to the Numatics EtherCAT module.
10. Scroll through the EtherCAT module settings and note the MAC address of the unit.
11. Run the Numatics tftp-load.bat program.
12. Enter information for the MAC Address, you LAN Interface used, similar to below:



```
IP Address [192.168.3.120]:
MAC Address [00-15-24-00-43-EC]: 00-15-24-00-3A-A7
Interface [Local Area Connection]: Local Area Connection 2
Install new boot loader [n]:
Image file [G3_240-310_1_001_42194-0.bin]:

IP Address : 192.168.3.120
MAC Address: 00-15-24-00-3A-A7
Image file : G3_240-310_1_001_42194-0.bin
Interface  : Local Area Connection 2

Proceed [Y/n]:
```

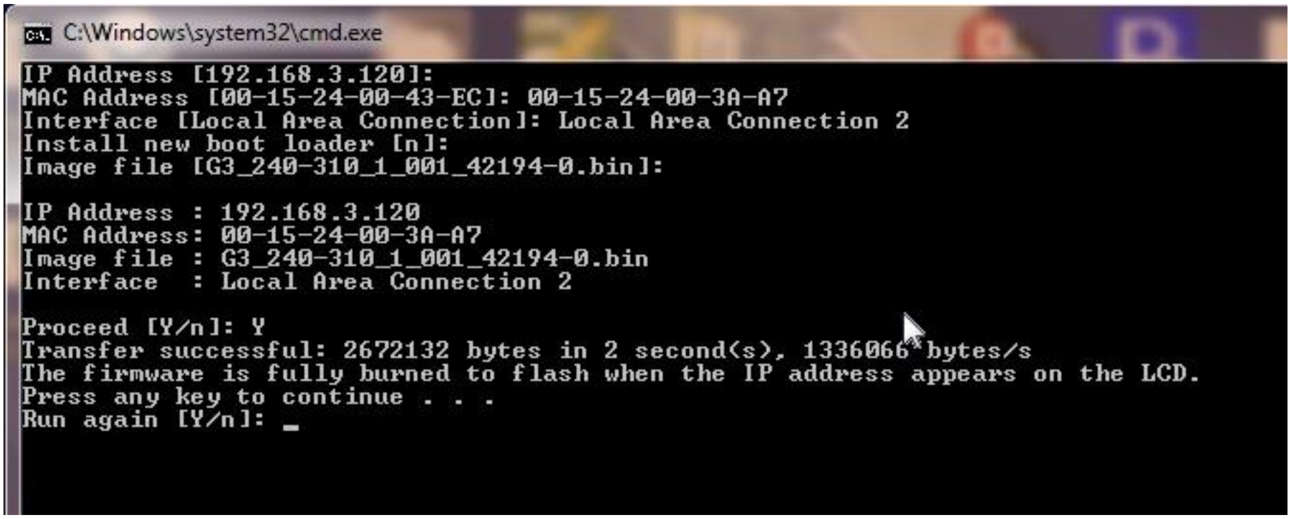
13. On the Numatics controller:

- Push NEXT on the graphic display of the G3 Node until Diagnostics is shown

- Push SET to enter the menu
- Push NEXT to move through the Diagnostics menu until "LOAD FIRMWARE" is shown
- Push SET to enter the menu
- Select yes by pushing SET and then NEXT until YES is highlighted.
- Push SET to select YES
- Push SET again to confirm choice
- Display should start flashing "WAITING FOR FIRMWARE LOAD"
- Now select Y

- You will see a transfer successful message in DOS if everything transferred OK
- Wait for the G3 node to reboot **DO NOT CYCLE POWER ON THE NODE!**

You may confirm the successful firmware download by verifying the firmware revision on the G3 web page or through the display.

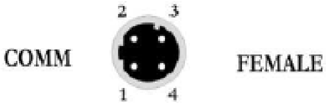


```
IP Address [192.168.3.120]:
MAC Address [00-15-24-00-43-EC]: 00-15-24-00-3A-A7
Interface [Local Area Connection]: Local Area Connection 2
Install new boot loader [n]:
Image file [G3_240-310_1_001_42194-0.bin]:

IP Address : 192.168.3.120
MAC Address: 00-15-24-00-3A-A7
Image file : G3_240-310_1_001_42194-0.bin
Interface  : Local Area Connection 2

Proceed [Y/n]: Y
Transfer successful: 2672132 bytes in 2 second(s), 1336066 bytes/s
The firmware is fully burned to flash when the IP address appears on the LCD.
Press any key to continue . . .
Run again [Y/n]: _
```

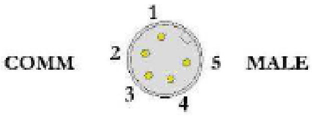
Power Connector

Power Connector Pin-Out

Pin No.	*Function*	*Description*
1	0 VDC Common (Valves and Outputs)	0 VDC Voltage used to power outputs (valve coils and discrete outputs) SW
2	0 VDC Common (Node and Inputs)	0 VDC Voltage used to power discrete inputs and node electronics UNSW
3	Earth Ground	Protective Earth
4	+24 VDC (Node and Inputs)	Voltage used to power discrete inputs and node electronics UNSW
5	+24 VDC (Valves and Outputs)	Voltage used to power outputs (valve coils and discrete outputs) SW



PIN 1= TX +
PIN 2= RX +
PIN 3= TX -
PIN 4= RX -



PIN 1= 0 VDC VALVES & OUT (SW)
PIN 2= 0 VDC NODE & IN (UNSW)
PIN 3= EARTH GROUND
PIN 4= +24 VDC NODE & IN (UNSW)
PIN 5= +24 VDC VALVES & OUT (SW)

ATTENTION

- *Power common (0 VDC) pins 1 and 2 are isolated from each other to allow separate (isolated) power supply connection if required. However, they can be tied together if a single common, non-isolated, application is preferred.*
- *The combined draw of the +24VDC Valves and Outputs and +24VDC Node and Inputs pins cannot exceed 8 Amps, at any given moment in time.*
- *The Node and Inputs pin supplies power to the node electronics. This pin must be powered at all times for communication node to be functional.*

Appendix

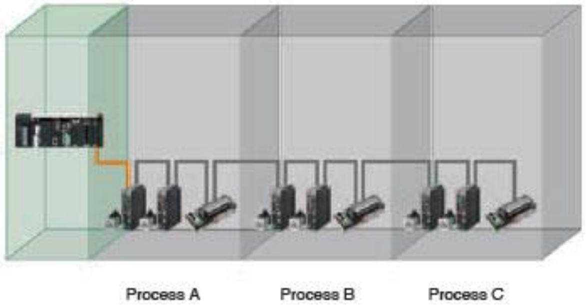
K

[K] Omron GX-JC06 EtherCAT Junction Slaves

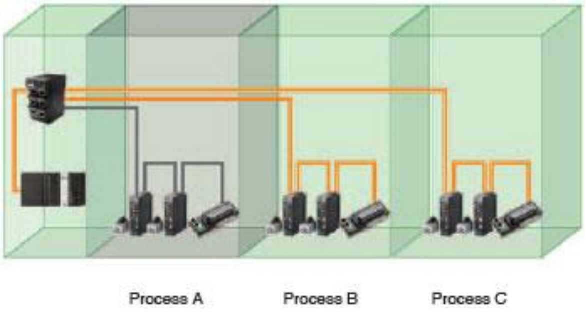
EtherCAT devices are typically wired in a daisy chain, where an IN/OUT port exist on each device. The Omron EtherCAT Junction Slave is similar to an Ethernet switch, allowing for flexible cabling. Detailed information can be found on their web site: http://www.ia.omron.com/products/family/3079/feature.html.

Cabling

Typical EtherCAT cabling as detailed by Omron:

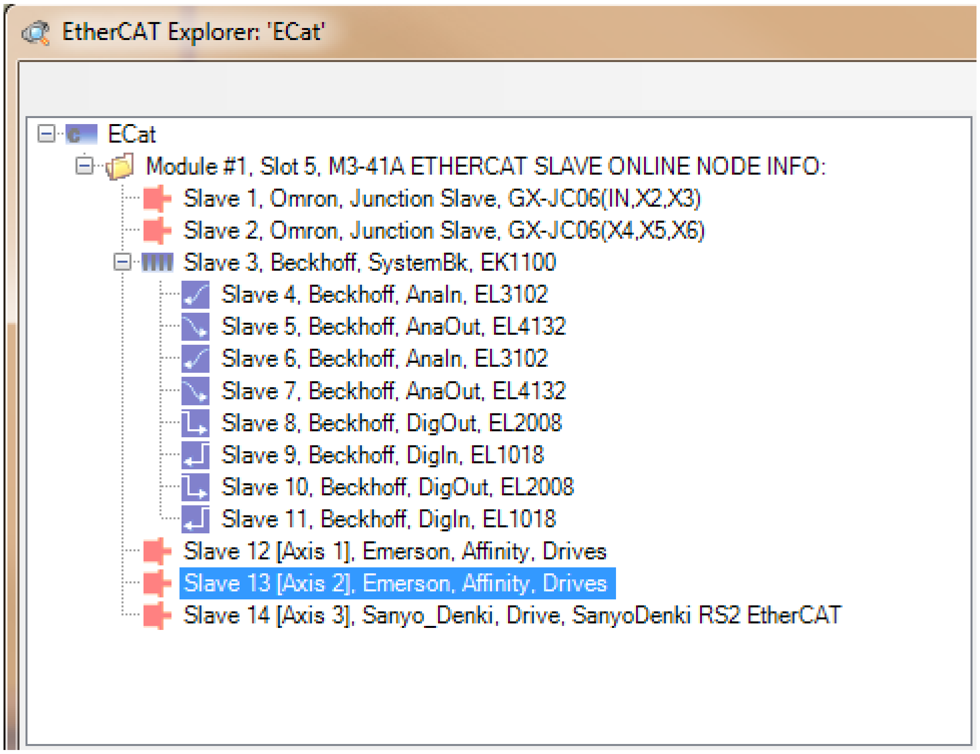


Cabling with an EtherCAT Junction Slave:



Example with M3-41 Module

The following example shows how address assignment is done using the Junction Slave. In this example the M3-41 master is plugged into the Junction Slave IN port (1). The Beckhoff IO is plugged into port 4, Emerson drives into port 5, and Sanyo Denki into port 6.



EtherCAT Explorer: 'ECat'

```
ECat
  Module #1, Slot 5, M3-41A ETHERCAT SLAVE ONLINE NODE INFO:
    Slave 1, Omron, Junction Slave, GX-JC06(IN,X2,X3)
    Slave 2, Omron, Junction Slave, GX-JC06(X4,X5,X6)
    Slave 3, Beckhoff, SystemBk, EK1100
      Slave 4, Beckhoff, AnaIn, EL3102
      Slave 5, Beckhoff, AnaOut, EL4132
      Slave 6, Beckhoff, AnaIn, EL3102
      Slave 7, Beckhoff, AnaOut, EL4132
      Slave 8, Beckhoff, DigOut, EL2008
      Slave 9, Beckhoff, DigIn, EL1018
      Slave 10, Beckhoff, DigOut, EL2008
      Slave 11, Beckhoff, DigIn, EL1018
    Slave 12 [Axis 1], Emerson, Affinity, Drives
    Slave 13 [Axis 2], Emerson, Affinity, Drives
    Slave 14 [Axis 3], Sanyo_Denki, Drive, SanyoDenki RS2 EtherCAT
```

Note that the ports that are not used have no effect on the address assignment. All of the devices on a lower number port are assigned addresses first, followed by those on the next higher port number. The Junction Slave device itself appears as one or more slave nodes to the EtherCAT master and can operate as a 64 bit distributed clock reference.

Appendix

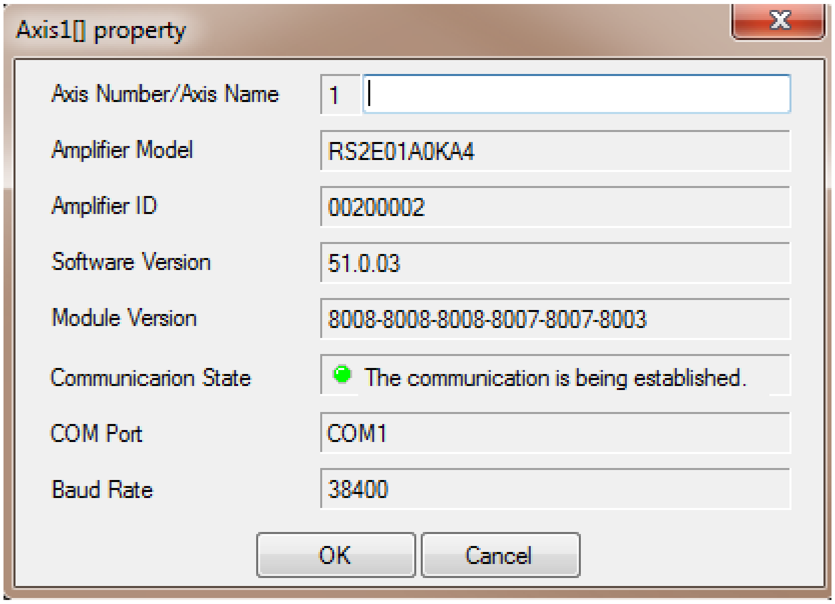
L

[L] Sanyo Denki

Sanyo Denki manufactures a number of drives. The drive currently supported is the single axis RS2E. The configuration utility used is “R Advanced Model – Setup Software”. This section provides information that may be specific to this manufacturer.

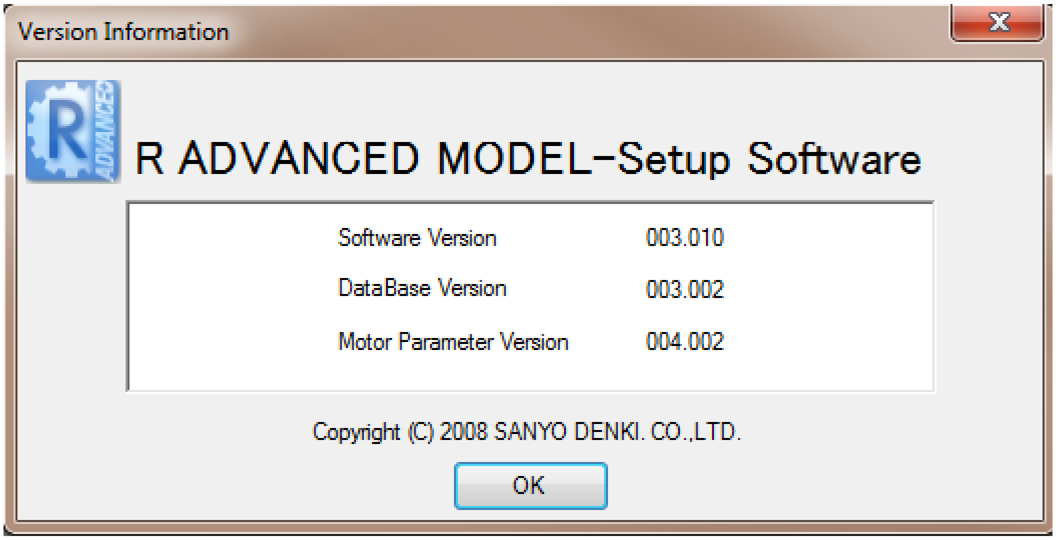
eCAT_driveType – 6

Drive Information & Firmware



Axis1[] property

Axis Number/Axis Name	1
Amplifier Model	RS2E01A0KA4
Amplifier ID	00200002
Software Version	51.0.03
Module Version	8008-8008-8008-8007-8007-8003
Communicarion State	● The communication is being established.
COM Port	COM1
Baud Rate	38400



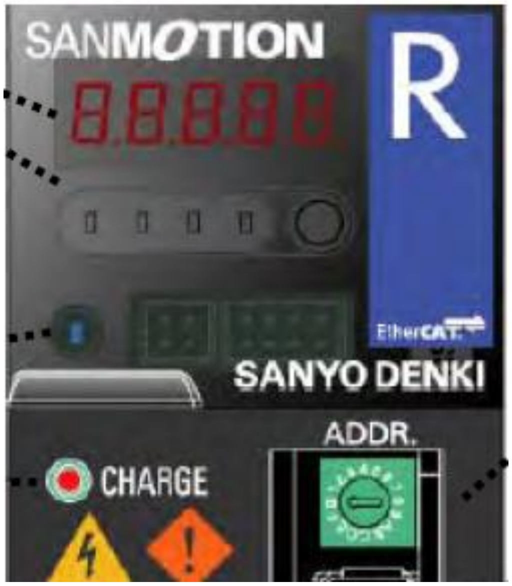
R ADVANCED MODEL–Setup Software

Software Version	003.010
DataBase Version	003.002
Motor Parameter Version	004.002

Copyright (C) 2008 SANYO DENKI. CO.,LTD.

Station Alias

In an EtherCAT network, slaves are automatically assigned addresses based on their position in the bus. When a device, such as a drive, must have a fixed assigned identification that is independent of cabling, a Station Alias is needed. Sanyo Denki provides a single 16-position rotary switch with hexadecimal encoding for this purpose. This 4-bit switch is used to set the lower bits 3 to 0 of the alias, while bits 15 to 4 are written using the Setup Software, with a default of 0. Since the M3-41 only supports up to 16 drives, the single switch will suffice using the setup software defaults for the upper bits. A switch setting of 0 will default to automatic addressing.



Sanyo Denki Drive Mode Transitions

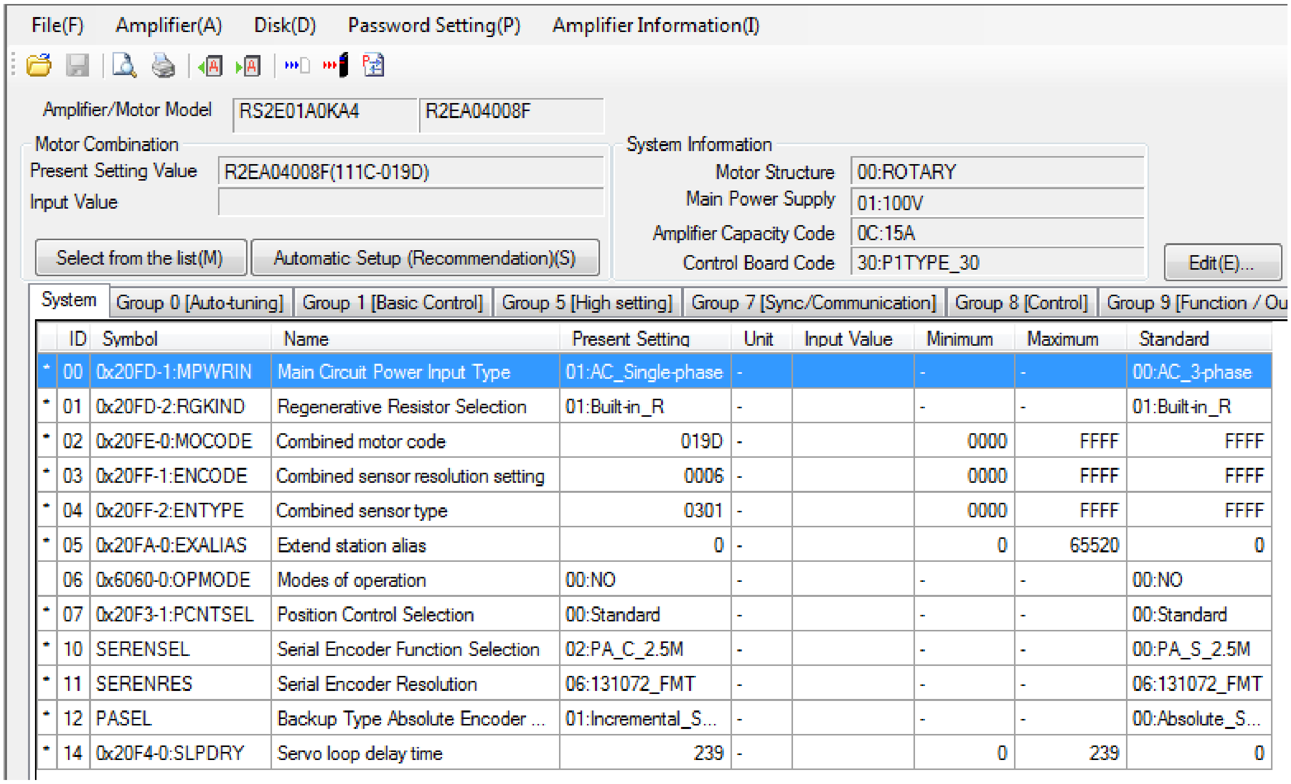
Unlike other drive manufacturers, Sanyo Denki requires that the Operation Enabled bit be disabled in the control word prior to changing a drive mode (such as cyclic sync position mode to profile position mode). To allow this to operate correctly, a 500-millisecond delay will occur during the first move instruction after changing a mode. This will only happen once, until a mode is changed again. Additional move instructions using the same drive mode will not be delayed.

Sanyo Denki Power-Up Delay

As just discussed the Sanyo Denki drive requires a 500-millisecond delay after a drive mode change. This also includes transitioning to Cyclic Sync Position mode at power up. The 500-millisecond delay will occur with the first move. Additional commands following the first move will not be delayed.

R Advanced Model – Setup Software

CTC used R Advanced Model Setup Software, on a Windows 7 64-bit computer to test and configure the drives. Typically R Advanced Model will be needed for tuning and limited setup. Below are some setup screens and their typical parameters at power up (offline).

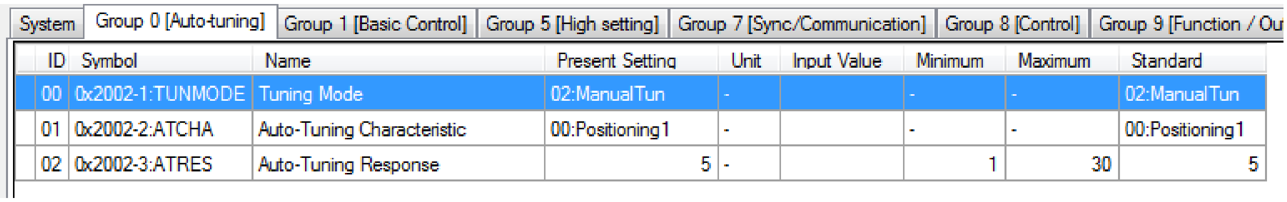


Amplifier/Motor Model: RS2E01A0KA4 | R2EA04008F

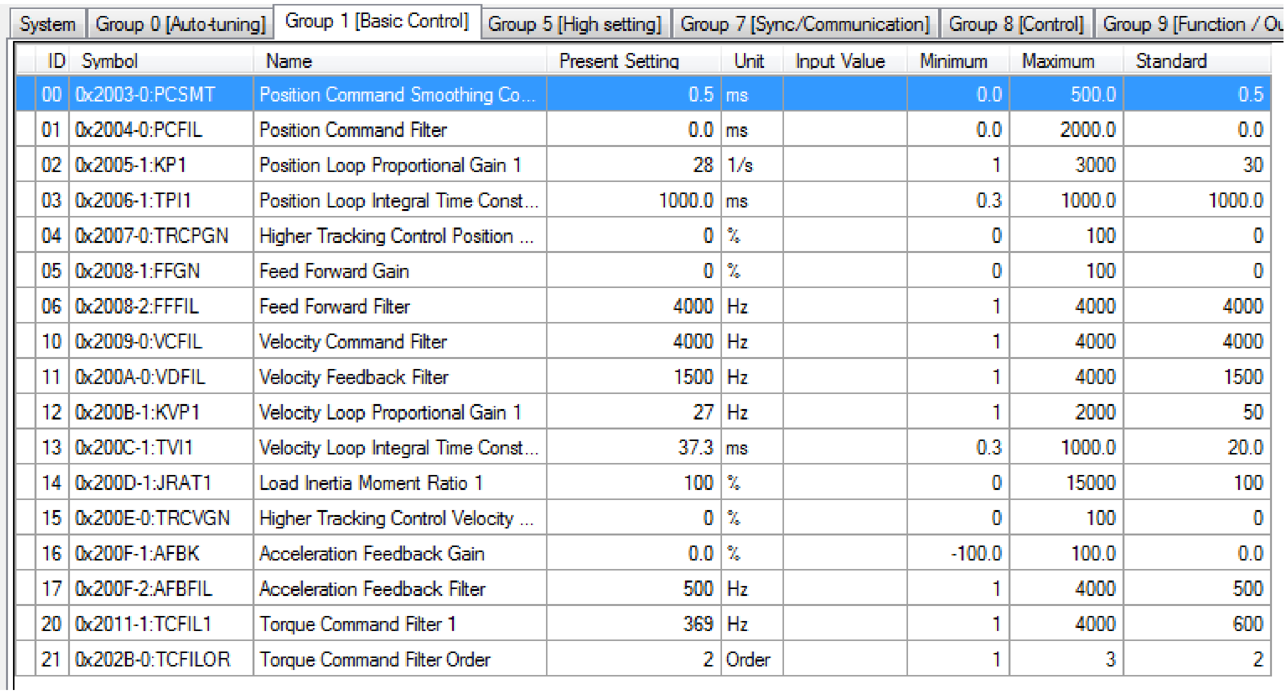
Motor Combination
Present Setting Value: R2EA04008F(111C-019D)
Input Value:

System Information
Motor Structure: 00:ROTARY
Main Power Supply: 01:100V
Amplifier Capacity Code: 0C:15A
Control Board Code: 30:P1TYPE_30

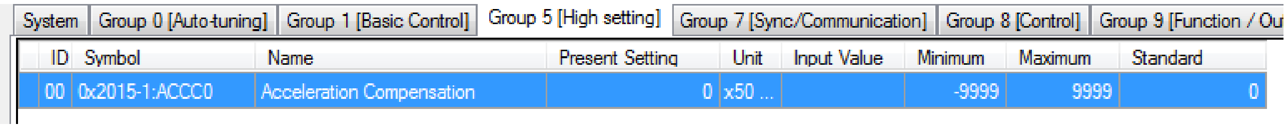
	ID	Symbol	Name	Present Setting	Unit	Input Value	Minimum	Maximum	Standard
*	00	0x20FD-1:MPWRIN	Main Circuit Power Input Type	01:AC_Single-phase	-		-	-	00:AC_3-phase
*	01	0x20FD-2:RGKIND	Regenerative Resistor Selection	01:Built-in_R	-		-	-	01:Built-in_R
*	02	0x20FE-0:MOCODE	Combined motor code	019D	-		0000	FFFF	FFFF
*	03	0x20FF-1:ENCODE	Combined sensor resolution setting	0006	-		0000	FFFF	FFFF
*	04	0x20FF-2:ENTYPE	Combined sensor type	0301	-		0000	FFFF	FFFF
*	05	0x20FA-0:EXALIAS	Extend station alias	0	-		0	65520	0
	06	0x6060-0:OPMODE	Modes of operation	00:NO	-		-	-	00:NO
*	07	0x20F3-1:PCNTSEL	Position Control Selection	00:Standard	-		-	-	00:Standard
*	10	SERENSEL	Serial Encoder Function Selection	02:PA_C_2.5M	-		-	-	00:PA_S_2.5M
*	11	SERENRES	Serial Encoder Resolution	06:131072_FMT	-		-	-	06:131072_FMT
*	12	PASEL	Backup Type Absolute Encoder ...	01:Incremental_S...	-		-	-	00:Absolute_S...
*	14	0x20F4-0:SLPDRY	Servo loop delay time	239	-		0	239	0



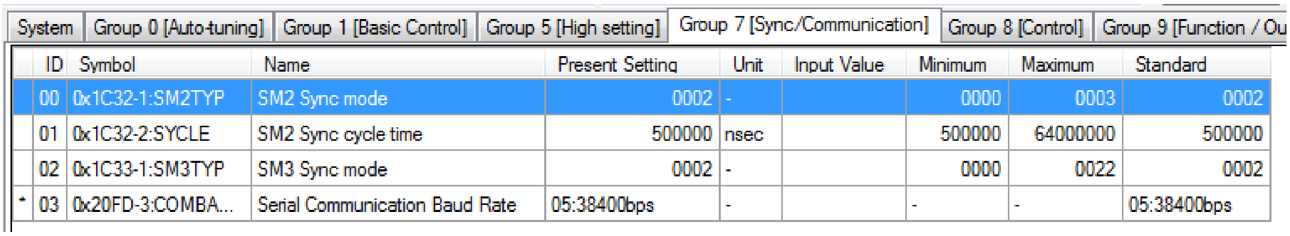
ID	Symbol	Name	Present Setting	Unit	Input Value	Minimum	Maximum	Standard
00	0x2002-1:TUNMODE	Tuning Mode	02:ManualTun	-		-	-	02:ManualTun
01	0x2002-2:ATCHA	Auto-Tuning Characteristic	00:Positioning1	-		-	-	00:Positioning1
02	0x2002-3:ATRES	Auto-Tuning Response	5	-		1	30	5



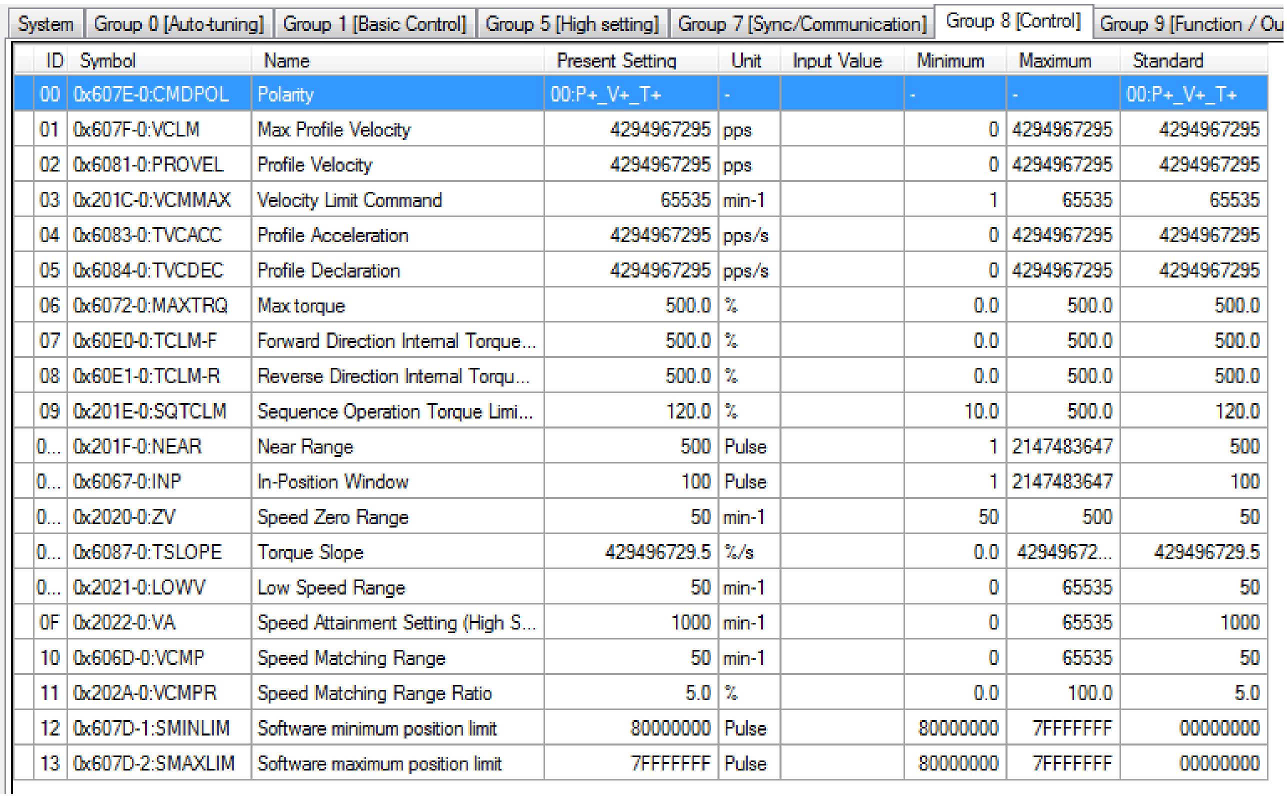
ID	Symbol	Name	Present Setting	Unit	Input Value	Minimum	Maximum	Standard
00	0x2003-0:PCSMT	Position Command Smoothing Co...	0.5	ms		0.0	500.0	0.5
01	0x2004-0:PCFIL	Position Command Filter	0.0	ms		0.0	2000.0	0.0
02	0x2005-1:KP1	Position Loop Proportional Gain 1	28	1/s		1	3000	30
03	0x2006-1:TPI1	Position Loop Integral Time Const...	1000.0	ms		0.3	1000.0	1000.0
04	0x2007-0:TRCPGN	Higher Tracking Control Position ...	0	%		0	100	0
05	0x2008-1:FFGN	Feed Forward Gain	0	%		0	100	0
06	0x2008-2:FFFIL	Feed Forward Filter	4000	Hz		1	4000	4000
10	0x2009-0:VCFIL	Velocity Command Filter	4000	Hz		1	4000	4000
11	0x200A-0:VDFIL	Velocity Feedback Filter	1500	Hz		1	4000	1500
12	0x200B-1:KVP1	Velocity Loop Proportional Gain 1	27	Hz		1	2000	50
13	0x200C-1:TVI1	Velocity Loop Integral Time Const...	37.3	ms		0.3	1000.0	20.0
14	0x200D-1:JRAT1	Load Inertia Moment Ratio 1	100	%		0	15000	100
15	0x200E-0:TRCVGN	Higher Tracking Control Velocity ...	0	%		0	100	0
16	0x200F-1:AFBK	Acceleration Feedback Gain	0.0	%		-100.0	100.0	0.0
17	0x200F-2:AFBFIL	Acceleration Feedback Filter	500	Hz		1	4000	500
20	0x2011-1:TCFIL1	Torque Command Filter 1	369	Hz		1	4000	600
21	0x202B-0:TCFILOR	Torque Command Filter Order	2	Order		1	3	2



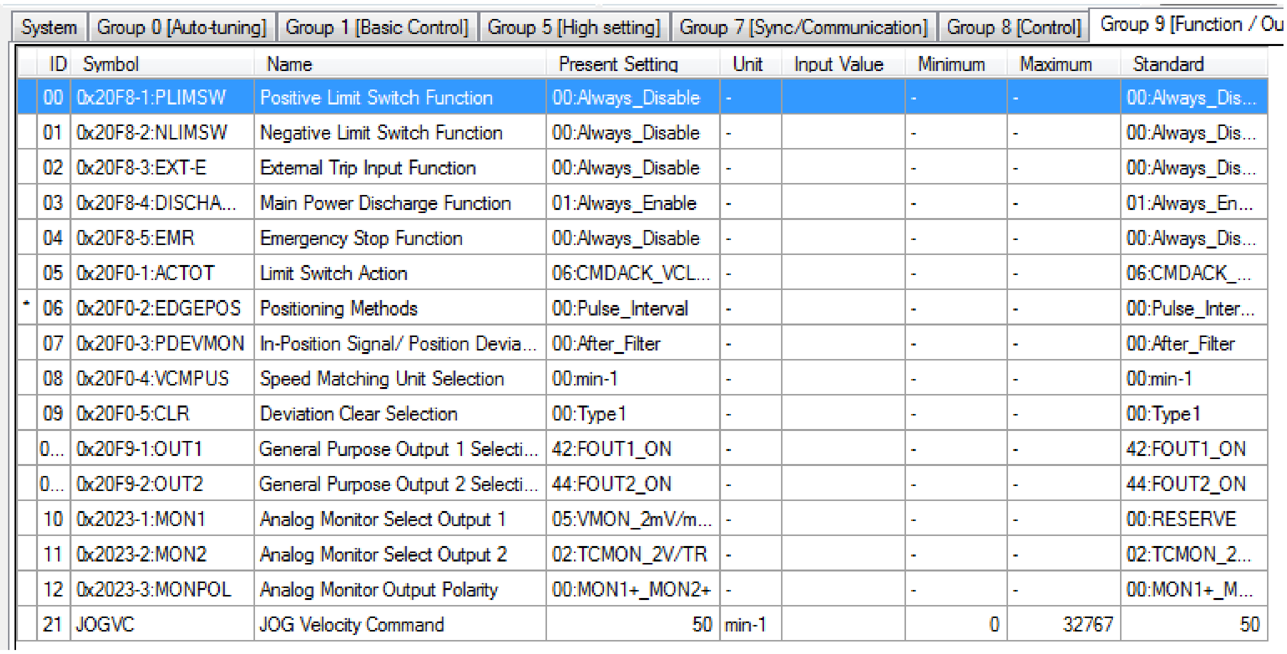
ID	Symbol	Name	Present Setting	Unit	Input Value	Minimum	Maximum	Standard
00	0x2015-1:ACCC0	Acceleration Compensation	0	x50 ...		-9999	9999	0



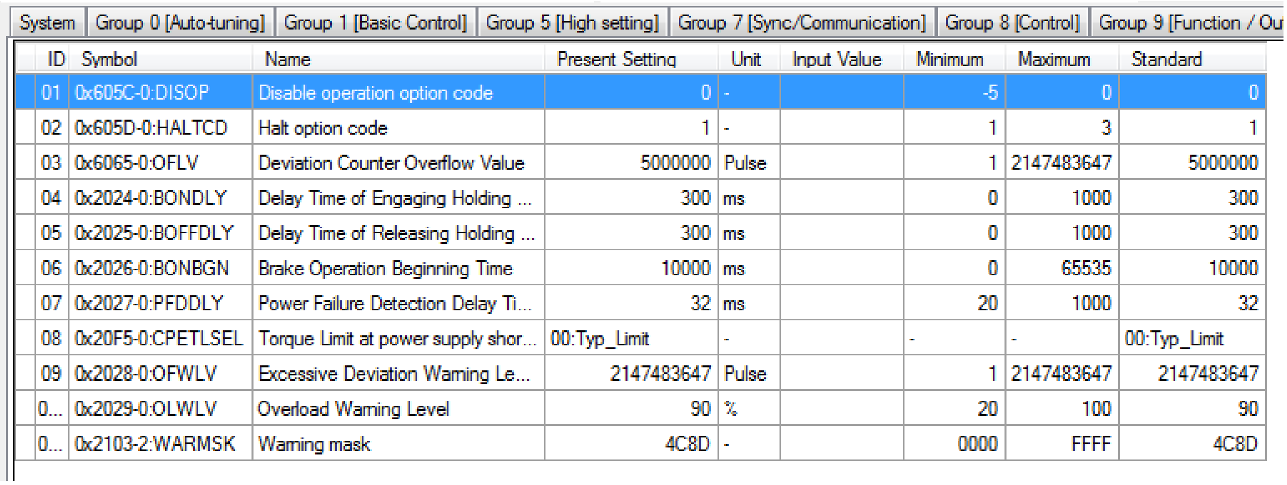
	ID	Symbol	Name	Present Setting	Unit	Input Value	Minimum	Maximum	Standard
	00	0x1C32-1:SM2TYP	SM2 Sync mode	0002	-		0000	0003	0002
	01	0x1C32-2:SYCLE	SM2 Sync cycle time	500000	nsec		500000	64000000	500000
	02	0x1C33-1:SM3TYP	SM3 Sync mode	0002	-		0000	0022	0002
*	03	0x20FD-3:COMBA...	Serial Communication Baud Rate	05:38400bps	-		-	-	05:38400bps



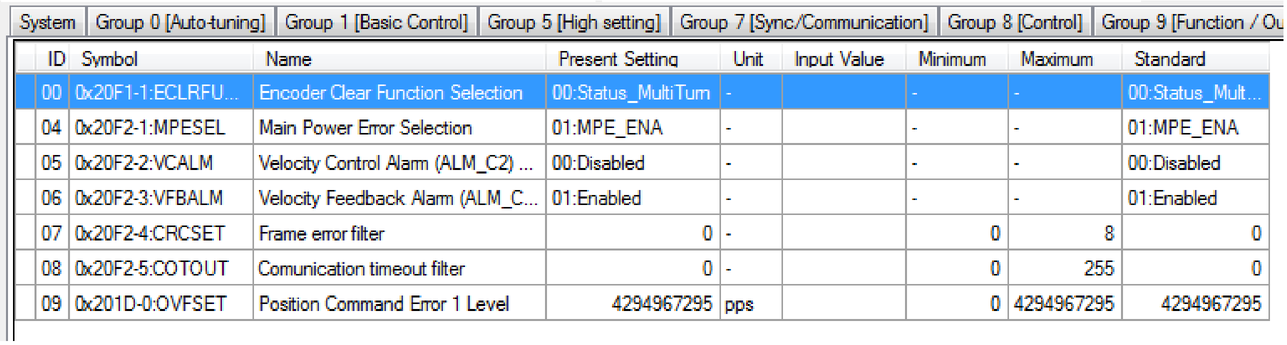
ID	Symbol	Name	Present Setting	Unit	Input Value	Minimum	Maximum	Standard
00	0x607E-0:CMDPOL	Polarity	00:P+_V+_T+	-		-	-	00:P+_V+_T+
01	0x607F-0:VCLM	Max Profile Velocity	4294967295	pps		0	4294967295	4294967295
02	0x6081-0:PROVEL	Profile Velocity	4294967295	pps		0	4294967295	4294967295
03	0x201C-0:VCMMAX	Velocity Limit Command	65535	min-1		1	65535	65535
04	0x6083-0:TVCACC	Profile Acceleration	4294967295	pps/s		0	4294967295	4294967295
05	0x6084-0:TVCDEC	Profile Declaration	4294967295	pps/s		0	4294967295	4294967295
06	0x6072-0:MAXTRQ	Max torque	500.0	%		0.0	500.0	500.0
07	0x60E0-0:TCLM-F	Forward Direction Internal Torque...	500.0	%		0.0	500.0	500.0
08	0x60E1-0:TCLM-R	Reverse Direction Internal Torqu...	500.0	%		0.0	500.0	500.0
09	0x201E-0:SQTCLM	Sequence Operation Torque Limi...	120.0	%		10.0	500.0	120.0
0...	0x201F-0:NEAR	Near Range	500	Pulse		1	2147483647	500
0...	0x6067-0:INP	In-Position Window	100	Pulse		1	2147483647	100
0...	0x2020-0:ZV	Speed Zero Range	50	min-1		50	500	50
0...	0x6087-0:TSLOPE	Torque Slope	429496729.5	%/s		0.0	42949672...	429496729.5
0...	0x2021-0:LOWV	Low Speed Range	50	min-1		0	65535	50
0F	0x2022-0:VA	Speed Attainment Setting (High S...	1000	min-1		0	65535	1000
10	0x606D-0:VCMP	Speed Matching Range	50	min-1		0	65535	50
11	0x202A-0:VCMPR	Speed Matching Range Ratio	5.0	%		0.0	100.0	5.0
12	0x607D-1:SMINLIM	Software minimum position limit	80000000	Pulse		80000000	7FFFFFFF	00000000
13	0x607D-2:SMAXLIM	Software maximum position limit	7FFFFFFF	Pulse		80000000	7FFFFFFF	00000000



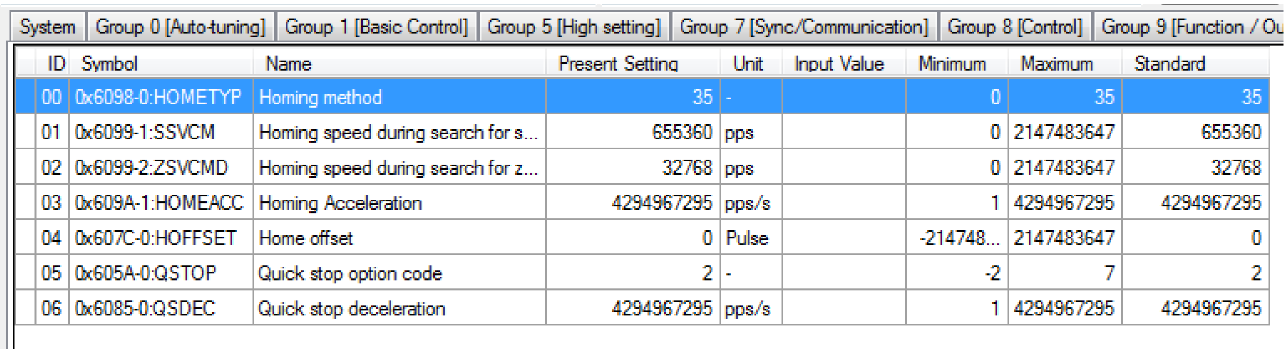
ID	Symbol	Name	Present Setting	Unit	Input Value	Minimum	Maximum	Standard
00	0x20F8-1:PLIMSW	Positive Limit Switch Function	00:Always_Disable	-		-	-	00:Always_Dis...
01	0x20F8-2:NLIMSW	Negative Limit Switch Function	00:Always_Disable	-		-	-	00:Always_Dis...
02	0x20F8-3:EXT-E	External Trip Input Function	00:Always_Disable	-		-	-	00:Always_Dis...
03	0x20F8-4:DISCHA...	Main Power Discharge Function	01:Always_Enable	-		-	-	01:Always_En...
04	0x20F8-5:EMR	Emergency Stop Function	00:Always_Disable	-		-	-	00:Always_Dis...
05	0x20F0-1:ACTOT	Limit Switch Action	06:CMDACK_VCL...	-		-	-	06:CMDACK_...
* 06	0x20F0-2:EDGEPOS	Positioning Methods	00:Pulse_Interval	-		-	-	00:Pulse_Inter...
07	0x20F0-3:PDEVMON	In-Position Signal/ Position Devia...	00:After_Filter	-		-	-	00:After_Filter
08	0x20F0-4:VCMPUS	Speed Matching Unit Selection	00:min-1	-		-	-	00:min-1
09	0x20F0-5:CLR	Deviation Clear Selection	00:Type1	-		-	-	00:Type1
0...	0x20F9-1:OUT1	General Purpose Output 1 Selecti...	42:FOUT1_ON	-		-	-	42:FOUT1_ON
0...	0x20F9-2:OUT2	General Purpose Output 2 Selecti...	44:FOUT2_ON	-		-	-	44:FOUT2_ON
10	0x2023-1:MON1	Analog Monitor Select Output 1	05:VMON_2mV/m...	-		-	-	00:RESERVE
11	0x2023-2:MON2	Analog Monitor Select Output 2	02:TCMON_2V/TR	-		-	-	02:TCMON_2...
12	0x2023-3:MONPOL	Analog Monitor Output Polarity	00:MON1+_MON2+	-		-	-	00:MON1+_M...
21	JOGVC	JOG Velocity Command	50	min-1		0	32767	50



ID	Symbol	Name	Present Setting	Unit	Input Value	Minimum	Maximum	Standard
01	0x605C-0:DISOP	Disable operation option code	0	-		-5	0	0
02	0x605D-0:HALTCD	Halt option code	1	-		1	3	1
03	0x6065-0:OFLV	Deviation Counter Overflow Value	5000000	Pulse		1	2147483647	5000000
04	0x2024-0:BONDLY	Delay Time of Engaging Holding ...	300	ms		0	1000	300
05	0x2025-0:BOFFDLY	Delay Time of Releasing Holding ...	300	ms		0	1000	300
06	0x2026-0:BONBGN	Brake Operation Beginning Time	10000	ms		0	65535	10000
07	0x2027-0:PFDDLY	Power Failure Detection Delay Ti...	32	ms		20	1000	32
08	0x20F5-0:CPETLSEL	Torque Limit at power supply shor...	00:Typ_Limit	-		-	-	00:Typ_Limit
09	0x2028-0:OFWLV	Excessive Deviation Warning Le...	2147483647	Pulse		1	2147483647	2147483647
0...	0x2029-0:OLWLV	Overload Warning Level	90	%		20	100	90
0...	0x2103-2:WARMSK	Warning mask	4C8D	-		0000	FFFF	4C8D

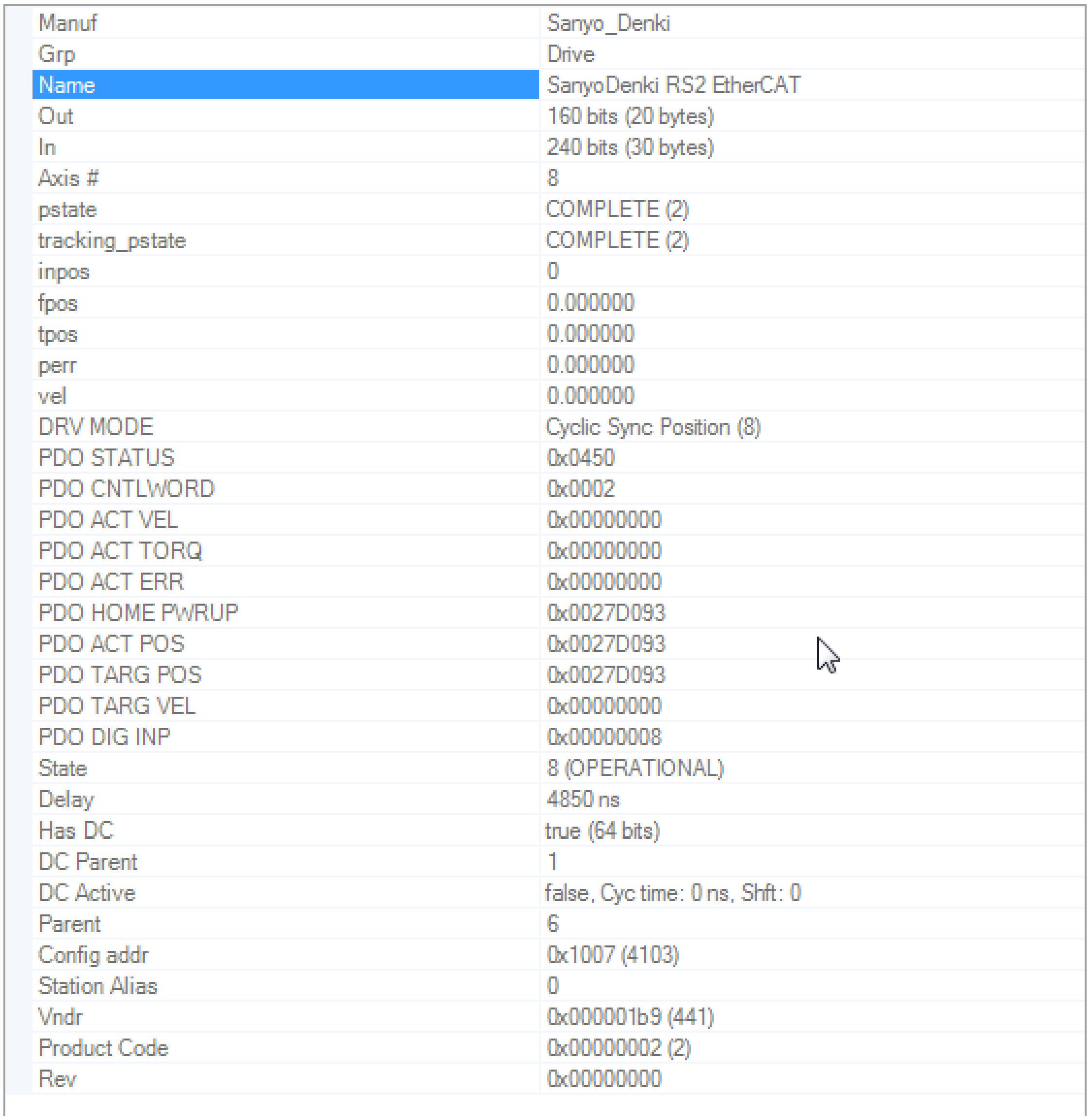


ID	Symbol	Name	Present Setting	Unit	Input Value	Minimum	Maximum	Standard
00	0x20F1-1:ECLRFU...	Encoder Clear Function Selection	00:Status_MultiTurn	-		-	-	00:Status_Mult...
04	0x20F2-1:MPESEL	Main Power Error Selection	01:MPE_ENA	-		-	-	01:MPE_ENA
05	0x20F2-2:VCALM	Velocity Control Alarm (ALM_C2) ...	00:Disabled	-		-	-	00:Disabled
06	0x20F2-3:VFBALM	Velocity Feedback Alarm (ALM_C...	01:Enabled	-		-	-	01:Enabled
07	0x20F2-4:CRCSET	Frame error filter	0	-		0	8	0
08	0x20F2-5:COTOUT	Comunication timeout filter	0	-		0	255	0
09	0x201D-0:OVFSET	Position Command Error 1 Level	4294967295	pps		0	4294967295	4294967295



ID	Symbol	Name	Present Setting	Unit	Input Value	Minimum	Maximum	Standard
00	0x6098-0:HOMETYP	Homing method	35	-		0	35	35
01	0x6099-1:SSVCM	Homing speed during search for s...	655360	pps		0	2147483647	655360
02	0x6099-2:ZSVCMD	Homing speed during search for z...	32768	pps		0	2147483647	32768
03	0x609A-1:HOMEACC	Homing Acceleration	4294967295	pps/s		1	4294967295	4294967295
04	0x607C-0:HOFFSET	Home offset	0	Pulse		-214748...	2147483647	0
05	0x605A-0:QSTOP	Quick stop option code	2	-		-2	7	2
06	0x6085-0:QSDEC	Quick stop deceleration	4294967295	pps/s		1	4294967295	4294967295

EtherCAT Explorer View



Manuf	Sanyo_Denki
Grp	Drive
Name	SanyoDenki RS2 EtherCAT
Out	160 bits (20 bytes)
In	240 bits (30 bytes)
Axis #	8
pstate	COMPLETE (2)
tracking_pstate	COMPLETE (2)
inpos	0
fpos	0.000000
tpos	0.000000
perr	0.000000
vel	0.000000
DRV MODE	Cyclic Sync Position (8)
PDO STATUS	0x0450
PDO CNTLWORD	0x0002
PDO ACT VEL	0x00000000
PDO ACT TORQ	0x00000000
PDO ACT ERR	0x00000000
PDO HOME PWRUP	0x0027D093
PDO ACT POS	0x0027D093
PDO TARG POS	0x0027D093
PDO TARG VEL	0x00000000
PDO DIG INP	0x00000008
State	8 (OPERATIONAL)
Delay	4850 ns
Has DC	true (64 bits)
DC Parent	1
DC Active	false, Cyc time: 0 ns, Shft: 0
Parent	6
Config addr	0x1007 (4103)
Station Alias	0
Vndr	0x000001b9 (441)
Product Code	0x00000002 (2)
Rev	0x00000000

Appendix

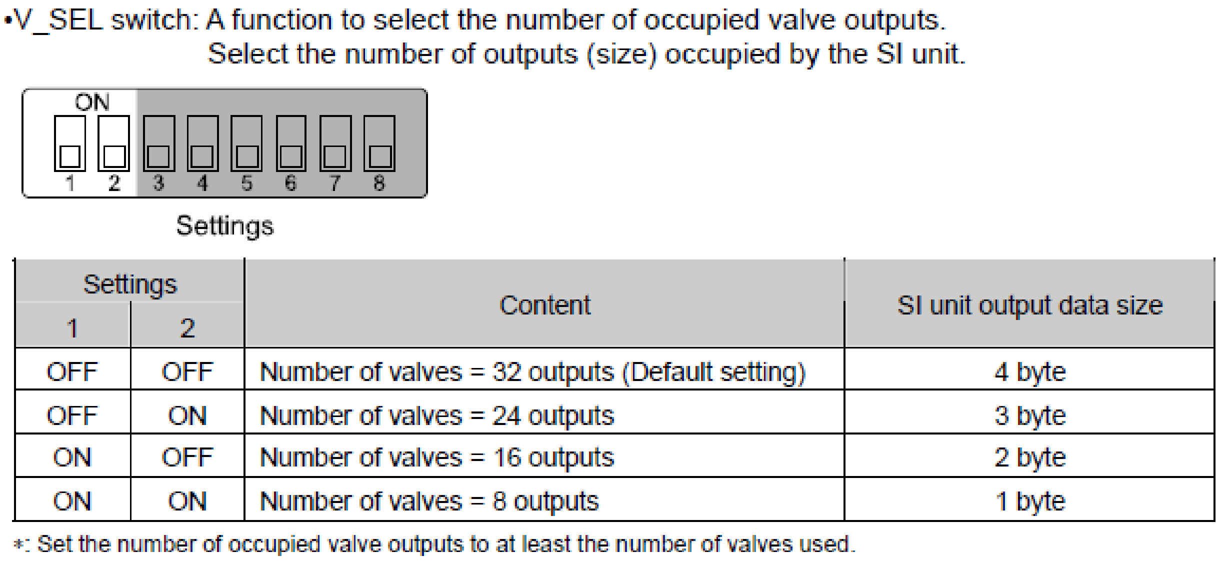
M

[M] SMC Corporation

SMC Corporation manufactures a number solenoid valves and manifolds. That currently supported is the EX600. This section provides information that may be specific to this manufacturer.

EX600 Fieldbus System

The valves module allows for 8 to 32 valves. A dip switch must be set to configure the number of valves present or the wrong number of valves will be reported to EtherCAT and placed in the 5300 Output section.



•V_SEL switch: A function to select the number of occupied valve outputs.
Select the number of outputs (size) occupied by the SI unit.

ON
1
2
3
4
5
6
7
8

Settings

Settings		Content	SI unit output data size
1	2		
OFF	OFF	Number of valves = 32 outputs (Default setting)	4 byte
OFF	ON	Number of valves = 24 outputs	3 byte
ON	OFF	Number of valves = 16 outputs	2 byte
ON	ON	Number of valves = 8 outputs	1 byte

*: Set the number of occupied valve outputs to at least the number of valves used.

Also note that diagnostics are not supported for this device.

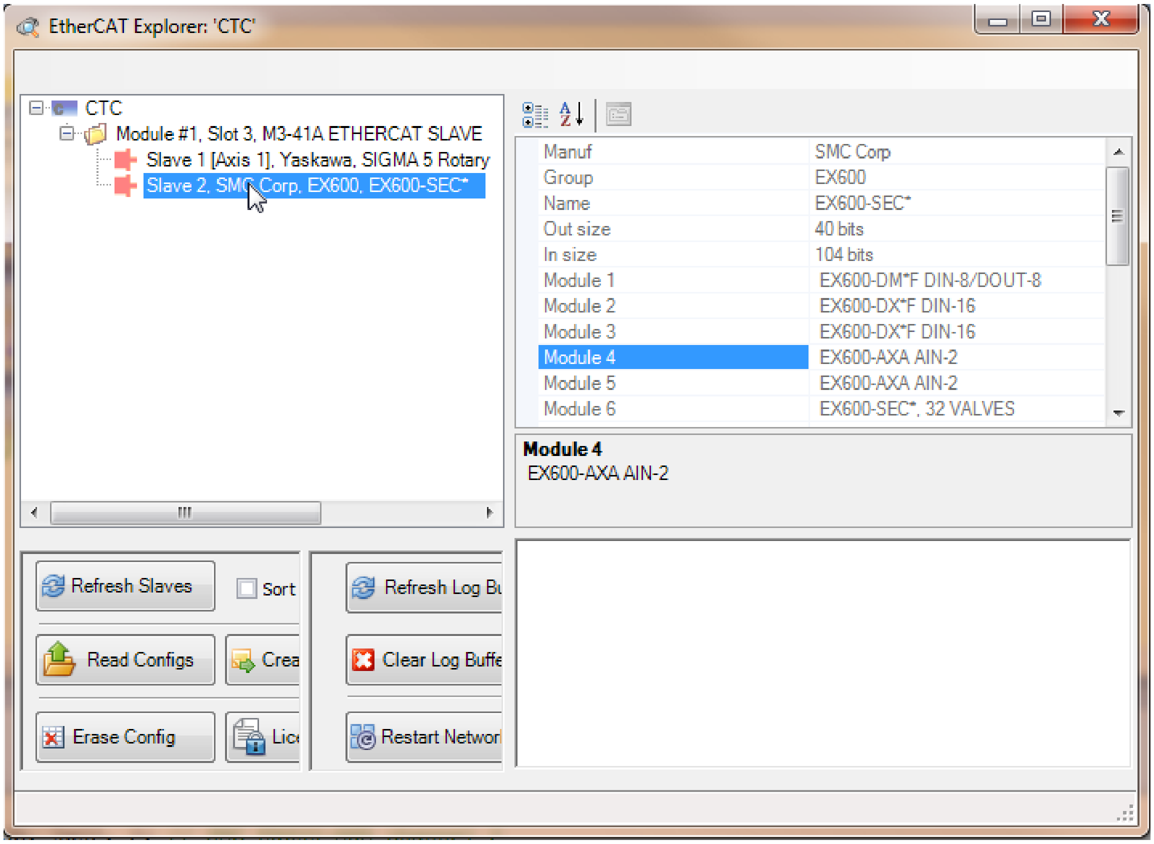
DC Sync

The EX600 reports within the QuickBuilder Explorer that it is capable of operating as a 64 bit distributed clock reference. **Upon testing it has been found that the device does not support the ARMW packet that is required to distribute the clock amongst the slave devices and therefore must not be used as the first device on the network, in a distributed clock environment.**

SDO Configuration

By default, QuickBuilder uses the SMC EX600 in its default configuration. Each module has numerous option settings which can be customized offline using an EtherCAT Configurator, such as Beckhoff's. SMC provides no non-EtherCAT method of configuration. Reference SMC's Operation Manual, EX##-OMO0027 (page 68), for detailed object mapping information. Page 53 describes configuration using the Beckhoff EtherCAT Configurator (simplified version of TwinCAT).

An alternative to using an EtherCAT Configurator would be to write to the individual objects that require customization. This would normally be done once during initialization using the 'sdo write' MSB instruction. Since the EX600 has no dedicated MSB one of the drives MSB's must be used to issue the sdo writes. Prior to doing any sdo writes the EX600 slave address and module slot must be determined:



CTC
Module #1, Slot 3, M3-41A ETHERCAT SLAVE
Slave 1 [Axis 1], Yaskawa, SIGMA 5 Rotary
Slave 2, SMC Corp, EX600, EX600-SEC*

Manuf	SMC Corp
Group	EX600
Name	EX600-SEC*
Out size	40 bits
In size	104 bits
Module 1	EX600-DM*F DIN-8/DOUT-8
Module 2	EX600-DX*F DIN-16
Module 3	EX600-DX*F DIN-16
Module 4	EX600-AXA AIN-2
Module 5	EX600-AXA AIN-2
Module 6	EX600-SEC*, 32 VALVES

Module 4
EX600-AXA AIN-2

Referencing the EtherCAT Explorer image, slave #2 is the SMC unit. In our example we will configure the analog modules (EX600-AXA) for 4-20 ma loop, given the default is +/- 10V. In the EX600 rack modules 4 and 5 are analog input modules, each with 2 channels. According to the SMC Operations Manual, page 83, the EX600-AXA configuration objects (in hex) start at 80x0.00 and end at 80xB.02 where the x is the 'Module # - 1' and the .## is the index. From the manual 80xB is the Analog Input Range setting whose contents are defined as follows:

0 = -10…+10 V
1 = -5…+5 V
2 = -20…+20 mA
3 = 0…10 V
4 = 0…5 V
5 = 1…5 V
6 = 0…20 mA
7 = 4…20 mA

Substituting the Module #, the objects that must be written are 0x803b and 0x804b, for modules 4 and 5 respectively. The default value of 0 must be changed to 7 for 4-20 mA. A sample piece of code appears below which is run on Slave 1, the Yaskawa drive MSB.

```
[beginTest]
// Setup the SMC for 4-20ma, it is slave device 2, slot 4/5 are AXA modules
// according to the EtherCAT Explorer.  According to SMC slot configuration
// address for a module is 0x80#0, zero based. Thus 0x803X & 0x804X are the
// configuration blocks for the EX600-AXA modules.  According to their
// Operation Manual, page 83, range parameter:
// 0x803B.01 = channel 1
// 0x803B.02 = channel 2
// 0x804B.01 = channel 3
// 0x804B.02 = channel 4
// Enumerated values that can be written and their meaning:
//    0=-10…+10 V
//    1=-5…+5 V
//    2=-20…+20 mA
//    3=0…10 V
//    4=0…5 V
//    5=1…5 V
//    6=0…20 mA
//    7=4…20 mA
// A range value of 0 is +/- 10V, 7 is for 4 - 20 ma.  The configuration seems
// to be non-volatile so can also be done using the Beckhoff EtherCAT
// Configurator.  Other parameters such as Monitoring Over/Under range,
// limits, etc., may be of interest and are available in their manual.
//
// Must write to SMC from a running MSB controlling a drive since it
// has no MSB itself.

// Configure all channels for 4-20ma
value = 7;
sdo write value, 2, 0x803b, 0x01, 2; // SMC EX600-AXA channel 1
delay 10ms;
sdo write value, 2, 0x803b, 0x02, 2; // SMC EX600-AXA channel 2
delay 10ms;
sdo write value, 2, 0x804b, 0x01, 2; // SMC EX600-AXA channel 3
delay 10ms;
```

```
sdo write value, 2, 0x804b, 0x02, 2; // SMC EX600-AXA channel 4
delay 10ms;

[run]

// Begin the move, 1 rev/second for 2 revolutions
move at 1 for 2;
wait for in position;
// Delay 1 second once in position
delay 1000 ms;
// Do a relative move back 2 revolutions at 1 rev/second
move at 1 for -2;
wait for in position;
// Delay 1 second once in position
delay 1000 ms;
// Do it again, forever...
goto run;
```

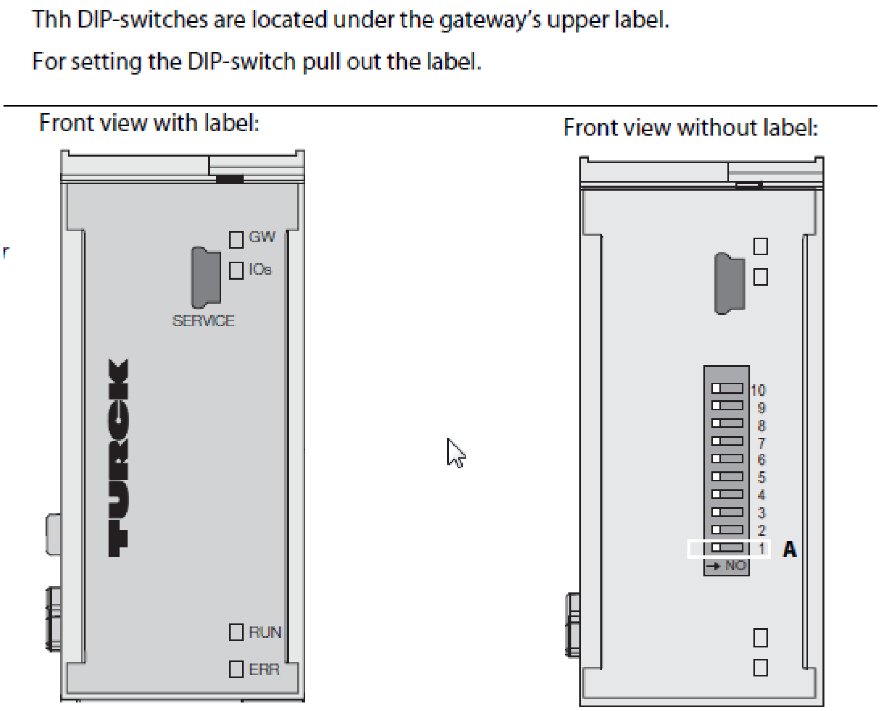
Appendix N

[N] Turck

Turck manufactures an IO controller which can operate as a slave device with the M3-41 EtherCAT network. This section discusses the implementation of the RFID reader option within this controller. Much of the information was derived from the Turck “BLident RFID-S Startup Guide”.

Synchronization via hardware using the CFG-Switch

Prior to operation the Turck BL-20 EtherCAT Gateway must have its detected module configuration saved to its configured module list or an error will result preventing proper online operation. This is done by installing the desired modules, powering the unit, removing its plastic label, and setting dip switch #1 ON.

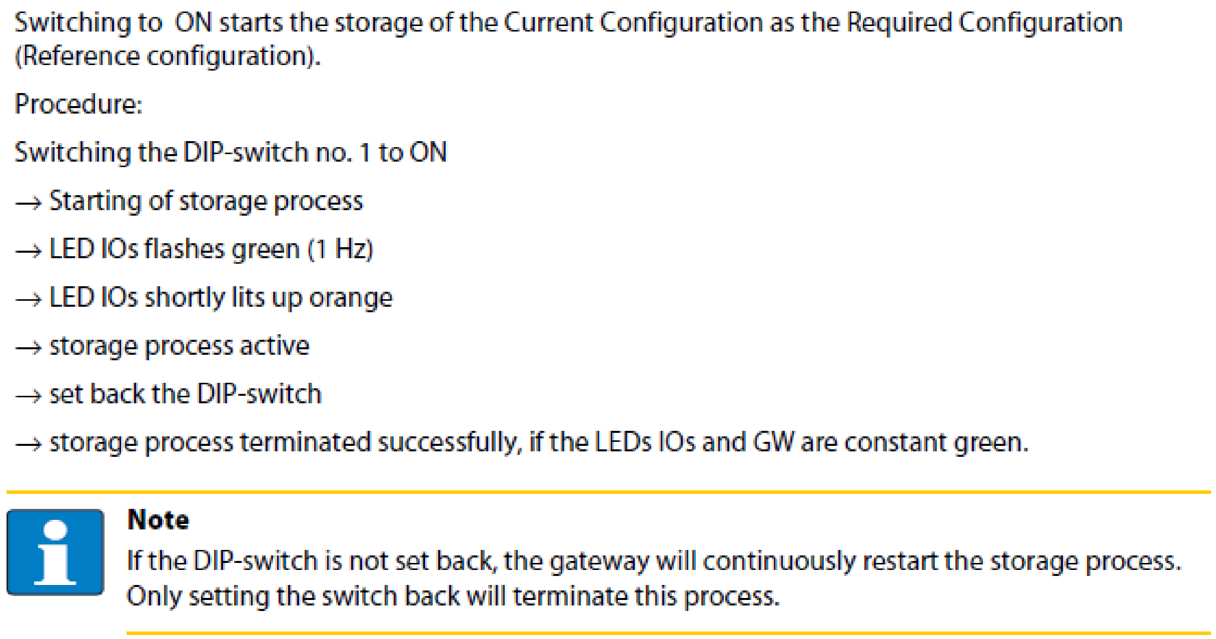


Thh DIP-switches are located under the gateway’s upper label.

For setting the DIP-switch pull out the label.

Front view with label:

Front view without label:



Switching to ON starts the storage of the Current Configuration as the Required Configuration (Reference configuration).

Procedure:

Switching the DIP-switch no. 1 to ON

→ Starting of storage process

→ LED IOs flashes green (1 Hz)

→ LED IOs shortly lits up orange

→ storage process active

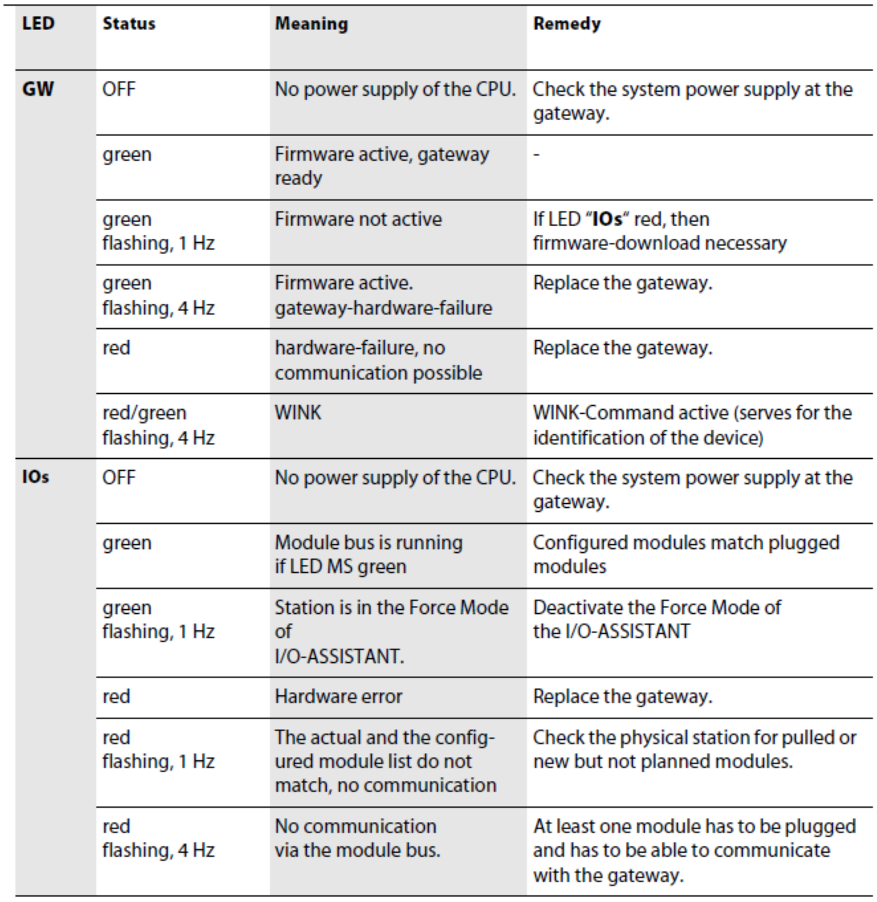
→ set back the DIP-switch

→ storage process terminated successfully, if the LEDs IOs and GW are constant green.

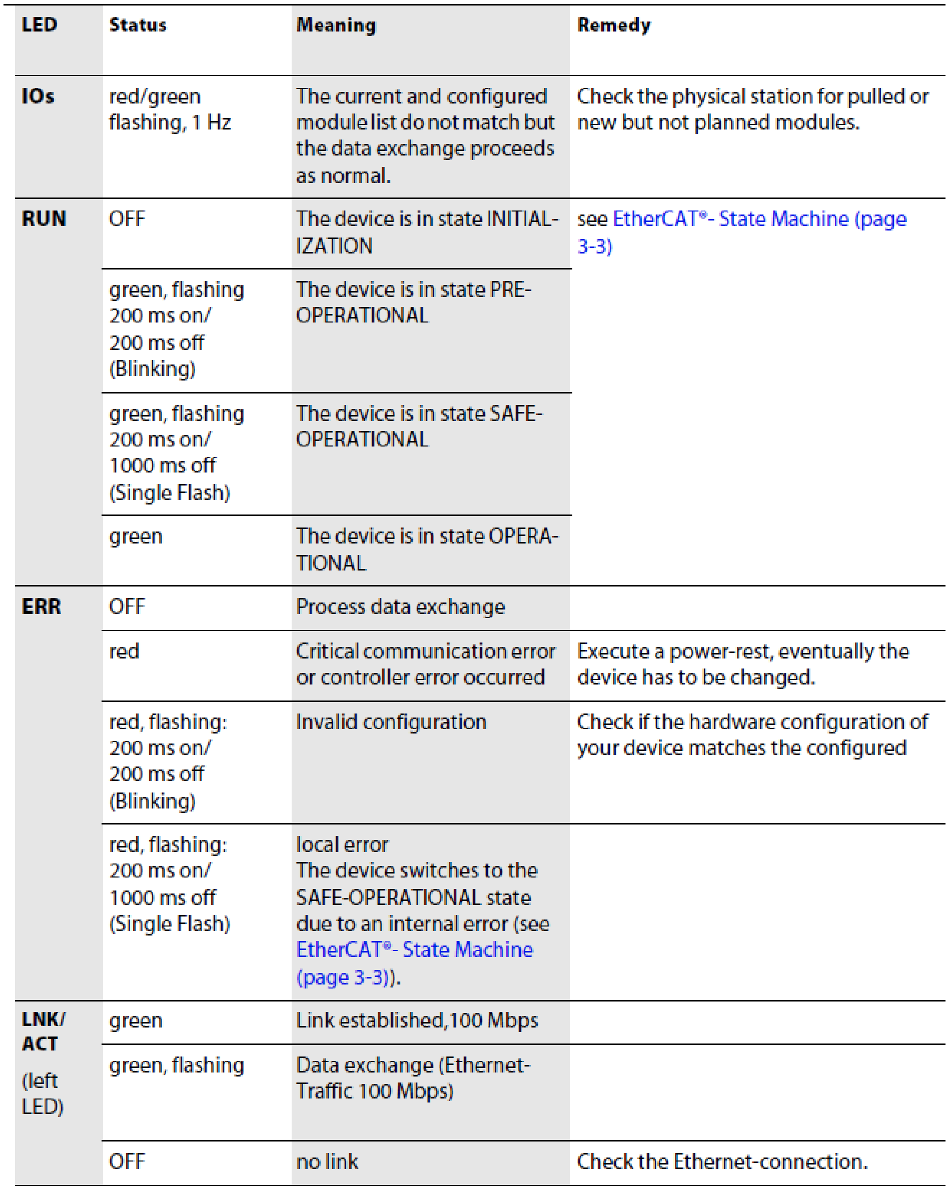
Note

If the DIP-switch is not set back, the gateway will continuously restart the storage process. Only setting the switch back will terminate this process.

Diagnostic LEDs



LED	Status	Meaning	Remedy
GW	OFF	No power supply of the CPU.	Check the system power supply at the gateway.
	green	Firmware active, gateway ready	-
	green flashing, 1 Hz	Firmware not active	If LED "**IOs**" red, then firmware-download necessary
	green flashing, 4 Hz	Firmware active. gateway-hardware-failure	Replace the gateway.
	red	hardware-failure, no communication possible	Replace the gateway.
	red/green flashing, 4 Hz	WINK	WINK-Command active (serves for the identification of the device)
IOs	OFF	No power supply of the CPU.	Check the system power supply at the gateway.
	green	Module bus is running if LED MS green	Configured modules match plugged modules
	green flashing, 1 Hz	Station is in the Force Mode of I/O-ASSISTANT.	Deactivate the Force Mode of the I/O-ASSISTANT
	red	Hardware error	Replace the gateway.
	red flashing, 1 Hz	The actual and the configured module list do not match, no communication	Check the physical station for pulled or new but not planned modules.
	red flashing, 4 Hz	No communication via the module bus.	At least one module has to be plugged and has to be able to communicate with the gateway.



LED	Status	Meaning	Remedy
IOs	red/green flashing, 1 Hz	The current and configured module list do not match but the data exchange proceeds as normal.	Check the physical station for pulled or new but not planned modules.
RUN	OFF	The device is in state INITIAL-IZATION	see EtherCAT®- State Machine (page 3-3)
	green, flashing 200 ms on/ 200 ms off (Blinking)	The device is in state PRE-OPERATIONAL	
	green, flashing 200 ms on/ 1000 ms off (Single Flash)	The device is in state SAFE-OPERATIONAL	
	green	The device is in state OPERA-TIONAL	
ERR	OFF	Process data exchange	
	red	Critical communication error or controller error occurred	Execute a power-rest, eventually the device has to be changed.
	red, flashing: 200 ms on/ 200 ms off (Blinking)	Invalid configuration	Check if the hardware configuration of your device matches the configured
	red, flashing: 200 ms on/ 1000 ms off (Single Flash)	local error The device switches to the SAFE-OPERATIONAL state due to an internal error (see EtherCAT®- State Machine (page 3-3)).	
LNK/ ACT (left LED)	green	Link established,100 Mbps	
	green, flashing	Data exchange (Ethernet-Traffic 100 Mbps)	
	OFF	no link	Check the Ethernet-connection.

EtherCAT Connector

ETH1 is the EtherCAT input connector (closest to the power connector), ETH2 is the output.

RFID

The M3-41 module is capable of supporting 16 channels of RFID. The Turck BL-2RFID-S supports 2 channels per module; therefore up to 8 modules can be supported. The purpose of the RFID interface is to be able to read and write RFID tags at high speed. Any size RFID tag is supported with a read/write burst of 256 bytes available with MSB programming, offset by a modifiable address register.

The interface to the RFID channel consists of a number of properties which are mapped to the reader interface. The M3-41 adds some unique high speed options such as ensuring the tag goes away and becomes present before reading as well as the option to verify that the tag ID (8 byte license plate unique to all tags) is different on the next tag seen, prior to a read or write operation.

When interfacing with the RFID reader transfers are done in groups of up to 8 bytes per access. This is loaded into two integer arrays that are 32 deep (4 bytes per integer X 2 arrays X 32 deep = 256 bytes). Integer arrays are used since MSB's do not support strings. String manipulation can be done by QuickBuilder using a high speed transfer mechanism built into the MSB 'host read' & 'host write' instructions. The 'host read' & 'host write' instructions have direct byte wide access to the integer arrays and can transfer QuickBuilder strings to/from the RFID buffers as needed. MSB's may also manipulate data themselves at the integer level.

RFID Property Variables

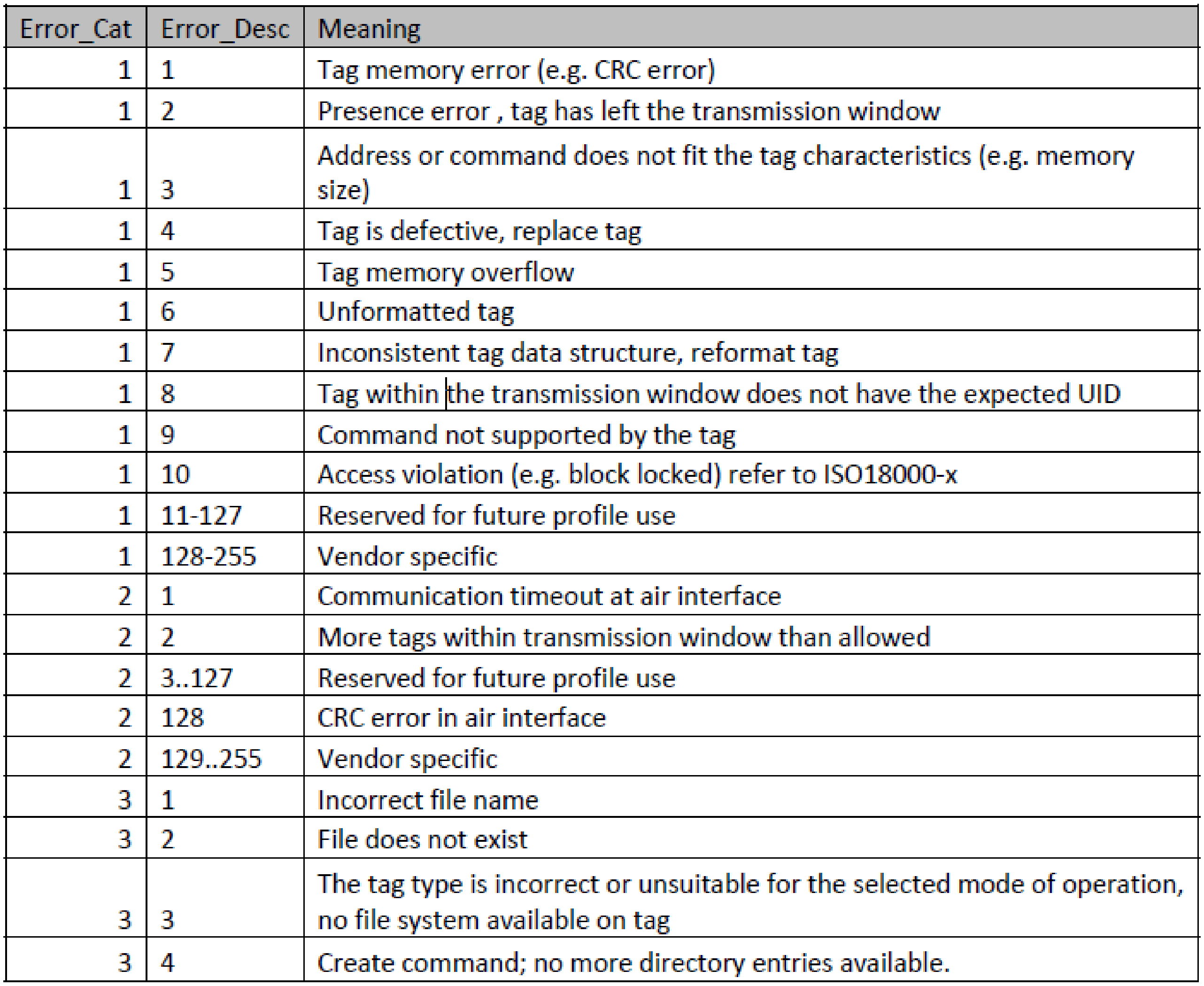
RFID_totalChannels – Read only, represents the total number of RFID channels available in the system.

RFID_channel – Read/write, selects the RFID channel to be operated on by the properties that follow. Entries of 1 to *RFID_totalChannels* are the valid selections, with 0 disabling access. All properties should be initialized to their proper values before setting the RFID_channel to a non-zero value.

RFID_state – Read only, represents the current state of the RFID interface logic state machine as it executes any requests issued by the RFID_control property variable. Possible values are as follows:

RFID_OFF	0
RFID_IDLE	1
RFID_READING_1	2
RFID_READING_2	3
RFID_READING_WAIT_DONE	4
RFID_WRITING_1	10
RFID_WRITING_2	11
RFID_WRITING_3	12
RFID_WRITING_4	13
RFID_WRITE_DONE	15
RFID_WRITE_WAITTAG	20
RFID_READ_WAITTAG	21
RFID_WAITNOTAG	22
RFID_ERROR	30

RFID_error – Read only, Turck specific error where bits 7 to 0 represent the category and bits 15 to 8 are the description. Any time the *RFID_error* property is non-zero an error is present. To clear the error the RFID reader must be reset using the *RFID_control* property RESET bit.



Error_Cat	Error_Desc	Meaning
1	1	Tag memory error (e.g. CRC error)
1	2	Presence error , tag has left the transmission window
1	3	Address or command does not fit the tag characteristics (e.g. memory size)
1	4	Tag is defective, replace tag
1	5	Tag memory overflow
1	6	Unformatted tag
1	7	Inconsistent tag data structure, reformat tag
1	8	Tag within the transmission window does not have the expected UID
1	9	Command not supported by the tag
1	10	Access violation (e.g. block locked) refer to ISO18000-x
1	11-127	Reserved for future profile use
1	128-255	Vendor specific
2	1	Communication timeout at air interface
2	2	More tags within transmission window than allowed
2	3..127	Reserved for future profile use
2	128	CRC error in air interface
2	129..255	Vendor specific
3	1	Incorrect file name
3	2	File does not exist
3	3	The tag type is incorrect or unsuitable for the selected mode of operation, no file system available on tag
3	4	Create command; no more directory entries available.

RFID_address – Read/write, this address is forwarded to the RFID reader to determine where in its memory block to begin accessing data for read and write operations. A value of 0 is the first address. *RFID_address* is auto-incremented after any read or write by the amount in *RFID_bytesTransferred* therefore set it back to the desired start location after each read or write.

RFID_index – Read/write, the index is used to select which RFID_data_readl/RFID_data_readh or RFID_data_writel/RFID_data_writeh array item is to be operated on. Where 0 is the first item, up to 31 (32 array items for 256 bytes total possible).

```
int RFID_data_readl[32];
int RFID_data_readh[32];
int RFID_data_writel[32];
int RFID_data_writeh[32];
```

RFID_data_readl – Read/write, the first 32 bit integer or 4 bytes of data transferred from the RFID tag. This property is an array of 32 deep, indexed by the *RFID_index* property.

```
int RFID_data_readl[32];
```

RFID_data_readh – Read/write, the second 32 bit integer or 4 bytes of data transferred from the RFID tag. This property is an array of 32 deep, indexed by the *RFID_index* property in parallel to *RFID_data_readl*.

```
int RFID_data_readh[32];
```

RFID_data_writel – Read/write, the first 32 bit integer or 4 bytes of data transferred to the RFID tag. This property is an array of 32 deep, indexed by the *RFID_index* property.

```
int RFID_data_writel[32];
```

RFID_data_writeh – Read/write, the second 32 bit integer or 4 bytes of data transferred to the RFID tag. This property is an array of 32 deep, indexed by the *RFID_index* property in parallel to *RFID_data_writel*.

```
int RFID_data_writeh[32];
```

RFID_status – Read/write, 32 bit integer with only the first 8 bits reflecting the status as returned by the Turck RFID reader.

	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	0	Done	Busy	Error	Trans_Conn	Trans_On	TP	TFR	Reserved

- *Done* – Slice is ready to receive command. This bit will be off until previous command bit is turned off.
- *Busy* – Slice is currently processing command. This is normally on when transceiver is waiting for a tag to be presented.
- *Error* – Slice has encountered an error during last command. Refer to Error_Cat and Error_Desc for details. This bit is not always set so check RFID_error for nonzero.
- *Trans_Conn* – Transceiver is correctly connected and communicating with the slice.
- *Trans_On* – Transceiver has been turned on by slice.
- *TP* – Tag present; Tag is present in transceiver field. LED on transceiver will blink rapidly.
- *TFR* – Tag Fully Read; Tag has been present in transceiver field long enough so that entire tag memory has been stored in buffer. This bit does not need to be on to indicate a command has been completed.

RFID_control – Read/write, 32 bit integer which is used to request RFID transactions to occur, read and writing different aspects of the tag and transceiver. Some of the bits are defined by Turck but have been enhanced by CTC for additional features.

<u>From Turck:</u>

	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	0	Transceiver	Next	Tag_ID	Read	Write	Tag_Info	Trans_Info	Reset

- *Transceiver* – Turns on and off transceiver. Used only if two transceivers are close enough to cross talk. Otherwise this bit should be set to an "Always On". If transceiver is off, LED will blink slowly (default), and be solid if transceiver is on. It is suggested to always leave this bit set unless low power operation is required.

- *Next* – If turned on while command is processed, the next command run will require a new tag id to enter the field. This bit should not be used instead reference the upper bits supplied by CTC for similar functionality
- *Tag_ID* – Turn on to read the Unique Identifier (UID) from tag. These are always unique for every tag in the world. The bit is cleared automatically when the operation is complete and data is present or error occurred.
- *Read* – Turn on to read data from a tag. The bit is cleared automatically when the operation is complete and data is present or error occurred.
- *Write* – Turn on to write data to a tag. The bit is cleared automatically when the operation is complete and data is present or error occurred.
- *Tag_Info* – Turn on to read information about tag in field, including tag manufacturer and memory available in tag. The bit is cleared automatically when the operation is complete and data is present or error occurred.
- *Trans_Info* – Turn on to read information about transceiver connected to the channel. It can return data such as type of transceiver, hardware and software revisions. The bit is cleared automatically when the operation is complete and data is present or error occurred.
- *Reset* – Turning this bit on will reset any in-process or queued commands. Use this bit to clear any errors that occur. This bit must be cleared manually to remove the device from reset.

CTC bit enhancements for RFID control property:

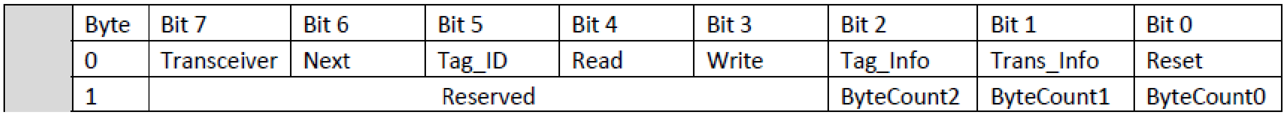
USER_NO_TAGFIRST – Bit 15, set this bit if no tag present is to be detected prior to starting the requested read or write operation.

USER_READ_TAGID_FIRST – Bit 14, set this bit if the unique tag id is to be read prior to the requested read or write operation. This bit automatically sets bit 5, Tag_ID, during operation.

USER_NEW_TAGID – Bit 12, set this bit in conjunction with Bit 14 when the tag id is to be different than that previously read. The tag id is a unique 8 byte value present on all RFID tags. Prevents mistakenly reading and writing the same tag. Prior to any requested read or write the tag id is read and if different from that previously (RFID_lasttagIDl & RFID_lasttagIDh) it is stored to the properties RFID_tagIDl and RFID_tagIDh, the requested read or write operation will then automatically be completed.

RFID_controlActive – Read only, represents the value actually being transferred to the Turck RFID controller at any moment. During operation bits are set/cleared automatically by the M3-41 module, especially when transferring multiple blocks of data. Useful for diagnostic purposes.

RFID_count – Read only, represents the value actually being transferred to the Turck RFID controller as the needed byte count. This property is automatically set based upon the *RFID_bytesTotal* required. Useful for diagnostic purposes.



Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Transceiver	Next	Tag_ID	Read	Write	Tag_Info	Trans_Info	Reset
1	Reserved					ByteCount2	ByteCount1	ByteCount0

- ByteCount – These 3 bits represent the number of bytes to Read or Write
  - 000 = 1 byte
  - 001 = 2 bytes
  - 010 = 3 bytes
  - 011 = 4 bytes
  - 100 = 5 bytes
  - 101 = 6 bytes
  - 110 = 7 bytes
  - 111 = 8 bytes

RFID_bytesTotal – Read/write, this property must be set to the total number of bytes to be transferred to/from the RFID reader. If to the reader RFID_data_writel/h array is used, if from the reader RFID_data_readl/h is used. For a single transfer this is typically set to 8. This property is also used in conjunction with the 'host read' and 'host write' commands when transfers are done with the RFID_data_readl and RFID_data_writel properties. These properties can be used to transfer strings to/from Quickbuilder variants.

RFID_bytesTransferred - Read/write, this property represents the number of bytes that have been transferred during a read/write RFID operation as well as 'host read' and 'host write'. If a 'host read' is used to read a QuickBuilder variant string this property will represent the length of the string after the access.

RFID_tagIDl – Read/write, 32 bit integer, this property is automatically set when a TAG ID read operation is performed. This property represents the lower 4 bytes.

RFID_tagIDh – Read/write, 32 bit integer, this property is automatically set when a TAG ID read operation is performed. This property represents the upper 4 bytes.

RFID_lasttagIDl – Read/write, 32 bit integer, this property is automatically set when a TAG ID read operation is performed and a unique id is found, different than that previously read. This property represents the lower 4 bytes. The RFID_lasttagIDl/h is checked against the latest tag id read to ensure no duplicates are found, if that option is enabled.

RFID_lasttagIDh – Read/write, 32 bit integer, this property is automatically set when a TAG ID read operation is performed and a unique id is found, different than that previously read. This property represents the upper 4 bytes. The RFID_lasttagIDl/h is checked against the latest tag id read to ensure no duplicates are found, if that option is enabled.

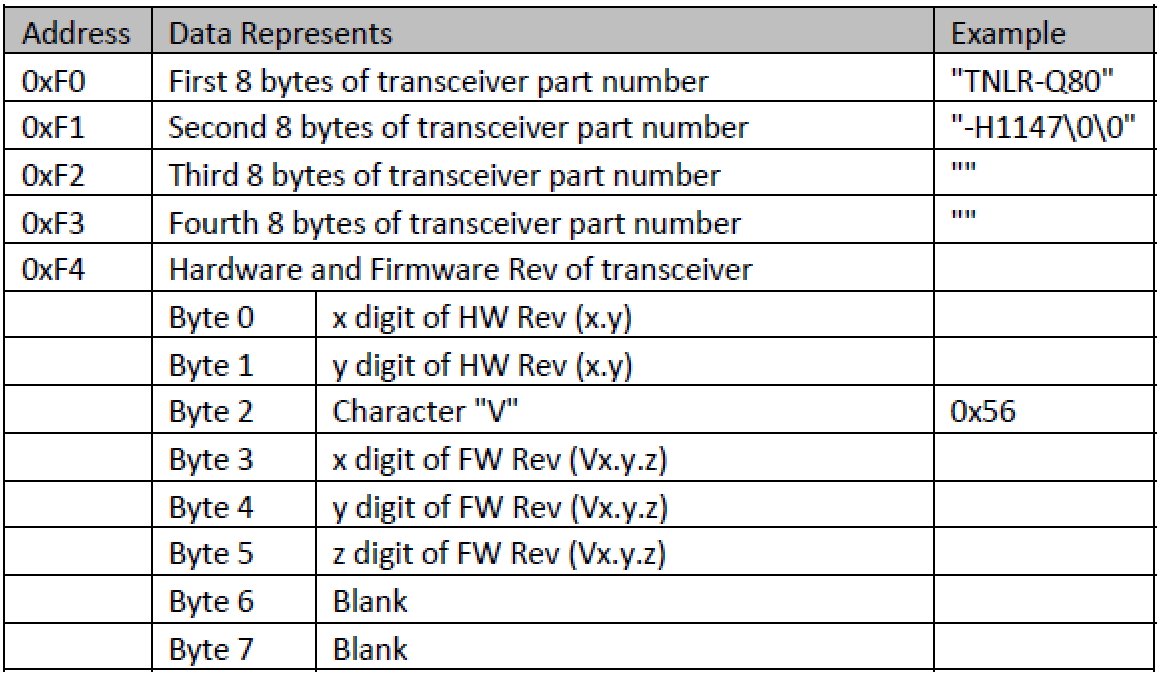
TAG INFO

This command is initiated on the rising edge of the input bit. The command is executed when a tag is present in the interface. The command returns 8 bytes of information to the "Read_Data" bytes.

- Byte 0: Number of Memory blocks
- Byte 1: Number of Bytes per block
- Byte 2: DSFID
- Byte 3: AFI
- Byte 4: ICID
- Bytes 5-7: Always "0"

TRANS INFO

This command is initiated on the rising edge of the input bit. The command is executed immediately. The command returns 8 bytes of information to the *RFID_data_readl/h* array, with the content dependent on the value of *RFID_address*. Below is an example supplied by Turck.



Address	Data Represents		Example
0xF0	First 8 bytes of transceiver part number		"TNLR-Q80"
0xF1	Second 8 bytes of transceiver part number		"-H1147\0\0"
0xF2	Third 8 bytes of transceiver part number		""
0xF3	Fourth 8 bytes of transceiver part number		""
0xF4	Hardware and Firmware Rev of transceiver		
	Byte 0	x digit of HW Rev (x.y)	
	Byte 1	y digit of HW Rev (x.y)	
	Byte 2	Character "V"	0x56
	Byte 3	x digit of FW Rev (Vx.y.z)	
	Byte 4	y digit of FW Rev (Vx.y.z)	
	Byte 5	z digit of FW Rev (Vx.y.z)	
	Byte 6	Blank	
	Byte 7	Blank	

Strings

MSB’s cannot operate directly on strings but QuickBuilder can. Thus to manipulate string data it is first read from the Tag as 1 or more bytes of data, stored in the RFID_data_readl/h data array, and then transferred to QuickBuilder for further processing. This is done using the ‘host write’ command referencing the RFID_data_readl or RFID_data_writel storage locations. RFID_bytesTotal are the number of bytes to transfer.

Assume a tag was read with 128 bytes (up to 223 bytes may be transferred with ‘host read/write’ commands):

```
RFID_bytesTotal = 128;
```

```
host write RFID_data_readl,36103;    // Write the integer array as bytes of a
                                     // string, to QB automatically null
                                     // terminated.
```

QB and MSB handshake as needed as the data is manipulated…

```
RFID_bytesTotal = 0;
host write RFID_data_writel,36103;   // Read the modified string, from QB
                                     // and store to tag write array.
                                     // RFID_bytesTotal is set to size.
```

Programming Examples

```
// TURCK RFID TEST Application

/*
RFID_control (Read/Write)

        Bit 0 - Reset: Turning this on will reset any in-process or queued commands.
        Bit 1 - Trans_Info: Turn on to read information about transceiver connected to the
                channel. It can return data such as type of transceiver, hardware and software
                revisions. See section 2.1.5 for details.
        Bit 2 - Tag_Info: Turn on to read information about tag in field, including tag
                manufacturer and memory available in tag. See section 2.1.4 for details.
        Bit 3 - Write: Turn on to write data to a tag.
        Bit 4 - Read: Turn on to read data from a tag.
        Bit 5 - Tag_ID: Turn on to read the Unique Identifier (UID) from tag. These are always
                unique for every tag in the world.
        Bit 6 - Next: If turned on while command is processed, the next command run will require
                a new tag id to enter the field.  Not used.
        Bit 7 - Transceiver: Turns on and off transceiver. Used only if two transceivers are
                close enough to cross talk. Otherwise this bit should be set to an "Always
                On". If transceiver is off, LED will blink slowly (default), and be solid if
                transceiver is on.
        Bit 12, USER_NEW_TAGID, set this bit in conjunction with Bit 14 when the tag id is
                to be different than that previously read.  The tag id is a unique 8 byte value
                present on all RFID tags.  Prevents mistakenly reading and writing the same tag.
                Prior to any requested read or write the tag id is read and if different from that
                previously (RFID_lasttagIDl & RFID_lasttagIDh) it is stored to the properties
                RFID_tagIDl and RFID_tagIDh, the requested read or write operation will then
                automatically be completed.
        Bit 14, USER_READ_TAGID_FIRST, set this bit if the unique tag id is to be read prior
                to the requested read or write operation.  This bit automatically sets bit 5,
                Tag_ID, during operation.
        Bit 15, USER_NO_TAGFIRST, set this bit if no tag present is to be detected prior to
                starting the requested read or write operation.

RFID_count (Read only)
        // ByteCount: These 3 bits represent the number of bytes to Read or Write per block
        // o 000 = 1 byte
        // o 001 = 2 bytes
        // o 010 = 3 bytes
        // o 011 = 4 bytes
        // o 100 = 5 bytes
        // o 101 = 6 bytes
        // o 110 = 7 bytes
        // o 111 = 8 bytes

RFID_status (Read only)

        Bit 0 - Reserved
        Bit 1 - TFR ? Tag Fully Read; Tag has been present in transceiver field long enough
                so that entire tag memory has been stored in buffer. This bit does not need
                to be on to indicate a command has been completed.
        BIT 2 - TP ? Tag present; Tag is present in transceiver field. LED on transceiver
                will blink rapidly.
        BIT 3 - Trans_On ? Transceiver has been turned on by slice.
        BIT 4 - Trans_Conn ? Transceiver is correctly connected and communicating with the
                slice.
        BIT 5 - Error ? Slice has encountered an error during last command. Refer to
                Error_Cat and Error_Desc for details.
        BIT 6 - Busy ? Slice is currently processing command. This is normally on when
                transceiver is waiting for a tag to be presented.
        BIT 7 - Done ? Slice is ready to receive command. This bit will be off until previous
                command bit is turned off.

RFID_error – Read only, Turck specific error where bits 7 to 0 represent the category
        and bits 15 to 8 are the description.  Any time the RFID_error property is non-zero
        an error is present.  To clear the error the RFID reader must be reset using the
        RFID_control property RESET bit.

// RFID_data_readl - data[0] - data[3]
// RFID_data_readh - data[4]-data[7]

*/

// Example to read QB variant 36102 and then write it back to 36103, row 0, column 0.
RFID_bytesTotal = 0;          // Clear the total bytes to 0, it will be set to actual
                              // size after the read is complete.
// Read the string "This is a test string for RFID." from QB variant 36102
host read RFID_data_writel, 36102;
host write RFID_data_writel,36103;    // Write the string back again, note RFID_bytesTotal was
                                      // set to the total number of bytes to write by prior
                                      // command.  You can verify with Debugger Watch Window.

// Disable RFID control prior to activating a channel
RFID_control = 0;      // Init to nothing, which is default.
RFID_state = 1;        // By default the state is RFID_OFF so set to RFID_IDLE for operation.
RFID_bytesTotal = 8;   // Number of bytes to tranfer each time to/from the tag

// Set address of tag to start transfer on
RFID_address = 0;

RFID_channel = 1;      // Set to first channel, this activates logic and scanning

delay 50 ms;           // Delay a bit to let EtherCAT scan update information to reader.

// Turn Transceiver on
RFID_control = RFID_control | 0x0080;
// Wait for acknowledgement back that both the Transceiver is turned on and connected.
[wait0]
// Trans_On/Trans_Conn bits will be set when transceiver is on.
if ((RFID_status & 0x0018)!=0x0018) goto wait0;

// Reset it the Transceiver
RFID_control = RFID_control | 0x0001;
delay 250 ms;                                // Allow reset to probagate the network
RFID_control = RFID_control & ~0x0001;       // Disable the Reset
delay 250 ms;                                // Allow Transceiver to come out of reset

// Wait until ready to receive a command
[wait1]
if ((RFID_status & 0x0080)==0) goto wait1;

// Read Trans Info for diagnostics reasons, not needed in actual operation.
RFID_control = RFID_control | 0x0002;

// Wait until done with command
[wait2]
if (RFID_control & 0x0002) goto wait2;
if (RFID_error !=0) goto processTransError;          // Ensure no errors

// Load the data for possible later use
TransInfo_data_low = RFID_data_readl;
TransInfo_data_high = RFID_data_readh;

/*--------------------------------------------*/
```

```
/*
        Test below will wait for a tag to be present then read the Tag ID
        when the tag becomes present, making sure the Tag ID is not the
        same as the previous tag.  If it is go back to waiting for a new
        tag, otherwise write an incrementing number.  Once written, read
        the tag back again.
*/

// Monitor for tag available to read
loopCounter = 1;

// ***** WAIT FOR TAG ******
[writeTag]
// Write the counter value when the tag becomes present
RFID_data_writel = loopCounter;
loopCounter = loopCounter + 1;
// ***** WRITE TAG ******
RFID_address = 0;      // Address is auto-incremented so reset to 0
// Set wait for no tag first, then read tag id and make sure not previous before write
RFID_control= RFID_control | 0x8000 | 0x4000 | 0x1000 | 0x0008;

// Wait for write flag to turn off, meaning writing complete, then check for error.
[waitWrite1]
if (RFID_control & 0x0008) goto waitWrite1;
if RFID_error != 0 goto processWriteErr;

// ***** READ TAG ******
// Read data back now
[readTag]
RFID_address = 0;  // Address is auto-incremented so reset to 0
// Read same tag data back so must disable net Tag flags.
RFID_control = (RFID_control & ~(0x8000 | 0x4000 | 0x1000)) | 0x0010;
// Wait for read flag to turn off, meaning reading complete, then check for error.
[waitRead1]
if (RFID_control & 0x0010) goto waitRead1;  // Wait for command to be accepted.
if RFID_error != 0 goto processReadErr;

// Save the read data away for diagnostic use.
Read_data_low = RFID_data_readl;
Read_data_high = RFID_data_readh;
// Go wait for the next tag
goto writeTag;

[processReadErr]
// Issue reset and save error off
readError = readError + 1;    // Bump number of times read error occured.
goto waitErr;       // GO issue reset and wait for error to clear.

[processWriteErr]
writeError = writeError + 1;  // Bump number of times write error occured.

lastError = RFID_error; // Save error code off.
[waitErr]
RFID_control = 0x0001;   // Transceiver on and reset bit set
[waitClr]
delay 10 ms;
if (RFID_error != 0) goto waitClr;
delay 10 ms;            // Allow to probagate over the network.
RFID_control = 0x0080;  // Transceiver on and reset bit clear
delay 10 ms;            // Allow to probagate over the network.
// Wait for error to clear

[waitErr1]
if (RFID_status & 0x0020) goto waitErr;  // Wait for error to go away
if (RFID_status & 0x0040) goto waitErr1;  // Wait for busy to go away
if ((RFID_status & 0x0080) == 0) goto waitErr1;  // Wait for command to be accepted
delay 10 ms;
RFID_state = 1;       // Enable read/write cycles since when error occurs state
                      // machine will hang at RFID_ERROR state for processing.
delay 10 ms;
goto writeTag;       // Go wait for next tag and begin writing again.
```

```
// Enter here if have error during Transceiver Info read
[processTransError]
lastError = RFID_error;
[stall]
goto stall;
```

Blank

Appendix

O

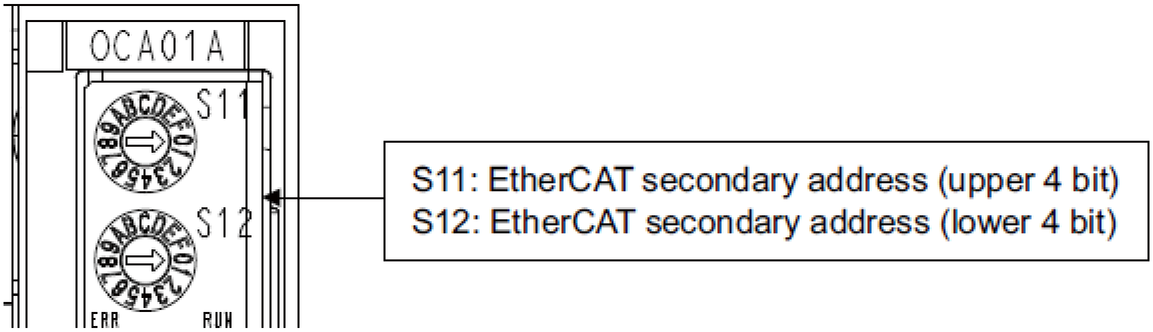
[O] Yaskawa

Yaskawa manufactures a number of drives. That currently supported is the single axis Sigma 5 rotary and linear drive. This section provides information that may be specific to this manufacturer.

eCAT_driveType – 3

Station Alias

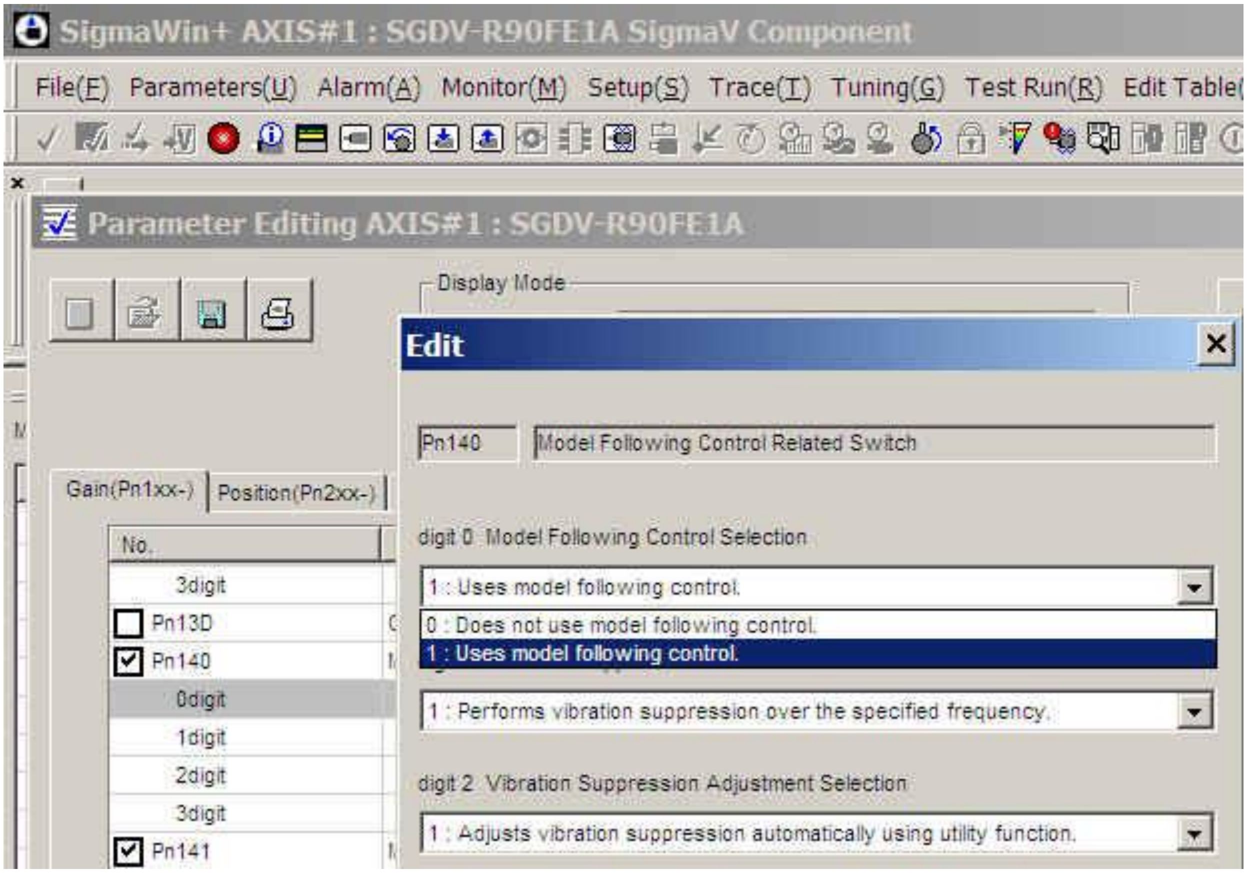
In an EtherCAT network, slaves are automatically assigned addresses based on their position in the bus. When a device, such as a drive, must have a fixed assigned identification that is independent of cabling, a Station Alias is needed. Yaskawa provides two 16-position rotary switches with hexadecimal encoding for this purpose. This allows for a setting of 0 to 255 (FFh), where 0 defaults to the automatic address assignment. As an example, if S11 is set to a 1 and S12 to an A this would be 1Ah or 1 X 16 + 10 = 26. Since the M3-41 only supports up to 16 drives S11 would always be set to 0 and only S12 used.


OCA01A
S11
S12
ERR
RUN
S11: EtherCAT secondary address (upper 4 bit)
S12: EtherCAT secondary address (lower 4 bit)

Yaskawa Position Lag & perr

By default Yaskawa uses what it terms ‘Model Following Control’ in both CSP and interpolated moves. This causes the actual position of the drive to lag from the desired target more than expected (typically 10X that of other drives). This actually provides for a smoother move. Yaskawa delays the move on purpose to better figure out the profile the Master wants to execute and then provides smoothing. When not gearing to dissimilar drives this is a good thing, since at the end of the move you will always be at the correct position, although it takes a bit longer and ‘perr’ will build up.

In some applications, it is desirable to reduce the lag between the commanded position and the actual position reported back by the drive. Do this by disabling ‘Model Following Control’ using SigmaWin. This is parameter Pn140.0 as shown below.


SigmaWin+ AXIS#1 : SGDV-R90FE1A SigmaV Component
File(F) Parameters(U) Alarm(A) Monitor(M) Setup(S) Trace(T) Tuning(G) Test Run(R) Edit Table(
Parameter Editing AXIS#1 : SGDV-R90FE1A
Display Mode
Edit
Gain(Pn1xx-)
Position(Pn2xx-)
No.

3digit
□ Pn13D
☑ Pn140
0digit
1digit
2digit
3digit
☑ Pn141

Pn140 Model Following Control Related Switch

digit 0 Model Following Control Selection

1 : Uses model following control.

0 : Does not use model following control.
1 : Uses model following control.

1 : Performs vibration suppression over the specified frequency.

digit 2 Vibration Suppression Adjustment Selection

1 : Adjusts vibration suppression automatically using utility function.

Note that Pn141, Model Following Gain may also need additional adjustment from its default value.

Drive IO Connector Mapping

The Yaskawa Sigma 5 drive has a number of inputs and outputs available for MSB control on the drive. The following table defines their connector mapping to MSB drive input/outputs.

	Yaskawa	
Name	Pin	MSB Assignment
General Purpose Input	SI0	Din1 (inputs[1])
Forward Run Prohibit Input	SI1	Din2
Reverse Run Prohibit Input	SI2	Din3
General Purpose Input	SI3	Din4
Probe 1 Latch Signal Input	SI4	Din5
Probe 2 Latch Signal Input	SI5	Din6
Home Switch Input	SI6	Din7
Brake Output (option)	SO1	Out1 (outputs[1])
General Purpose Output	SO2	Out2
General Purpose Output	SO3	Out3

MSB 'errorRegister' Variable Value Definitions

Reference the specific drive Manufacture for object 0x1001 definitions. For Yaskawa the definitions are as follows:

Bit 0 – Generic Error, 0: No error, 1: Error
Bit 1 to 7 – Reserved – 0: Always.

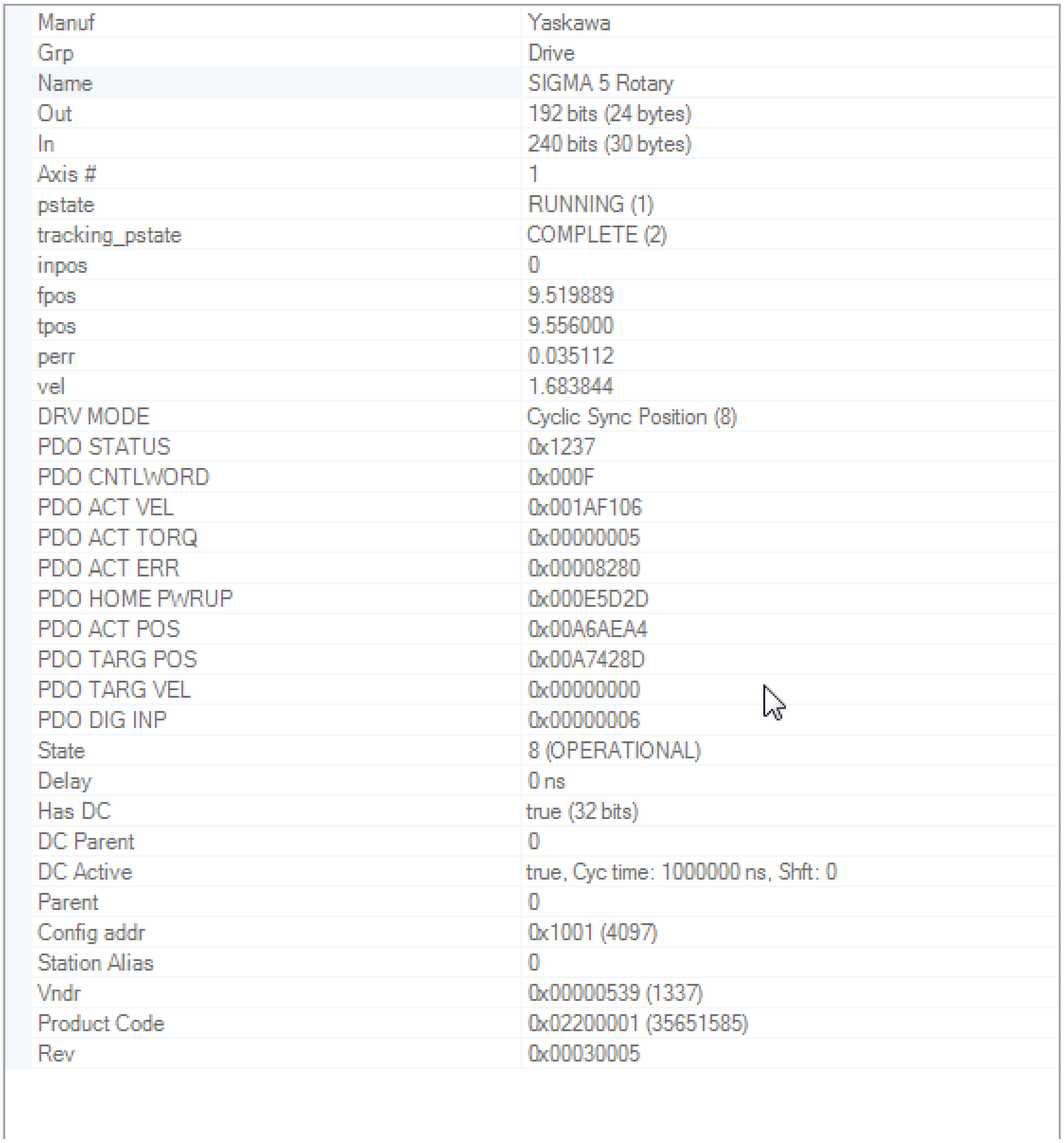
MSB 'errorCode' Variable Value Definitions

Reference the specific drive Manufacture for specific errorCode definitions. For Yaskawa object 0x603f is referenced with values given in hexidecimal:

errorCode	Description
0x0A10	The Sync0 event and the SERVOPACK cannot be synchronized.
0x0A11	The EtherCAT AL state became not 'Operational' while the DS402 drive state is in 'Operation enabled.
0x0A12	The events, receive process data and sync0, do not synchronize. (Failed to receive the process data.)
0x0A20	The parameter setting is out of range.
0x0A40	The initialization at power on sequence was

errorCode	Description
	failed.
0x0EA2	The data exchange between the EtherCAT (CoE) Network Module and the SERVOPACK was not synchronized.
0x0510	The servomotor speed is excessively high.
0x0511	The motor speed upper limit of the set encoder output pulse (pulse unit) (Pn212) is exceeded.
0x910	The motor was operating continuously under a torque largely exceeding ratings.
0x0720	The motor was operating continuously under a torque largely exceeding ratings.
0x710	The motor was operating for several seconds to several tens of seconds under a torque largely exceeding ratings.
0x911	Vibration at the motor speed was detected.
0x520	Vibration at the motor speed was detected.
0x0d00	Position error pulses exceeded parameter (Pn520).
0x0d01	Position error pulses accumulated too much.
0x0d30	Position data exceeded +/- 1879048192.
0x0cc0	Different multi-turn limits have been set in the encoder and the SERVOPACK.
0x0f10	With the main power supply ON, voltage was low for more than 1 second in phase-R, -S or -T.
0x????	Undefined, see manual.

EtherCAT Explorer View



Manuf	Yaskawa
Grp	Drive
Name	SIGMA 5 Rotary
Out	192 bits (24 bytes)
In	240 bits (30 bytes)
Axis #	1
pstate	RUNNING (1)
tracking_pstate	COMPLETE (2)
inpos	0
fpos	9.519889
tpos	9.556000
perr	0.035112
vel	1.683844
DRV MODE	Cyclic Sync Position (8)
PDO STATUS	0x1237
PDO CNTLWORD	0x000F
PDO ACT VEL	0x001AF106
PDO ACT TORQ	0x00000005
PDO ACT ERR	0x00008280
PDO HOME PWRUP	0x000E5D2D
PDO ACT POS	0x00A6AEA4
PDO TARG POS	0x00A7428D
PDO TARG VEL	0x00000000
PDO DIG INP	0x00000006
State	8 (OPERATIONAL)
Delay	0 ns
Has DC	true (32 bits)
DC Parent	0
DC Active	true, Cyc time: 1000000 ns, Shft: 0
Parent	0
Config addr	0x1001 (4097)
Station Alias	0
Vndr	0x00000539 (1337)
Product Code	0x02200001 (35651585)
Rev	0x00030005

Blank

Appendix

P

[P] General Anomalies and Tips

This section is meant to discuss information generic to EtherCAT and aid in the installation.

Network Switches

Network Switches – Network switches should not be used with EtherCAT. Network switches may cause single packet losses. If a switch is required, make sure it is specifically recommended for EtherCAT operation, such as the Omron EtherCAT Junction Slave. Loss of even a single packet may cause a drive error in some drives. This has been noted especially with Kollmorgen devices.

MSBs

MSBs – Do not use foreground MSBs. They typically are not needed and will slow down the control loop. Background MSBs operate outside the control loop and are more efficient. They are supported for legacy M3-40 applications but are not needed with the M3-41 unless limited to a very small MSB.

ppr/mppr

'ppr/mppr' (pulses/rev, and master pulses/rev) must be set properly for the type of encoder used, or improper motion will occur. Set these parameters using QuickBuilder and the properties section of the drive.

Blank

Appendix

Q

[Q] Test Suite

When CTC updates the M3-41 firmware, several tests are used to verify functionality. One such test suite MSB is included below to help users programming the M3-41 module. This test does a quick check of Cyclic Sync Position (Interpolated on some drives), Profile Position, Profile Velocity, and Homing modes. It also does a short segmented and camming table test.

MSB Test Module

```
// This is a background MSB.  Make sure inposw is set for the drive, typically
// .001 for 1048576 ppr.  Also set the ppr and mppr.  This program will set
// the ppr and mppr to a value commonly used, it may have to be changed based
// on the user setup.

// Enable the drive, turning power on to the amplifier.  The current position
// will be constantly updated so the drive does not move.

testCycleCount = 0;
inposw = .001;  // 'in position' window at .001 revolutions.

if eCAT_driveType == $DRIVE_EMERSON goto runDCSYNC;
goto Drive_Enable;
[runDCSYNC]
// This is only needed for Emerson/Control Techniques
delay 2000 ms;  // needed in case restart so syncs when cycle DC Sync on/off.
// Cycle time is 1 ms, start it 100 ms in the future.
// Note that we need to make sure that the first slave device in the EtherCAT
// Network supports 32/64-bit distributed clocks for this to work properly.
// Thus far that is Beckhoff, Wago, and Sanyo Denki.
dcsync -1, 1000000, 0, 0, 100000000;
delay 200 ms;     // starts 100 milliseconds into the future

/**************** ENABLE DRIVE *******************/
[Drive_Enable]
drive enable;
zero feedback position;

// Adjust the ppr and mppr based on the drive/encoder we have installed.
// This overrides that of the property sheet.
```

```
if eCAT_driveType == $DRIVE_COPLEY goto Copley;
if eCAT_driveType == $DRIVE_ELMO goto Elmo;
if eCAT_driveType == $DRIVE_YASKAWA goto Yaskawa;
if eCAT_driveType == $DRIVE_KOLLMORGEN goto Kollmorgen;
if eCAT_driveType == $DRIVE_SANYO_DENKI goto Sanyo_Denki;
if eCAT_driveType == $DRIVE_EMERSON goto Emerson;
if eCAT_driveType == $DRIVE_AMC goto AMC;
if eCAT_driveType == $DRIVE_ABB_MICROFLEX goto ABB;
// by default assume 1048576
[Yaskawa]
[Kollmorgen]
mppr = 1048576;
ppr = 1048576;
goto beginTest;

[ABB]
mppr = 524288;
ppr = 524288;
goto beginTest;

[Copley]
[Elmo]
mppr = 8000;
ppr = 8000;
goto beginTest;

[Sanyo_Denki]
mppr = 131072;
ppr = 131072;
goto beginTest;

[Emerson]
mppr = 65536;
ppr = 65536;
goto beginTest;

[AMC]
mppr = 12000;
ppr = 12000;
goto beginTest;

/**************** BEGIN THE TEST CYCLE *******************/
[beginTest]
state = 0;
host write state,20;    // Set register for CTCMON to monitor
                        // so can visually see where test at

cmode = $CYCLIC_SYNC_POSITION_MODE;       // Ensure in CSP mode
move at 30 to 0 using 50,50;  // If just starting should be at 0,
                              // if redoing program just dropped out of velocity
                              // mode and let's go back to 0.
wait for in position;         // Wait until back at 0.
//delay 3000 ms;        // Delay so can visually check for any position error
/**************** HOMING MODE TEST *******************/
[homing_Mode]
state = 1;
host write state,20;    // Set register for CTCMON to monitor
```

```
                        // so can visually see where test at

inpos_t = 250;    // 250 millisecond settling (can make it anything want
                  // but this is the time the drive will wait at position before
                        // notifying QuickBuilder MSB that it is home.
homing_method = 34;     // Mode setting for home to index pulse
homing_speed1 = 1;      // homing_speed1 is not use but set for default anyways
homing_speed2 = 1;      // in mode 34 only the homing_speed2 is used

cmode = $HOMING_MODE;         // Request Homing mode
move to 0 using 10000,10000;  // Tell the drive to initiate the move with accel
                              // of 10000 rev/sec^2.  That way stop quick.
wait for in position;   // The drive will stop once it see's the index pulse
                  // and tpos = fpos at that position therefore we will have
                  // an offset past home in tpos/fpos, not really absolute 0.

// Using CSP mode we can move back to absolute home position or
// 'zero feedback' to 0 out tpos/fpos. This is needed on some drives.
// Some homing modes 33/34 place you at 0 others just set fpos (feedback
// position) to present offset from it.
cmode = $CYCLIC_SYNC_POSITION_MODE; // Drop back to CSP mode so have control
if tpos <= 1 goto goHome;
// Get in the area first, for most drives offset is small (not Emerson)
move at 10 to .5 using 10,10;
wait for in position;         // Wait until move is complete and we are at .5
[goHome]
move at .1 to 0 using 10000,10000;  // Do any kind of absolute move back to 0 to
                                    // remove possible offset.
wait for in position;   // Wait until move is complete and we are at 0 which
                        // is the index pulse.

/**************** CSP Mode TEST *****************/
[CSP_Mode]
state = 2;
host write state,20;    // Set register for CTCMON to monitor
                        // so can visually see where test at
cmode = $CYCLIC_SYNC_POSITION_MODE; // Request CSP mode
move at 20 for 200 using 10,10;  // 20 rev/s for 200 revolutions with accel at
                                 // 10 rev/s^2 and decel at 10 rev/s^.
wait for in position;
// Do a relative move back
move at 20 for -200 using 10,10;  // 20 rev/s for -200 revolutions with accel at
                                  // 10 rev/s^2 and decel at 10 rev/s^.
wait for in position;
//delay 1000 ms;  // Make sure totally stopped before change modes.
                  // especially if Sanyo Denki since have to disable servo.
/***************** PROFILE MODE TEST *****************/
[profilePositionMode]
state = 3;
host write state,20;    // Set register for CTCMON to monitor
                        // so can visually see where test at
// Place the drive in Profile Position mode (Note Profile Position is not
// supported by Kollmorgen and it will use Interpolated automatically).
cmode = $PROFILE_POSITION_MODE;     // Request Profile Position mode
move at 10 for 50;      // Move at 10 revs/sec for 50 revolutions
// Wait until drive says we are in position
//delay 2 ms;
```

```
wait for in position;
move at 10 for -50;     // Move back again at 10 revs/sec for 50 revolutions
// Wait until drive says we are in position
wait for in position;

/**************** PROFILE VELOCITY MODE  *************/
[velocityMode]
state = 4;
host write state,20;    // Set register for CTCMON to monitor
                        // so can visually see where test at
// Emerson does not support Velocity mode
if (eCAT_driveType == $DRIVE_EMERSON) goto segmentedMode;

invel_t = 1;      // Time required, in milliseconds to be at target velocity
                  // before considered AT TARGET
invel_w = .01;    // Must be at target velocity +/- (.01 X target velocity) with
                  // drive AT TARGET set to satisfy move.  If target is 0, then
                  // becomes +/- this value for velocity.

cmode = $PROFILE_VELOCITY_MODE;     // Request Profile Velocity mode

move at 2 for 1;  // When in velocity mode distance is just sign of direction,
                  // 2 revs/s in positive direction
wait for in position;   // This is when attain requested velocity
delay 10000 ms;         // Run 2 rev/s for 10 seconds

move at 20 for 1;   // Now speed up to 20 rev/sec in the same positive direction
wait for in position;
delay 10000 ms;         // Run 20 rev/s for 10 seconds

move at 0 for 1;        // If the motor is not tuned may never get to here but
wait for in position;   // This is the proper way to stop in vel mode before
                        // moving to another mode.
// The 'in position' is just AT Velocity so now lets switch back to CSP mode
// Slightly different procedure since we are in velocity mode, must
// move to present position but force a switch to CSP mode so don't loose
// position information
cmode = $CYCLIC_SYNC_POSITION_MODE;       // Request CSP mode
delay 5 ms; // Delay is needed for the EtherCAT scanner to see that we want
            // to drop out of velocity mode.  EtherCAT scanner will do a
            // triangular move to present position.  If at velocity of 0
            // no move actually takes place just a switch in the drive mode.
wait for in position;   // Wait for move to finish, we should be at 0 velocity,
                        // if not set acc/dec/vmax, triangular move done

/**************** SEGMENTED MOVE TEST  *************/
[segmentedMode]
state = 5;
host write state,20;    // Set register for CTCMON to monitor
                        // so can visually see where test at

// Initialize the move variables, could also be constants
vel1=5;
vel2=11;
rate1=50;
rate2=5;
dist1=10;
```

```
dist2=60;

stop_dist=1;            // 1 revolution

cmode = $CYCLIC_SYNC_POSITION_MODE;       // Request CSP mode

segmove 1 clear;  // Clear any prior segments stored

// Add each of the desired segments for the move desired
segmove 1 accdec to vel1 using rate1;
segmove 1 slew until dist1;
segmove 1 accdec to vel2 using rate2;
segmove 1 slew until dist2;
segmove 1 accdec to 0 for stop_dist;
// Start the move now
segmove 1 start relative;
// Wait for it to run the segments
wait for in position;


/**************** CAMMING TABLE TEST  *************/
[cammingMode]
state = 6;
host write state,20;    // Set register for CTCMON to monitor
                        // so can visually see where test at

// This MSB effectively sets up a 1:1 gear ratio with the Master
// The first item in the table is master revolutions/second.
// The second entry is that of the slave, this MSB.
table 1 clear;          // Clear out the old data from table 1

table 1 addseries       // Load new data to table 1, could be from a file
                        // but this is a test.
0.000 ,     0.0000      : //set up a 1:1 ratio
1.000 ,     1.0000      :
2.000 ,     3.000 :
3.000 ,     5.0000      :
5.000 ,     0.0000      ; //CAMS should wrap back to zero, like a mechanical cam

table 1 precompute;     // Compute the cam...
// The master can reference another drive with the 'set master feedback1'
// command and setting the master_feedback variable to the drive desired.
// It can also reference its own created master, as shown below.
// This master is virtual and will increment by 1000 counts every
// Control Loop tick (1 ms).  This way only one drive is needed for
// testing camming.
set master virtual;     // We will use our own virtual master

// Set up the number of counts to increment per control loop
// for our virtual master.  This value will be divided by the slave
// ppr for actual master revs.
move master at (40 * ppr/131072) forever;

// Start running the cam table with master and slave scale of 1, do 2 passes.
table 1 start linear cam 1.0, 1.0, 2;
wait for in position;   // Wait for CAM table to be done
activeCAM_row = 0;      // This contains where the table left off when it
```

```
                        // exited and must be zero'd as it is where it will
                        // start next time through.  This allows us to
                        // start at any position in the cam table if nonzero.

/*********** END TEST, REPEAT ***************/
state = 7;        // Done
host write state,20;  // Set register for CTCMON to monitor
testCycleCount = testCycleCount + 1;
delay 1000 ms;          // Start test again in 1 second

goto beginTest;         // repeat the test...
```

Appendix

R

[R] Incentive EtherCAT Master & M3-41A Firmware Revisions

V1.72 (Incentive Only

1. **Active Online/Offline** – Added API interface to allow slave devices to be placed offline, then powered off and replaced, while other parts of the network are fully functional. Once done the slave can then be placed back online without restarting the network.
2. **Incentive Start/Stop** – Added the ability to start and stop the Incentive real-time environment from within the API as well as detecting the state of operation.

V1.71

3. **Code Sync** – Merge of all embedded 5300 source code with embedded PC source code.
4. **Remote Access** – Support for Incentive API remote access via the network.

V1.70

5. **Demo Mode** – Added 3 hour demo mode to EtherCAT Master (PC version only).
6. **Licensing** – Improved licensing scheme to allow for multiple EtherCAT networks, basing the configuration file names on the MAC Address of the EtherCAT network adapter.
7. **Constraint checking** – Improved trajectory calculations with regards to constraining calculations and having trapezoidal reference vmax. Previously only Profile type EtherCAT moves referenced it.
8. **File log** – IO Console (PC version only) is also written to a _log_[MAC Address].txt file by the EtherCAT Master.
9. **Common Var/Bit** – Fixed problem where common bits and vars were not set properly.

V1.67

1. **API Updates** – Added sdo read and write command as well as get_drive_information block property read.
2. **New Position Valid States** – Updated move commands that allow new commanded position during motion (trapezoidal move) to allow more states when valid. There were a

couple of states that only lasted 1 ms that were valid and would cause the command request to be ignored and/or fault.

3. **Mitsubishi Error Reporting** – Attempt now to clear messages from Mitsubishi drive and log alarm states at time of fault.
4. **SDO access** – Added mutex to sdo commands since can only have one active at a time on a drive. Problem occurs at the beginning of a move command and CSP mode is sent to the drive at the same time an application issues an sdo read or write.
5. **Mitsubishi Absolute mode** – Automatically detect Mitsubishi encoder mode and set encoder_mode flag accordingly.

V1.66

1. **Mitsubishi Homing** – Added change to allow warning flag during homing. Occurs in absolute encoder mode.
2. **Mitsubishi** – Fixed problem where input mapping size was wrong.

V1.65

1. **New position** – Enhanced new position command to all change of direction and fully incorporate trapezoidal moves. New variable called 'settling' added which allows the change in direction to be delayed in # of milliseconds.
2. **Mitsubishi Homing** – Updated homing mode for Mitsubishi to work. Tested with homing for index position. Mitsubishi had problem when set homing active it said immediately drive was homed when in fact it was not.
3. **Move to at/for** – Modified trapezoidal move to automatically become a 'new endposition' command when executed while the drive is moving. Also allows the control of both acceleration and deceleration whereas 'new endposition' only specified a rate.

V1.64

1. **set master common** – Corrected problem where having a slave axis track a master axis in camming did not work for mposc. Tracking fposc and tposc did work. Old M3-40A got this information on the backplane between boards. On the PC it is available locally so added functionality to EtherCAT.

V1.63

1. **axisptr/axisnum** – Corrected problem when accessing another drives properties from an MSB 'axisptr' and 'axisnum' also pointed to the other drive. They are not supposed to be mapped. Example in this manual now works properly.

V1.62

1. **Omron** – Added support for Omron NX network IO devices, digital only, analog to follow.

2. **.Net API** – Added full .Net API interface to QuickBuilder MSB language and access to both active program symbols as well as all registers. PC Runtime only.

V1.61

1. **Mitsubishi –** Changed default DC Sync to have 250uS Sync0 shift to accommodate for their possible clock jitter when used as a reference clock. Newer Mitsubishi firmware may resolve the problem in the J4 drive.
2. **New End Position** – Corrected problem with 'newvel' variable that when set less than current velocity the drive would fault. Now the drive changes speed to the 'newvel' regardless of whether it is faster or slower than the current velocity.

V1.60

1. **Sanyo Denki** – Corrected an initialization problem where the drive could generate and output configuration error. Now initialize similar to Mitsubishi and make sure DC Sync is enabled prior to going to the Operational state.
2. **Code Sync –** Synchronized code with Windows® PC Run-time.

V1.59

1. **Drive enable** – Modified 'drive enable' command to now track position when disabled and not restore absolute position when re-enabled. Allows for the drive to be disabled and then moved, for example sliding a table, then re-enabled again while maintaining position.
2. **Yaskawa Power Up** – Resolved a problem that if the drive is in an 'internal limit' state we now wait for it to be removed before attempting to fully enable it and make a move. If 'internal limit' is active the drive will not move and previously the logic would wait forever for the 'inpos' flag. This is typically due to a miss-wired drive or control logic disabled.

V1.58

1. **All IO** – Corrected problem where some configurations of unaligned outputs (non-8 bit boundaries) could cause some outputs to be skipped.

V1.57

1. **Mitsubishi** – Added support for Mitsubishi J4 Drive in CSP mode only. MR-J4-20TM-ECT.
2. **Sanyo Denki** – Corrected problem with tmax not limiting torque.
3. **LinMot –** Corrected issue where downloading a new program caused a short term vibration as the drive held position.

V1.56

1. **LinMot** – Added support for LinMot 1250/1450 linear drives.
2. **PC RunTime -** First source code sync with PC RunTime version of EtherCAT Master.

V1.55

1. **Multiple M3-41A's** – Added support for multiple M3-41A's in a 5300 rack. Problem was found where earlier revisions would operate from boot but would not reset the network properly when a new QuickBuilder program was downloaded.

V1.54

1. **Numatics** – Added support for Numatics IO and valve controller.

Appendix

S

[S] Incentive PCLogic Process

R70.01

1. **Integer non-volatiles** – Corrected problem where integer nonvolatile registers were not being saved properly to disk so that they could be restored after a reboot.

R69.99

1. **Non-Volatile Variants** – Corrected problem within Incentive that the non-volatile variant tables were not saved to disk properly for restore after a reboot.
2. **Non-Volatile Variants** – File format of Incentive was not the same as the 5300 due to the fact that PC's store doubles opposite of the ARM processor. Problem corrected. 5300 _nvar files can be copied to Incentive but Incentive cannot be copied back.