QEC ETG.7200 Test Tool

User Manual

Version 1.6 | April 2026

ICOP Technology Inc.

Revision History

Version	Date	Description of Changes
1.6	April 2026	Added Chapter 12 hands-on tutorials with five scenarios (12.1 identification / 12.2 PDO / 12.3 DC / 12.4 PP mode / 12.5 CSP mode), including device-screen photos; comprehensively corrected image styles and list numbering.
1.2	April 2026	Initial external release. Complete documentation of nine tabs with UI screenshot illustrations.
1.0	2025	Internal engineering version.

Table of Contents

(Memo page)

1. Overview

The ETG.7200 Test Tool is an EtherCAT device test application deployed on the QEC platform (86Duino + 86HMI). It provides a graphical tab-based operation interface for debugging, diagnosing, and functionally validating EtherCAT SubDevices in accordance with the ETG.7200 test specification.

The interface provides nine tabs across the top: Information, State Machine, Process Data, CoE, SII EEPROM, ESC Registers, DC (Distributed Clocks), Online (I/O), and CiA 402 (Servo Drive).

All tabs share a common global toolbar that keeps critical parameters synchronized across pages.

1.1 Global Toolbar

The following controls are visible on every tab. Changes made on any page are immediately reflected on all other pages.

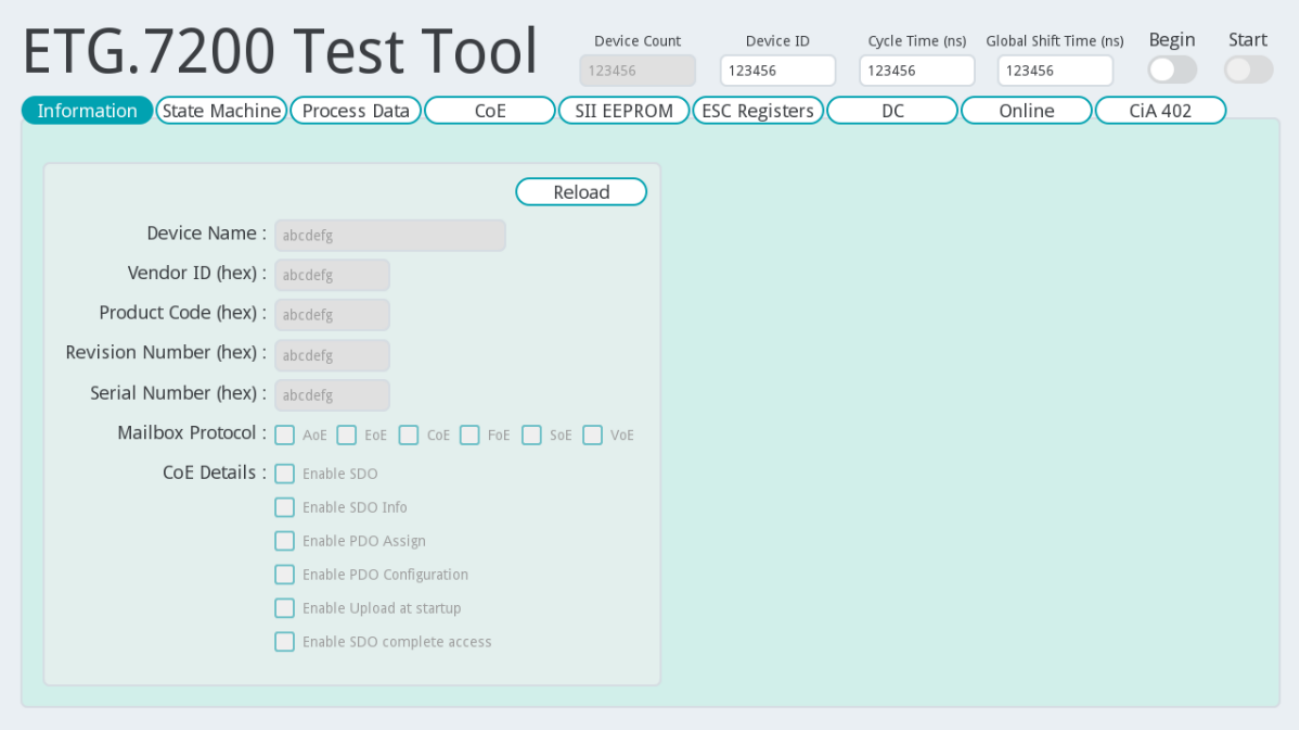
Control	Description
Device Count	Read-only. Number of SubDevices detected after Begin is enabled.
Device ID	Selects the target SubDevice for the current operation (zero-based). Range: 0 to Device Count − 1.
Cycle Time (ns) (Cycle Time)	Communication cycle of the EtherCAT MainDevice, in nanoseconds. Default: 1,000,000 (1 ms). Cannot be modified after Start is enabled.
Global Shift Time (ns) (Global Shift Time)	DC shift time offset, in nanoseconds. When set to 0, the MainDevice automatically calculates a suitable shift value after Start.
Begin (toggle switch)	Toggle ON: calls `master.begin()` to initialize the EtherCAT network. Toggle OFF: calls `master.end()` to release the network. This switch is locked while Start is enabled.
Start (toggle switch)	Toggle ON: calls `master.start()` to begin cyclic PDO communication. Begin must be enabled first. This switch is locked while Bootstrap mode is active.

1.2 Startup Procedure

1. Power on the QEC hardware and wait for the HMI to finish booting.
2. Set Cycle Time and Global Shift Time as needed.
3. Toggle Begin to ON; the system automatically updates Device Count.
4. Set Device ID to select the target SubDevice; starts from 0.
5. Toggle Start to ON to begin cyclic communication.
6. Switch to the desired tab and perform the corresponding test operation.

> **Note** *PDO Mapping and DC settings must be performed while Begin is enabled and Start is disabled.*

2. Information Tab



[Figure: screenshot — Figure 1 — Information tab]

The Information tab reads and displays the identification information and capability description of the selected SubDevice. All display fields are read-only and update only when the Reload button is clicked.

2.1 Device Identification

Field	Description
Device Name	ASCII name string read from the SubDevice object dictionary.
Vendor ID (hex)	32-bit vendor identifier (e.g., 00000002).
Product Code (hex)	32-bit product code.
Revision Number (hex)	32-bit hardware/firmware revision number.
Serial Number (hex)	32-bit serial number.

2.2 Mailbox Protocol

Six read-only checkboxes show the mailbox protocols declared as supported in the SubDevice SII EEPROM.

Checkbox	Protocol Description
AoE	ADS over EtherCAT
EoE	Ethernet over EtherCAT
CoE	CANopen over EtherCAT
FoE	File Access over EtherCAT
SoE	Servo Drive Profile over EtherCAT
VoE	Vendor-specific over EtherCAT

2.3 CoE Details

Six read-only checkboxes reflect the CoE Details byte and indicate the CoE services supported by the SubDevice.

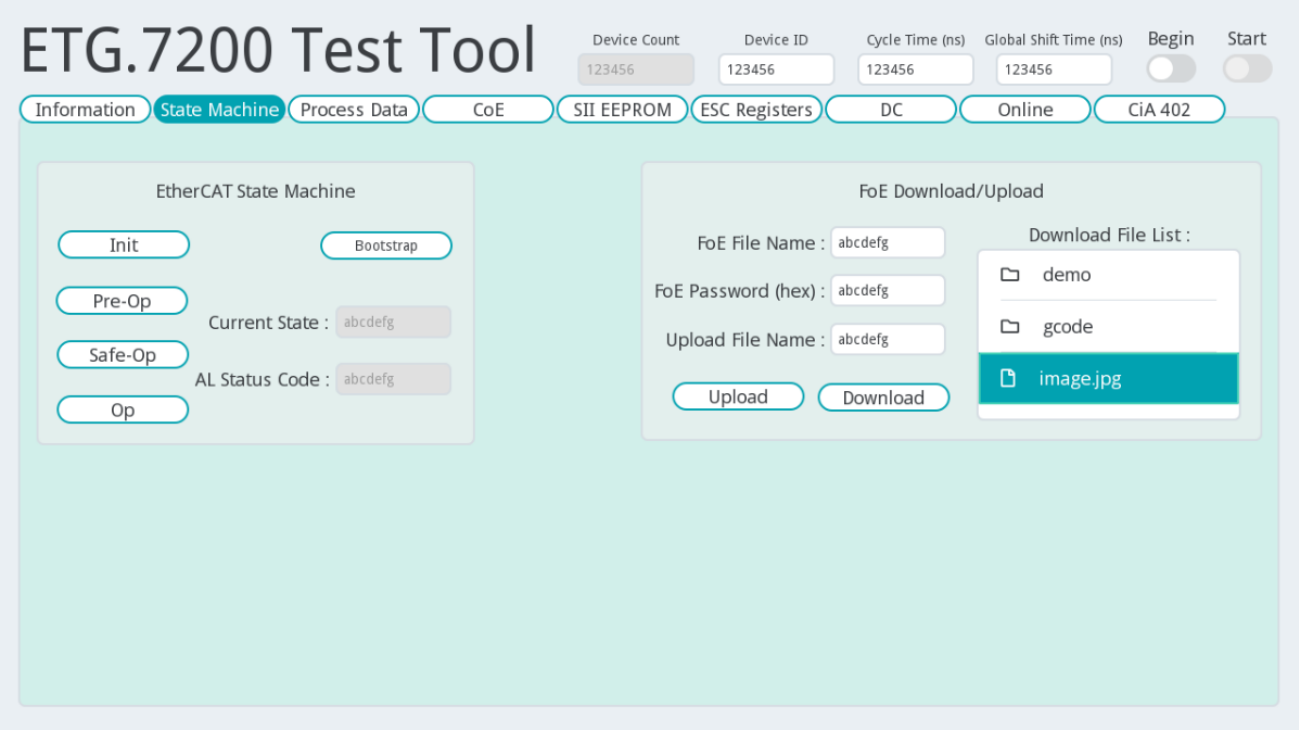
Checkbox	Meaning
Enable SDO	SDO communication supported.
Enable SDO Info	SDO Information service supported (required for OD browsing).
Enable PDO Assign	PDO Assignment via SDO supported.
Enable PDO Configuration	PDO Mapping configuration via SDO supported.
Enable Upload at startup	Device automatically uploads PDO configuration on startup.
Enable SDO complete access	SDO Complete Access transfer supported.

2.4 Reload Button

Clicking Reload queries the currently selected SubDevice and refreshes the values displayed in all fields above.

Begin must be enabled before this operation.

3. State Machine Tab



[Figure: screenshot — Figure 2 — State Machine tab]

The State Machine tab contains two independent function blocks: the EtherCAT State Machine (ESM) control on the left, and FoE firmware transfer on the right.

3.1 EtherCAT State Machine

This block transitions the selected SubDevice to any valid EtherCAT state and shows the current state in real time.

Control / Field	Description
Init button	Command the SubDevice to transition to Init state.
Pre-Op button	Command the SubDevice to transition to Pre-Operational state.
Safe-Op button	Command the SubDevice to transition to Safe-Operational state. Start must be enabled first.
Op button	Command the SubDevice to transition to Operational state. Start must be enabled first.
Bootstrap button	Command the SubDevice to enter Bootstrap state. While Bootstrap is active, the Start switch is locked to prevent accidental PDO communication.
Current State (read-only)	Displays the SubDevice's current AL state: Init, Pre-Op, Safe-Op, Op, or Bootstrap.
AL Status Code (AL Status Code, read-only)	4-digit hexadecimal AL status code. A non-zero value indicates an error on the SubDevice.

> **Note** *Safe-Op and Op buttons are operable only when both Begin and Start are enabled. The Start switch is disabled in Bootstrap mode to prevent stray PDO communication during firmware updates.*

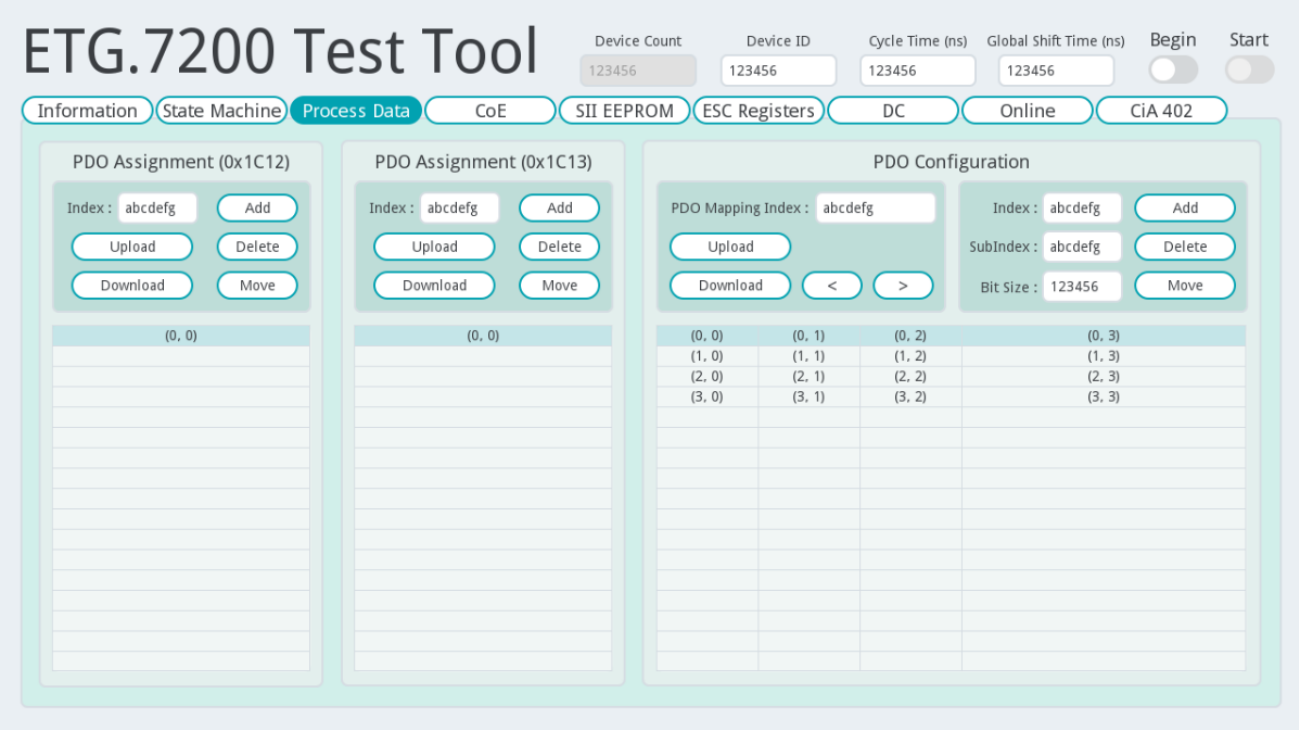
3.2 FoE Firmware Download / Upload

This block transfers firmware files between the QEC USB storage device and the SubDevice using the File Access over EtherCAT (FoE) protocol. The SubDevice must be transitioned to Bootstrap state before transfer.

Field / Control	Description
Download File List (Download File List)	File browser for the USB drive, displaying folders and files. Select a target firmware file and then click Download.
FoE File Name (FoE File Name)	File name sent to (or requested from) the SubDevice. Must match the name expected by the SubDevice bootloader (e.g., app.efw).
FoE Password (FoE Password, hex)	8-digit hexadecimal FoE access password. Enter the value specified by the SubDevice vendor. Default: 00000000.
Upload File Name (Upload File Name)	File name on the USB to which data fetched from the SubDevice is saved (e.g., upload.efw).
Download button	Reads the file selected in Download File List and sends it to the SubDevice via FoE Write. On completion the result is shown in a message box.
Upload button	Reads firmware from the SubDevice via FoE Read and saves it to the USB at the Upload File Name path. Maximum transfer size: 2 MB.

> **Note** *File transfers may take several seconds. Do not toggle the Begin or Start switches during transfer.*

4. Process Data Tab



[Figure: screenshot — Figure 3 — Process Data tab]

The Process Data tab is divided left-to-right into three blocks: PDO Assignment 0x1C12 (RxPDO assignment), PDO Assignment 0x1C13 (TxPDO assignment), and PDO Configuration (PDO mapping). PDO operations require Begin to be enabled and Start to be disabled.

> **Note** *PDO settings must be configured in Pre-Operational state. Write operations are disabled after Start is enabled.*

4.1 PDO Assignment 0x1C12 (RxPDO Assignment)

Specifies which RxPDO mapping objects (index range 0x1600–0x17FF) are assigned to the SubDevice receive channel.

Control	Description
Index field	Enter the RxPDO mapping index to add or remove (hex, e.g., 1600).
Upload button	Reads the current 0x1C12 assignment list from the SubDevice and displays it in the table below.
Download button	Writes the edited assignment list back to the SubDevice's 0x1C12 object. Requires Start to be disabled.
Add button	Appends the index value from the Index field to the end of the assignment list.
Delete button	Removes the index value in the Index field from the assignment list.
Move button	Moves the matching entry up by one position.
Assignment list table	Displays up to 16 assigned RxPDO indices.

4.2 PDO Assignment 0x1C13 (TxPDO Assignment)

Operation is identical to Section 4.1, but applies to the transmit channel (0x1C13). Valid index range: 0x1A00–0x1BFF.

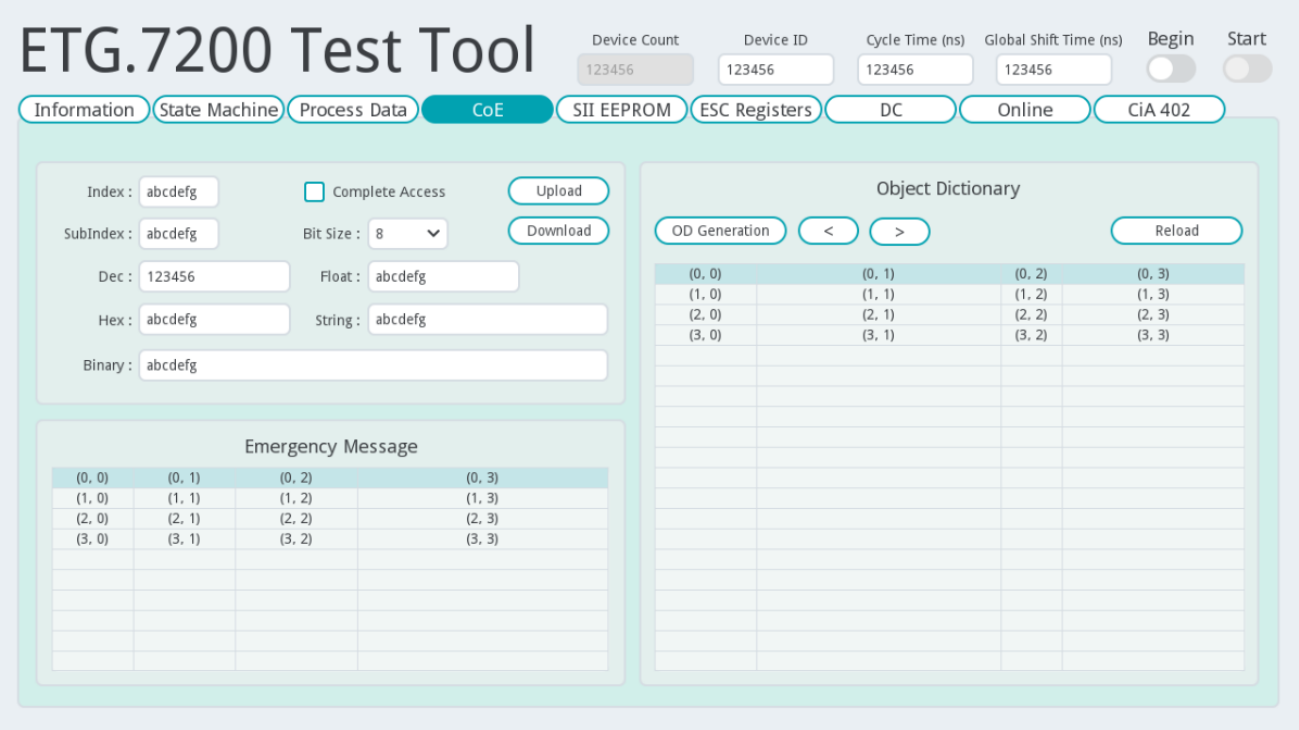
Control	Description
Index field	Enter the TxPDO mapping index to add or remove (hex, e.g., 1A00).
Upload / Download	Reads from or writes to the SubDevice's 0x1C13 object.
Add / Delete / Move	Operation is identical to Section 4.1, but applies to the TxPDO list.
Assignment list table	Displays up to 16 assigned TxPDO indices.

4.3 PDO Configuration

Reads and edits the mapping entries of the user-selected PDO object.

Control / Field	Description
PDO Mapping Index (PDO Mapping Index)	Enter the PDO mapping object index to configure (e.g., 1600 for the first RxPDO, 1A00 for the first TxPDO).
Upload button	Reads the current mapping configuration from the SubDevice via SDO and fills the mapping table.
Download button	Writes the mapping table contents back to the SubDevice. Requires Start to be disabled.
< and > buttons	Used to scroll forward and backward when mapping entries exceed the visible row count.
Mapping table	Displays Index, SubIndex, and Bit Size for each mapping entry in tabular form.
Index field	Object index to be mapped (hex, e.g., 6040).
SubIndex field	Sub-index of the object to be mapped (hex, e.g., 00).
Bit Size field	Bit width of the mapped data (e.g., 16).
Add button	Appends the specified Index / SubIndex / Bit Size entry to the end of the mapping table.
Delete button	Removes entries matching Index / SubIndex / Bit Size from the mapping table.
Move button	Moves the matching entry up by one position.

5. CoE Tab



[Figure: screenshot — Figure 4 — CoE tab]

The CoE tab contains two side-by-side blocks: a direct SDO access panel on the left and an Object Dictionary (OD) browser on the right. An Emergency Message log table sits at the lower left.

5.1 SDO Upload / Download

Performs a single SDO Upload (read) or SDO Download (write) on any specified object on the SubDevice.

Only one field function can be executed at a time.

Field / Control	Description
Index	4-digit hexadecimal object index (e.g., 6040).
SubIndex	2-digit hexadecimal sub-index (e.g., 00).
Bit Size (Bit Size, dropdown)	Data width: 8, 16, 32, or 64 bits. Disabled when Complete Access is selected.
Complete Access checkbox	When checked, transfers the entire object (all sub-indices) in a single SDO operation. The SubIndex and Bit Size controls are disabled simultaneously.
Dec (decimal)	Signed decimal representation of the data. Modifying this field clears the Hex, Float, String, and Binary fields.
Hex	Hexadecimal representation of the data. Takes precedence over the Dec field when modified together.
Float	Floating-point representation of the data.
String	ASCII string representation of the data.

Binary (raw bytes)	Raw byte dump filled by Upload; can also be edited manually as input data for Download.
Upload button	Reads the object from the SubDevice and fills all value fields with the result.
Download button	Writes the current value field contents to the SubDevice.

5.2 Object Dictionary Browser

Enumerates all accessible objects on the SubDevice via the SDO Info service, displaying Index, Name, Flags (access flags), and Value in a table.

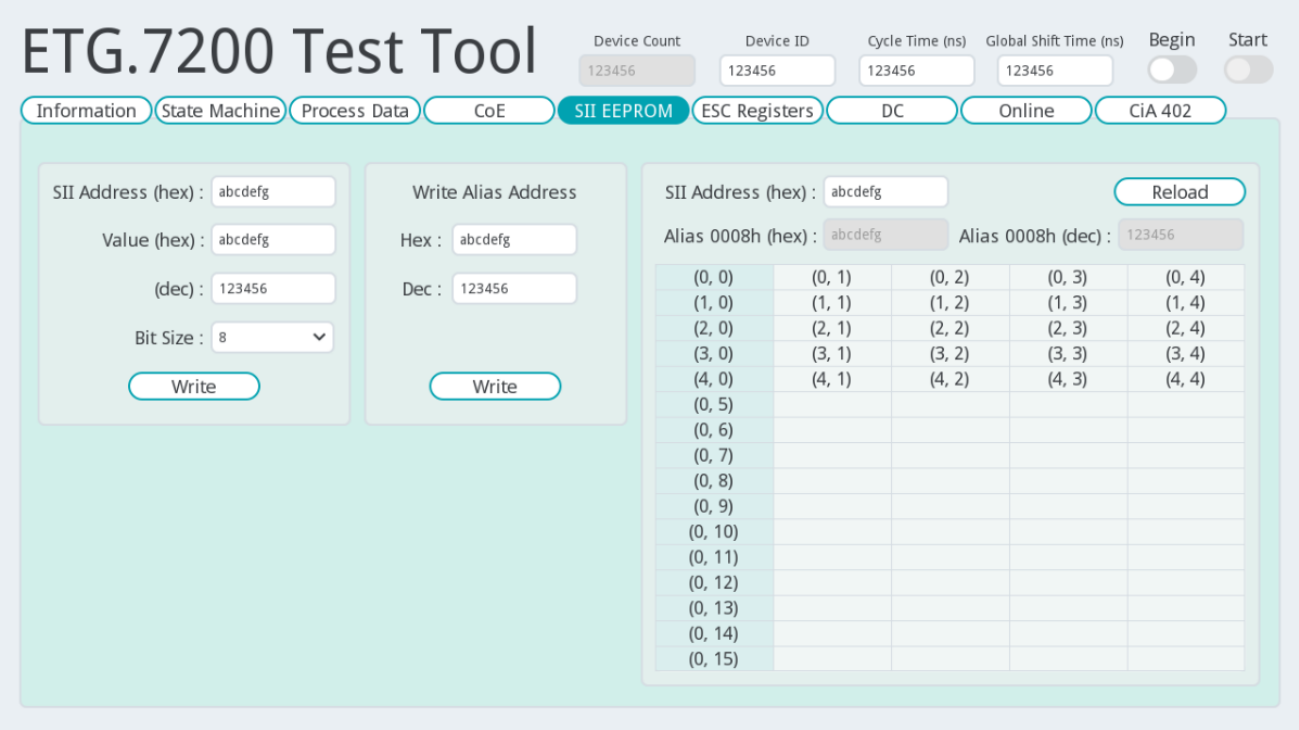
Control / Field	Description
OD Generation button	Queries the complete object dictionary list from the SubDevice and resets the view to the first page.
< and > buttons	19 entries per page; used for pagination.
Reload button	Performs SDO Upload on all currently visible readable objects, refreshing values in the Value column.
Index column	Object index and sub-index, format XXXX:XX (hex).
Name column	Object or entry name string reported by the SubDevice.
Flags column	Access flags: RO (read-only), WO (write-only), or RW (read-write).
Value column	Current value after Reload, formatted by data type (decimal, hex, floating-point, or string).

5.3 Emergency Message Log

Captures CoE Emergency messages received from any SubDevice during communication. A ring buffer stores up to 9 entries; when full, the oldest message is overwritten.

Field	Description
Column 0	Device ID of the SubDevice that sent the emergency message.
Column 1	16-bit Emergency Error Code.
Column 2	8-bit Error Register value.
Column 3	5 bytes of vendor-specific error data (hex).

6. SII EEPROM Tab



[Figure: screenshot — Figure 5 — SII EEPROM tab]

The SII EEPROM tab is divided into three blocks: a direct write panel on the left, a Station Alias write panel in the middle, and an EEPROM read/display panel on the right.

> **Note** *Writing to the SII EEPROM is an advanced feature. Verify all values before writing.*

6.1 Direct Write Panel

Field / Control	Description
SII Address (hex)	Starting word offset address for the write operation.
Value (hex)	Hexadecimal data to write to the specified address.
(dec) (decimal)	Decimal representation of the write value, synchronized with the Hex field.
Bit Size (dropdown)	Write data width: 8, 16, 32, or 64 bits.
Write button	Writes the specified value to the corresponding SII EEPROM address.

> **Note** *Writing incorrect data to the SII EEPROM can render the SubDevice unresponsive. Verify all values before writing.*

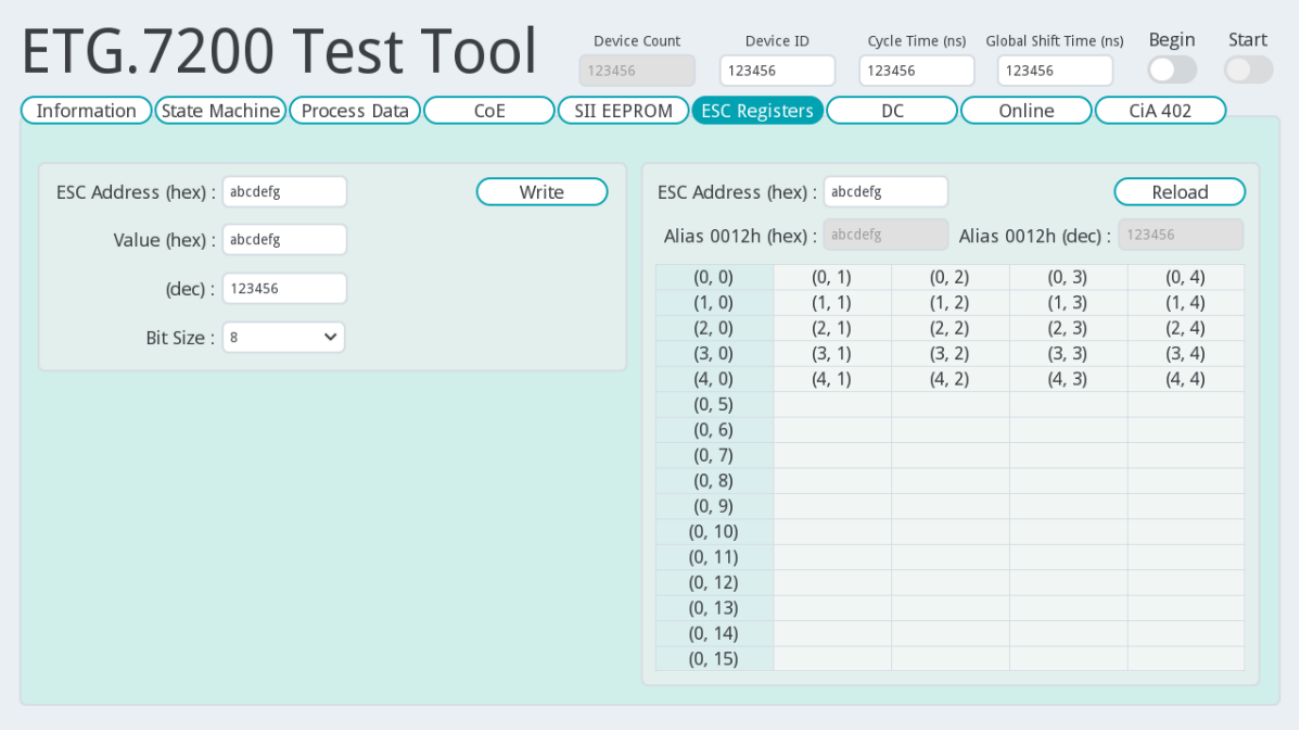
6.2 Write Alias Address Panel

Field / Control	Description
Hex field	Enter the station alias value in hex.
Dec field	Enter the station alias value in decimal (synchronized with the Hex field).
Write button	Writes the station alias to the SII EEPROM. The SubDevice should be in Init or Pre-Op state before this operation.

6.3 EEPROM Read / Display Panel

Field / Control	Description
SII Address (hex)	Starting address for read display.
Reload button	Reads the EEPROM contents starting at the specified address and fills the table below.
Alias 0008h (hex, read-only)	After Reload, shows the contents of EEPROM word 0x0008 (Station Alias).
Alias 0008h (decimal, read-only)	Decimal representation of the alias value.
Data table	Displays raw EEPROM contents in tabular form.

7. ESC Registers Tab



[Figure: screenshot — Figure 6 — ESC Registers tab]

The ESC Registers tab provides low-level read/write access to the EtherCAT Slave Controller (ESC) register space. The layout is identical to the SII EEPROM tab: write panel on the left, read/display panel on the right.

> **Note** *ESC register writes take effect immediately and cannot be undone — this is an advanced feature. Writing to reserved addresses or critical-function registers (e.g., AL Control 0x0120) may unexpectedly change the device state.*

7.1 Write Panel

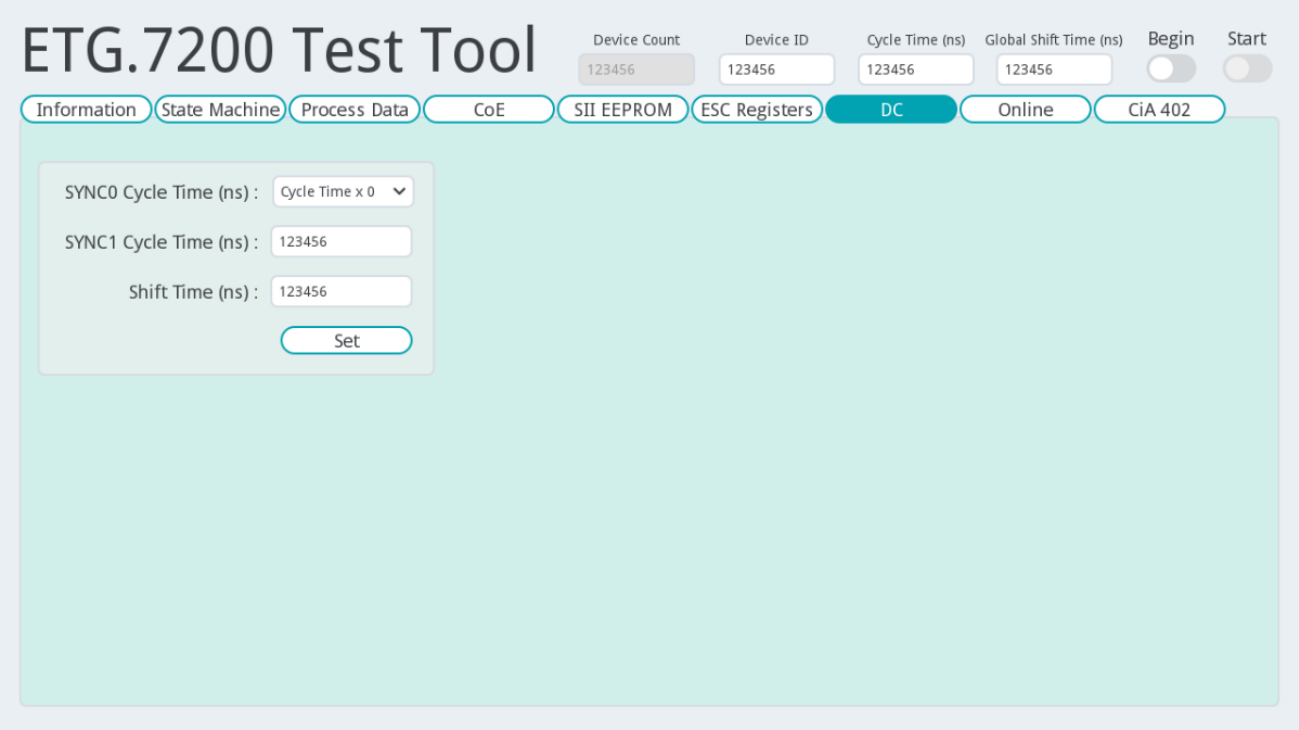
Field / Control	Description
ESC Address (ESC Address, hex)	4-digit hexadecimal register address to write (e.g., 0120 for AL Control).
Value (hex)	Hexadecimal value to write.
(dec) (decimal)	Decimal representation of the write value.
Bit Size (dropdown)	Data width: 8, 16, 32, or 64 bits.
Write button	Immediately writes the specified value to the ESC register. Begin must be enabled first.

> **Note** *ESC register writes take effect immediately and cannot be undone. Writing to reserved addresses or critical-function registers (e.g., AL Control 0x0120) may unexpectedly change the device state.*

7.2 Read / Display Panel

Field / Control	Description
ESC Address (ESC Address, hex)	Starting address for the read operation.
Reload button	Reads 256 bytes from the specified address and displays them in the table below, while also reading the station alias value from register 0x0012.
Alias 0012h (hex, read-only)	After Reload, shows the value of ESC register 0x0012 (Station Alias).
Alias 0012h (decimal, read-only)	Decimal representation of the alias value.
Register table	Displays the ESC register contents in a 16 row × 16 byte format. Each row header shows the starting address, and the remaining columns show the data in hexadecimal groups.

8. DC Tab (Distributed Clocks)



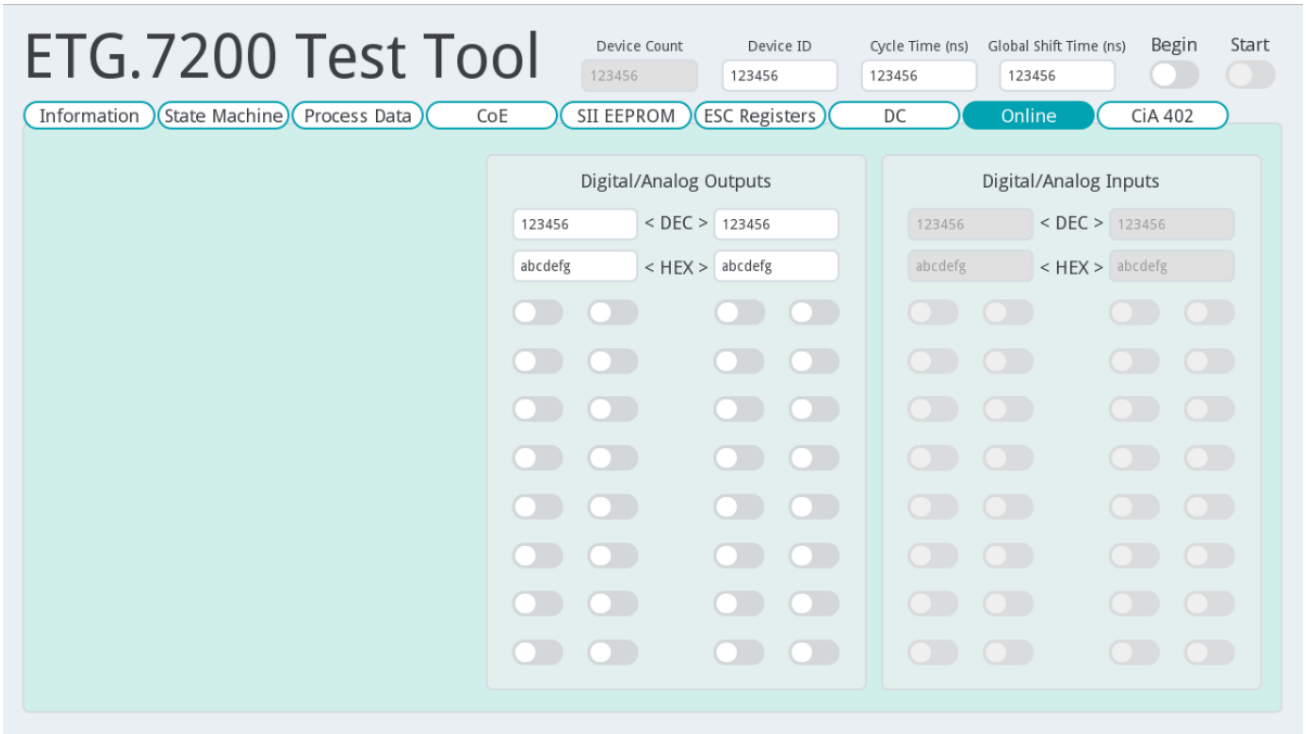
[Figure: screenshot — Figure 7 — DC tab]

The DC tab configures the Distributed Clocks (DC) synchronization parameters of the selected SubDevice. Clicking Set invokes `master.setDc()` to apply the settings. This operation requires Begin enabled and Start disabled.

Field / Control	Description
SYNC0 Cycle Time (ns) (dropdown)	Multiplies the global Cycle Time by the selected factor to form the SYNC0 pulse period. Options: Cycle Time × 0 (disabled) through × 10.
SYNC1 Cycle Time (ns)	SYNC1 pulse period, in nanoseconds. Set to 0 to disable SYNC1. Value must be ≥ 0.
Shift Time (ns)	Local DC shift time of the SubDevice relative to the network reference clock, in nanoseconds. Negative values are accepted.
Set button	Calls `master.setDc()` with the three parameters above and shows the result in a message box. Operable only when Begin is enabled and Start is disabled.

> **Note** *The absolute time values in the SYNC0 multiplier dropdown are derived from the current Cycle Time. If you need a non-default period, modify Cycle Time first, then configure DC.*

9. Online Tab (I/O)



[Figure: screenshot — Figure 8 — Online tab]

The Online tab provides real-time monitoring of PDO digital inputs and real-time control of PDO digital outputs. Each side supports up to 32 bits of data (two 16-bit words). This tab functions only after Start is enabled.

9.1 Digital / Analog Outputs (Left Panel)

The output panel offers three synchronized input methods for controlling PDO output values: decimal word, hexadecimal word, and individual bit toggle switches. The three representations remain synchronized. After Start is enabled, output data is written to the SubDevice every PDO cycle.

Element	Description
DEC field (Word 1 / Word 2)	16-bit output value as an unsigned decimal integer. Range: 0–65535. Word 1 corresponds to bits 0–15; Word 2 to bits 16–31.
HEX field (Word 1 / Word 2)	4-digit hexadecimal string representation of the same value. When the Hex field is modified, the Dec field and bit toggle switches update synchronously.
Bit toggle switches (32 total)	Individual bit-control switches arranged as 8 columns × 4 rows. When any switch is toggled, the corresponding Dec and Hex fields update immediately.

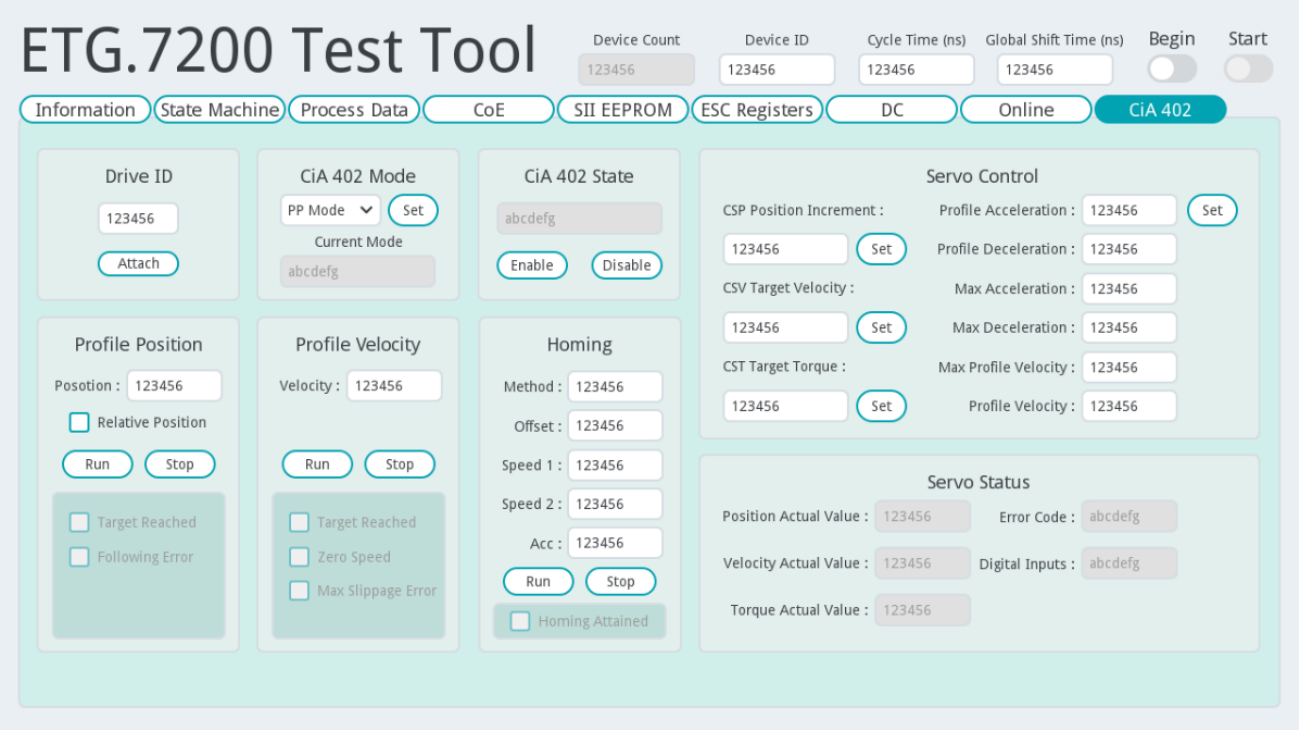
> **Note** *Bits beyond the SubDevice's configured PDO output width are automatically masked to 0 before writing.*

9.2 Digital / Analog Inputs (Right Panel)

The input panel reflects the latest PDO data received from the SubDevice. All fields are read-only and update automatically every polling cycle.

Element	Description
DEC field (Word 1 / Word 2)	Unsigned decimal integer representation of the current input value.
HEX field (Word 1 / Word 2)	4-digit hexadecimal string representation of the current input value.
Bit indicator switches (32 total)	Read-only toggle indicators; each switch corresponds to one input bit. ON = logical 1; OFF = logical 0.

10. CiA 402 Tab (Servo Drive)



[Figure: screenshot — Figure 9 — CiA 402 tab]

The CiA 402 tab implements the IEC 61800-7 / CiA 402 servo drive profile and includes five function blocks: Drive ID and connection, mode of operation, CiA 402 state control, servo control parameters, and mode-specific motion command panels.

10.1 Drive ID and Attach

Control	Description
Drive ID field	Axis number within the EtherCAT node (0–255). Must match the axis number configured in the drive firmware.
Attach button	Calls `motor.attach()` to connect the CiA 402 drive object to the selected Device ID and Drive ID. Upon Attach, the system automatically reads motion parameters from the drive and pre-fills the fields. Click again to perform Detach. Begin must be enabled first.

10.2 CiA 402 Mode

Control	Description
Mode dropdown	Select the target mode of operation: PP (Profile Position), PV (Profile Velocity), HM (Homing), CSP (Cyclic Synchronous Position), CSV (Cyclic Synchronous Velocity), CST (Cyclic Synchronous Torque).
Set button	Calls `motor.setCiA402Mode()` to set the selected mode. When switching to CSP mode, the current actual position is latched as the initial target position to prevent position jumps.
Current Mode (Current Mode, read-only)	Displays the mode of operation currently reported by the drive.

10.3 CiA 402 State

Control	Description
State display field (read-only)	Current CiA 402 state machine state: Not Ready To Switch On, Switch On Disabled, Ready To Switch On, Switch On, Operation Enabled, Fault, Fault Reaction Active, or Quick Stop Active.
Enable button	Executes the CiA 402 standard enable sequence to bring the drive into Operation Enabled state.
Disable button	Returns the drive to Switch On Disabled state.

10.4 Servo Control — Motion Parameters

The following fields apply to PP and PV modes. They are pre-loaded from the drive upon Attach, and written to the drive in a single batch when Set is clicked.

Field	Object / Description
Profile Acceleration	Object 0x6083 — acceleration, unit: user units/s².
Profile Deceleration	Object 0x6084 — deceleration, unit: user units/s².
Max Acceleration	Object 0x60C5 — maximum allowable acceleration.
Max Deceleration	Object 0x60C6 — maximum allowable deceleration.
Max Profile Velocity	Object 0x607F — upper velocity limit for profile motion.
Profile Velocity	Object 0x6081 — target velocity for PP mode motion.
Set button	Writes the six parameters above to the drive in a single batch.

10.5 Servo Control — Cyclic Synchronous Commands (CSP / CSV / CST)

The following commands are executed every EtherCAT cycle via CyclicCallback. The drive must be in Operation Enabled state for commands to take effect.

Field / Button	Description
CSP Position Increment (CSP Position Increment)	Position increment accumulated each cycle (e.g., 100 means 100 counts per 1 ms cycle). Negative values produce reverse motion.
CSP Set button	Apply the CSP position increment command. The drive must be in CSP mode and Operation Enabled.
CSV Target Velocity (CSV Target Velocity)	Constant velocity command for CSV mode, unit: user units/s. Negative values produce reverse motion.
CSV Set button	Apply the CSV velocity command.
CST Target Torque (CST Target Torque)	Constant torque command for CST mode. Range: −32768 to 32767 (0.1% of rated torque).
CST Set button	Apply the CST torque command.

10.6 Profile Position Panel

Field / Button	Description
Position field	Target position, unit: user units.
Relative Position checkbox	When checked, motion is executed as a relative displacement; otherwise, as an absolute position.
Run button	Executes profile position motion to the target position (`motor.pp_Run()`).
Stop button	Issues a relative-displacement-0 command to stop the drive and acknowledge the new setpoint.
Target Reached (read-only checkbox)	Reflects the statusword — set to ON when the drive reaches the target position.
Following Error (read-only checkbox)	Reflects the statusword — set to ON when the following error exceeds the configured limit.

10.7 Profile Velocity Panel

Field / Button	Description
Velocity field	Target velocity, unit: user units/s.
Run button	Commands the drive to accelerate to the target velocity (`motor.pv_Run()`).
Stop button	Commands the drive to decelerate to zero velocity (`motor.pv_Run(0)`).
Target Reached (read-only checkbox)	Set to ON when the drive velocity reaches the target velocity.
Zero Speed (read-only checkbox)	Set to ON when the drive velocity is below the zero-speed threshold.
Max Slippage Error (read-only checkbox)	Set to ON when the difference between commanded and actual velocity exceeds the limit.

10.8 Homing Panel

Field / Button	Description
Method (homing method)	Integer homing method code (−128 to 127). Default: 33 (index pulse homing). See CiA 402 Section 13 for full definitions.
Offset	Signed 32-bit offset appended to the home position after homing completes.
Speed 1 (search speed)	Speed used when searching for the limit switch.
Speed 2 (zeroing speed)	Crawl speed used when searching for the home position or index pulse.
Acc (acceleration)	Acceleration and deceleration used during the homing motion.
Run button	Writes all homing parameters to the drive and starts the homing procedure (`motor.hm_Run()`).
Stop button	Stops the homing operation (`motor.hm_Stop()`).
Homing Attained (read-only checkbox)	Set to ON when the drive reports successful completion of the homing procedure.

10.9 Servo Status

All fields below update automatically every polling cycle while Attach is active.

Field	Description
Position Actual Value	Current position from object 0x6064, unit: user units.
Velocity Actual Value	Current velocity from object 0x606C.
Torque Actual Value	Current torque from object 0x6077 (0.1% of rated torque).
Error Code	4-digit hexadecimal drive error code from object 0x603F.
Digital Inputs	8-digit hexadecimal value of the drive digital inputs object 0x60FD.

11. Troubleshooting

Symptom	Possible Cause	Resolution
Error code shown after toggling Begin	Based on the error code returned	Based on the returned error code, consult the error code reference table.
Start switch is grayed out and unusable	Bootstrap mode is active	Disable Bootstrap on the State Machine tab, then enable Start.
SDO Upload / Download return an error	SubDevice not in Pre-Op or higher state, or based on the error code returned	Verify Begin is enabled and the SubDevice has transitioned at least to Pre-Op state. And based on the returned error code, consult the error code reference table.
FoE Download fails	SubDevice not in Bootstrap state	Command the SubDevice to Bootstrap state on the State Machine tab first, then perform the FoE transfer.
CiA 402 Attach fails	Drive ID mismatch	Verify the Drive ID matches the axis number configured in the drive firmware.
Position jump when attaching in CSP mode	Stale initial target position	The system prevents this automatically: on Attach, the current actual position is latched as CspTargetPosition.
Emergency message appears on CoE tab	Drive or I/O fault	Record the Error Code and Error Register, and consult the SubDevice vendor's error code documentation.
Bit toggle switches have no effect	Start is not yet enabled	Output bit toggle switches take effect on the SubDevice only after Start is enabled.

12. Hands-on Tutorials

12.1 Scenario A — First-time Connection and SubDevice Identification

Purpose: After connecting an EtherCAT SubDevice to the QEC platform for the first time, use the ETG.7200 tool to establish the MainDevice–SubDevice link and read identification information from the Information tab, confirming that hardware, wiring, and tool communication are all normal.

Prerequisites:

- QEC hardware (86Duino + 86HMI) is powered on; the HMI has booted and shows the ETG.7200 main screen.
- The EtherCAT SubDevice under test is connected to the QEC's designated EtherCAT port (typically ETH0, EC OUT), and the SubDevice has its own power.
- The SubDevice datasheet (or ESI file) is at hand to verify identification info such as Vendor ID and Product Code.

Steps:

1. Confirm the Cycle Time on the global toolbar remains at the default 1,000,000 ns and Global Shift Time stay at 0.

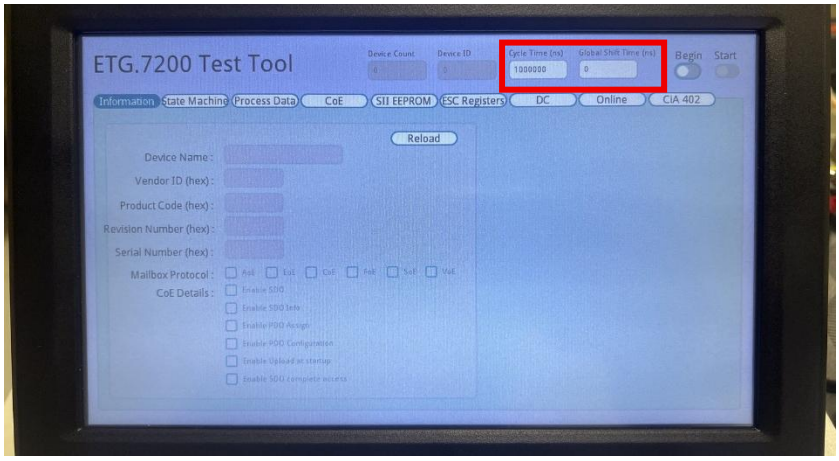


Figure 12.1-1 Before toggling Begin, Cycle Time stays at the default.

2. Toggle the Begin switch to ON. The system calls `master.begin()` to initialize the EtherCAT network and scan SubDevices.
3. Check the Device Count field; it should update automatically to the number of connected SubDevices. With one device connected, Device Count should be 1.

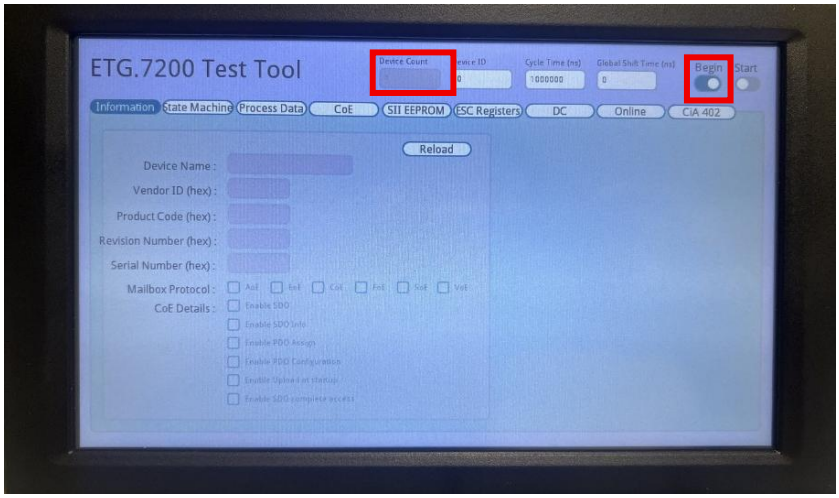


Figure 12.1-2 After toggling Begin to ON, Device Count automatically updates to the number of detected SubDevices.

4. Set Device ID = 0 to select the target SubDevice for this operation.
5. Click the tab label to switch to the Information tab.
6. Click the Reload button; the tool queries the SubDevice and refreshes all identification fields.

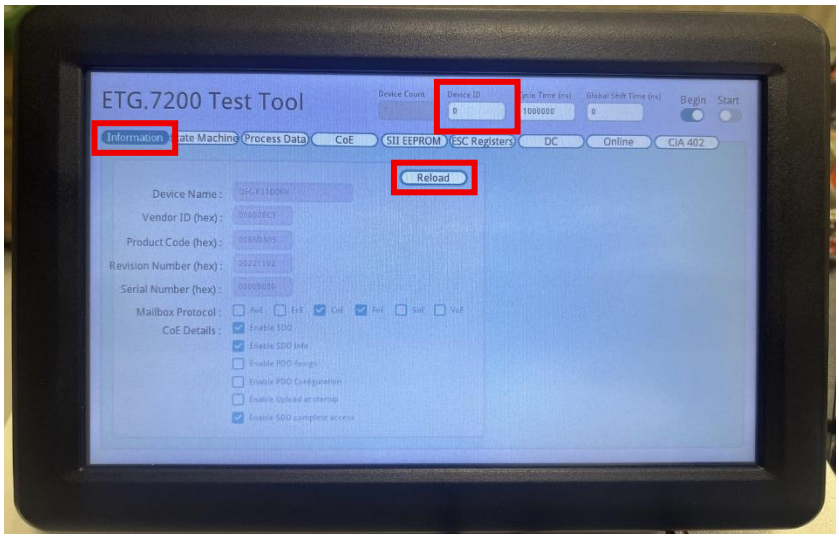


Figure 12.1-3 After a successful Reload, all Information-tab fields are populated and the supported Mailbox protocols are checked.

7. Review Device Name, Vendor ID, Product Code, Revision Number, and Serial Number, and verify against the SubDevice datasheet or ESI file.
8. Inspect the Mailbox Protocol and CoE Details checkboxes to confirm that the protocols declared by the SubDevice (e.g., CoE, FoE) match the testing needs that follow.

Expected Results:

- Device Count matches the number of physically connected SubDevices.
- All identification fields on the Information tab have valid values; Vendor ID and Product Code match the datasheet.
- The Mailbox Protocol block has at least CoE checked (supported by virtually all EtherCAT SubDevices).

Verification:

Compare the Vendor ID (hex) shown on the Information tab against the Vendor Id field in the SubDevice datasheet or ESI file; an exact match indicates the MainDevice and SubDevice can communicate via mailbox correctly.

Common Issues:

- Device Count shows 0: Check the EtherCAT cable, SubDevice power, and the network interface used by the QEC. Try unplugging and re-plugging the EtherCAT cable, then toggle Begin OFF and ON again to retry.
- Reload button has no response or returns an error: Confirm Begin is enabled; if it still fails, transition the SubDevice to Pre-Op or higher first, then Reload again.
- Identification fields show abnormal or garbled values: The SubDevice SII EEPROM contents may be corrupted; check and re-program on the SII EEPROM tab.

> **Note** *At the end of this scenario, keep Begin ON and Start OFF to facilitate the subsequent PDO and DC configuration in Scenarios B and C.*

12.2 Scenario B — Configure PDO and Verify I/O

Purpose: Using an EtherCAT SubDevice with digital or analog I/O as an example, configure PDO Assignment and PDO Mapping on the Process Data tab and verify output and input signals on the Online tab.

Prerequisites:

- Scenario A is completed; SubDevice identification can be read from the Information tab.
- Begin is ON and Start is OFF. PDO Mapping must be done while Start is disabled.
- The SubDevice ESI file or PDO mapping documentation is at hand to identify the available RxPDO / TxPDO indices and the corresponding objects (0x1C12 & 0x1C13).
- Set Begin to ON and Start to ON. EtherCAT enters OP state, executing I/O functions via PDO.
- For output verification, an observable indicator (LED), relay, or actual controlled device should be available at the field; for input verification, a triggerable signal source (button, sensor, etc.) should be available.

Steps:

1. Switch to the State Machine tab and ensure the SubDevice is in Pre-OP state (Current State shows Pre-OP).

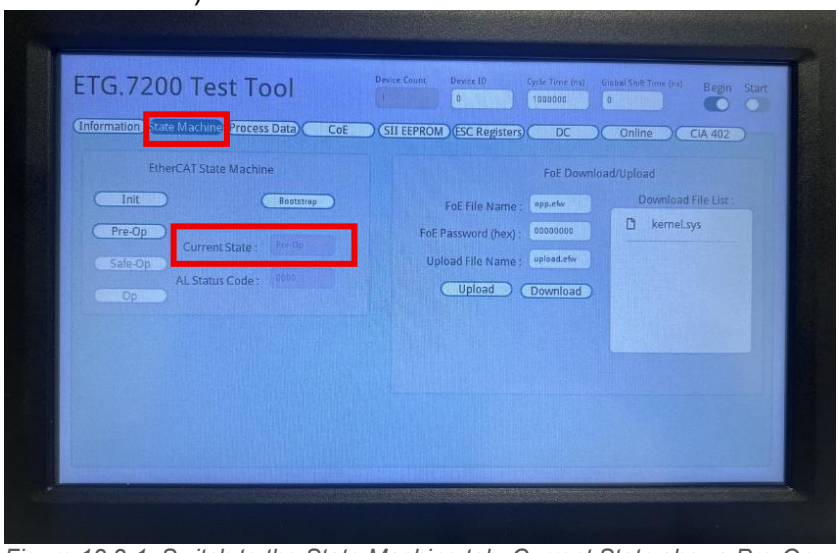


Figure 12.2-1 Switch to the State Machine tab; Current State shows Pre-Op.

2. Switch to the Process Data tab.

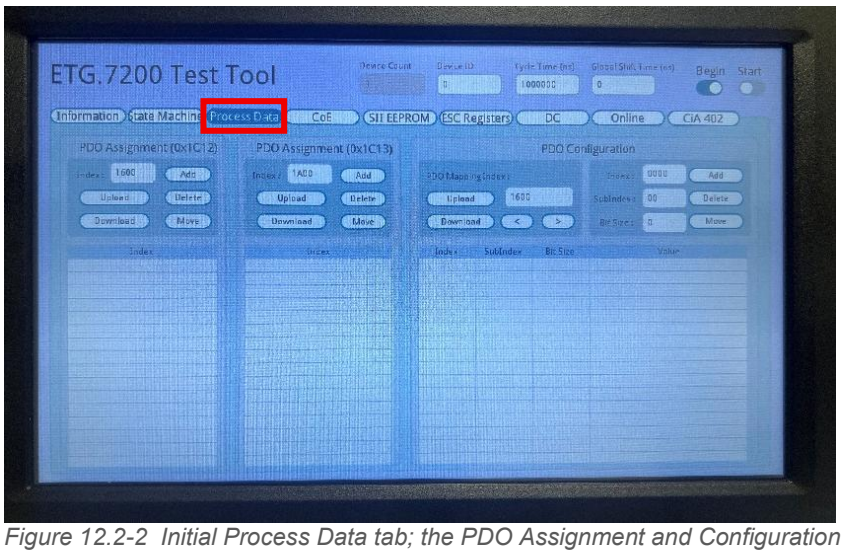


Figure 12.2-2 Initial Process Data tab; the PDO Assignment and Configuration blocks have no entries yet.

3. In the PDO Assignment 0x1C12 (RxPDO Assignment) block, this is the RxPDO index to enable (e.g., 0x1600). Click Upload to display the default 0x1C12 object in the table below; click Download to download Assignment and Mapping to the SubDevice via SDO. Click Upload again to verify the write was successful.
4. In the PDO Assignment 0x1C13 (TxPDO Assignment) block, this is the TxPDO index to enable (e.g., 0x1A00). The procedure is the same as step 3. (Because this document demonstrates a Digital Output module, no TxPDO index applies.)

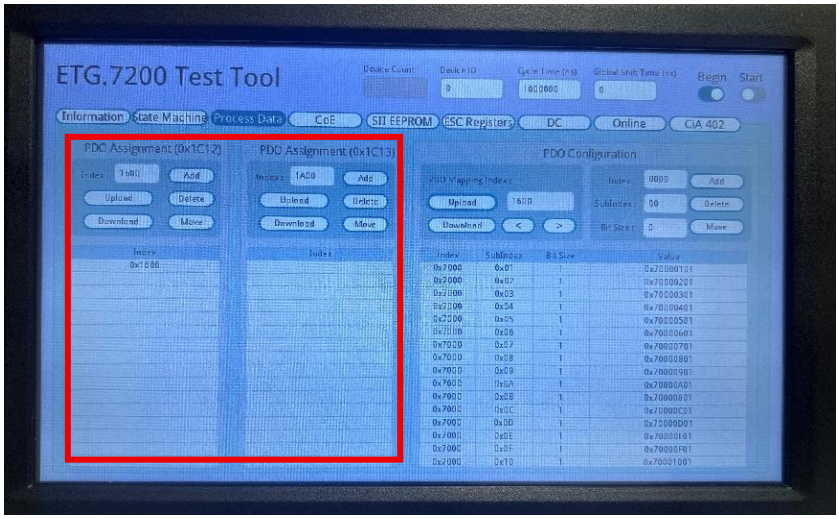


Figure 12.2-3 After clicking Upload on 0x1C12, the left Index area shows the assigned RxPDO (e.g., 0x1600).

5. In the PDO Configuration block, view or modify the mapping entries (Index, Subindex, Bit Length) for each assigned PDO. To customize, add a new row and fill in the target object and bit length.

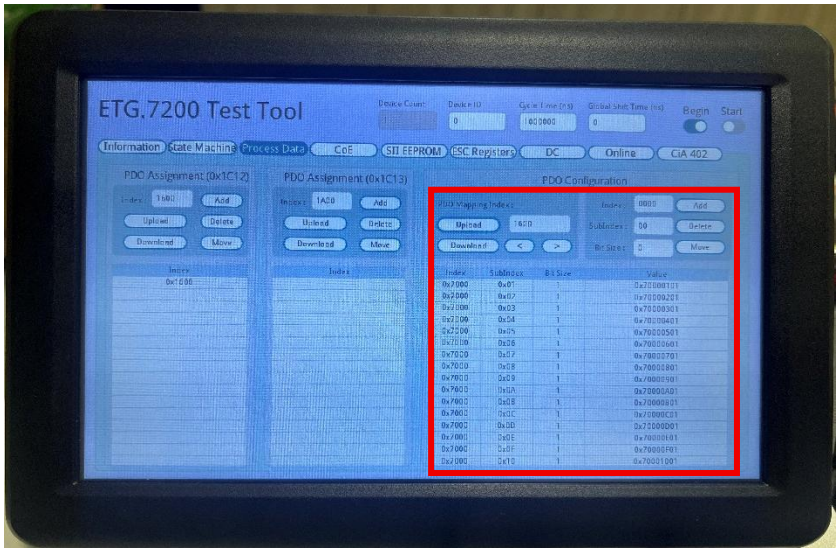


Figure 12.2-4 The PDO Configuration block shows the mapping entries of 0x1600 (SubIndex, Bit Size, Value).

6. Once done, toggle the Start switch to ON to set EtherCAT to OP state and enable cyclic PDO communication.
7. Confirm that Current State shows Op and AL Status Code is 0x0000.

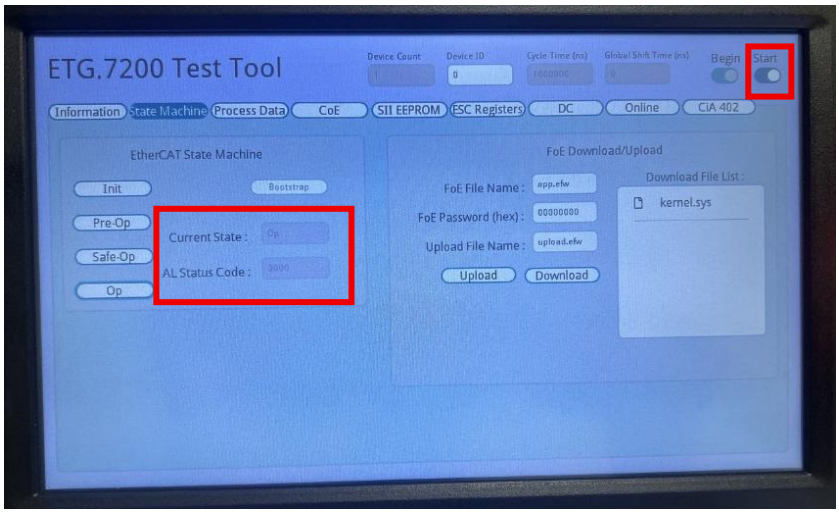


Figure 12.2-5 State Machine tab: after enabling Start, Current State switches to Op with AL Status Code 0x0000.

8. Switch to the Online tab and inspect the output panel on the left and the input panel on the right.
9. Toggle any switch in the left output block, or enter values in the analog output fields, and observe whether the corresponding field device responds.

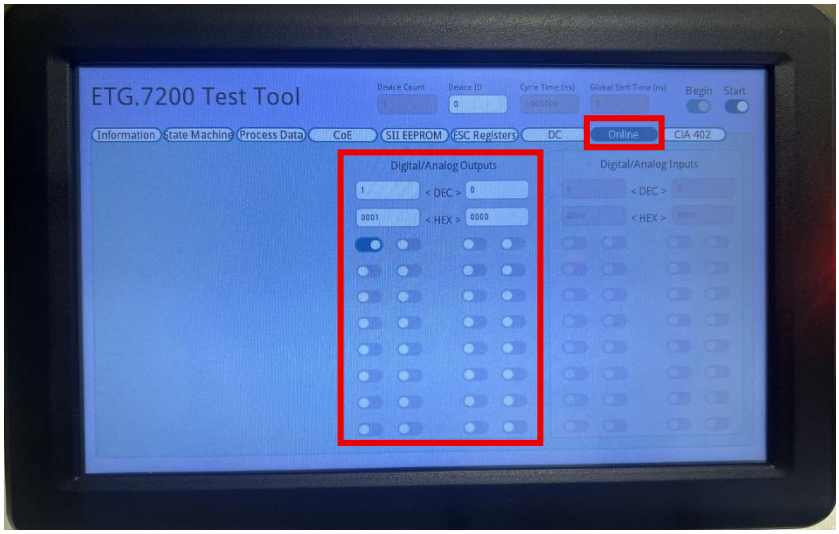


Figure 12.2-6 Online tab: after toggling the first output bit on the left, the corresponding LED lights up at the field.

Expected Results:

- After Start is enabled, the SubDevice transitions smoothly through Init, Pre-Op, Safe-Op, and Op states.
- Field output actions correspond exactly to operations on the Online tab; field input changes are reflected on the screen in real time.
- AL Status Code remains 0x0000 throughout, with no errors.

Verification:

Use a multimeter, logic analyzer, or field device (e.g., LED, relay) for bidirectional verification: measure actual voltage changes when toggling outputs, and observe synchronized bit changes on screen when triggering inputs.

Common Issues:

- Apply / Download fails with an ECAT_ERR_MASTER_NOT_IN_PREOP (-1102) class error: Verify the SubDevice is in Pre-OP state and Start is OFF, then retry.
- Transition to Safe-Op fails (AL Status Code non-zero): typically PDO Mapping doesn't match the SubDevice object dictionary; return to the Process Data tab and check mapping lengths and object addresses.
- Bit toggle switches have no effect: Start is not yet enabled, or the SubDevice is not in Op state — output bits will not take effect.

12.3 Scenario C — Configure DC Synchronization

Purpose: For SubDevices supporting Distributed Clocks (DC), complete DC parameter configuration and enable SYNC0/SYNC1 so that SubDevice operation is strictly synchronized to the MainDevice clock, laying the groundwork for subsequent high-precision motion control (e.g., CSP/CSV).

Prerequisites:

- Scenario A is completed and the SubDevice can be identified.
- Begin is ON and Start is OFF. DC parameters must be configured while Start is disabled.
- The SubDevice supports DC: confirm on the Information tab or check via `isSupportDC()` API.
- The system Cycle Time has been decided (e.g., 1 ms, 500 µs, 250 µs) and is within the SubDevice's allowed range.

Steps:

1. Set Cycle Time to the desired value on the global toolbar (e.g., 1,000,000 ns for 1 ms).
2. Leave Global Shift Time at 0; the MainDevice will compute a suitable shift automatically at Start.

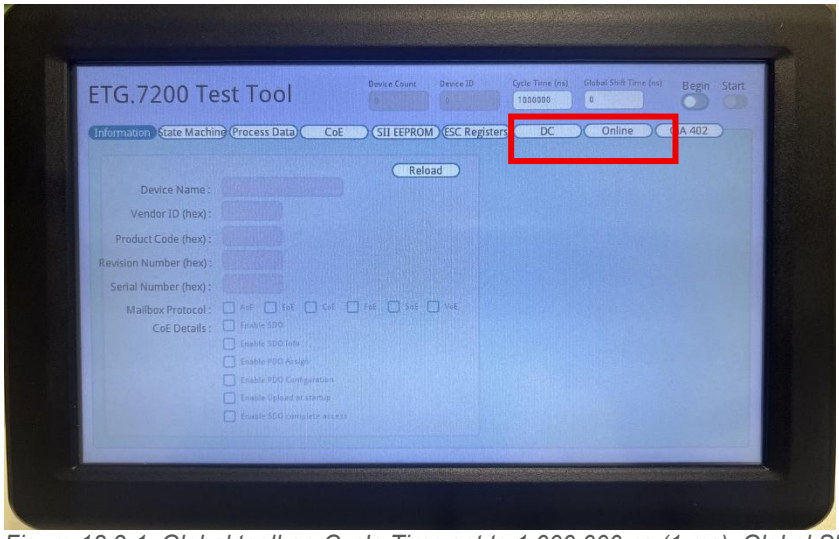


Figure 12.3-1 Global toolbar: Cycle Time set to 1,000,000 ns (1 ms), Global Shift Time left at 0.

3. Switch to the DC tab.
4. On the DC mode dropdown, choose SYNC0, SYNC0+SYNC1, or Disable as required. SYNC0 is typically recommended for CiA 402 servo drives.

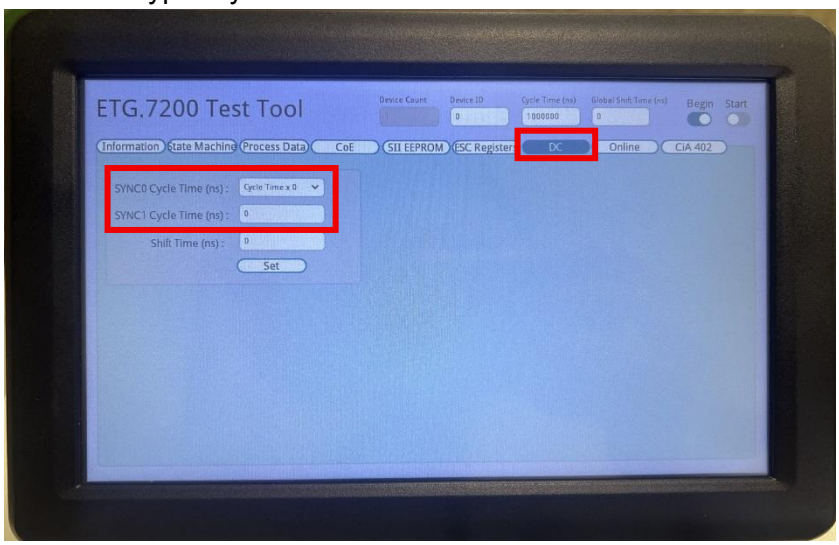


Figure 12.3-2 Initial DC tab: SYNC0 Cycle Time multiplier not yet configured.

5. Set SYNC0 Cycle Time, typically equal to or an integer multiple of Cycle Time (e.g., both at 1,000,000 ns). Use the SYNC0 dropdown to select.

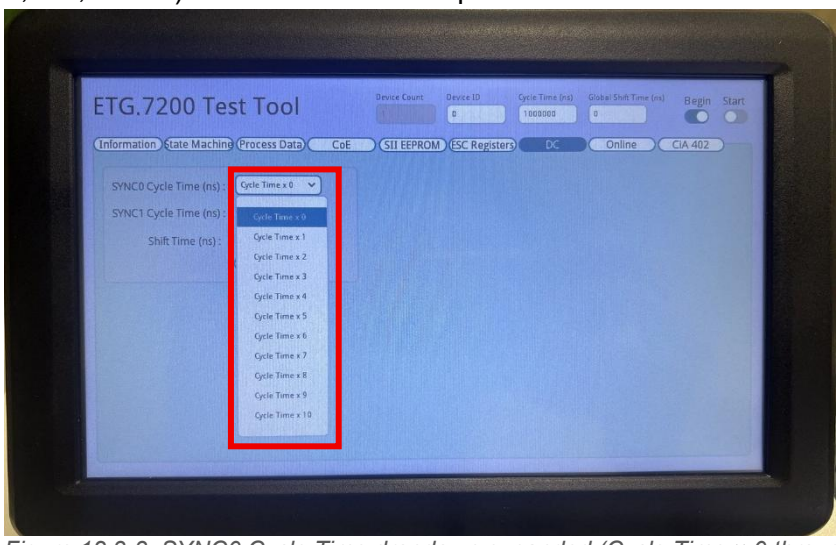


Figure 12.3-3 SYNC0 Cycle Time dropdown expanded (Cycle Time × 0 through × 10 selectable).

6. Click Set; the tool calls `setDc()` to apply the DC settings to the SubDevice. On success, a confirmation dialog appears.

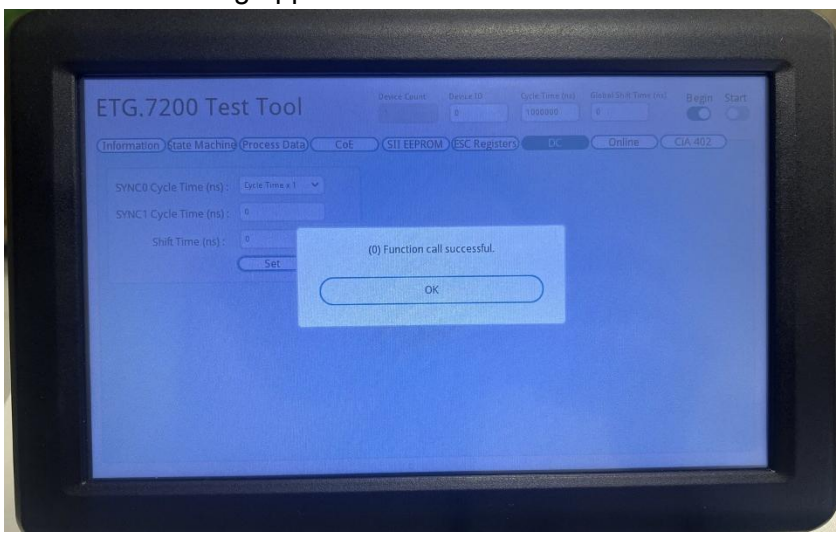


Figure 12.3-4 After clicking Set, "(0) Function call successful" is returned, indicating `setDc()` completed.

7. If other DC-capable SubDevices remain, change Device ID and repeat the steps above to ensure all DC-synchronized SubDevices are configured.

8. When done, toggle Start to ON; EtherCAT transitions to Op state, and the MainDevice has automatically computed an appropriate Global Shift Time at Start.

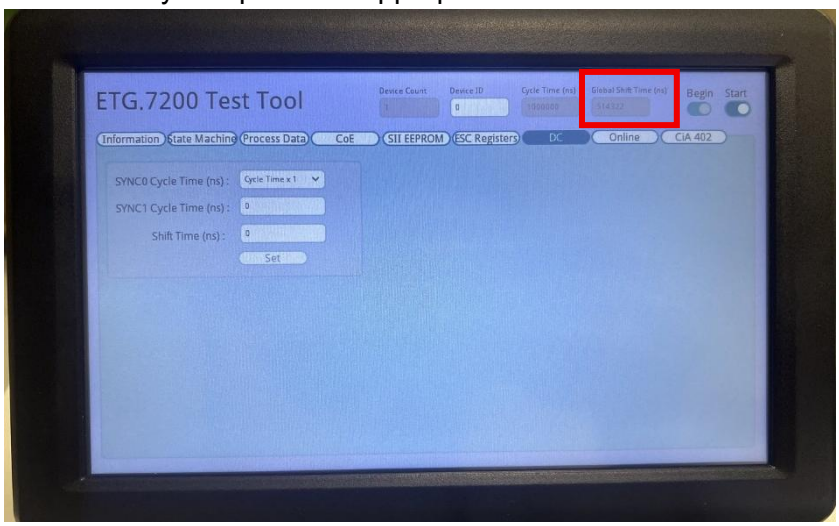


Figure 12.3-5 After toggling Start ON, the MainDevice automatically computes and populates a suitable Global Shift Time.

9. Return to the State Machine tab; AL Status Code should remain 0x0000 and Current State should stay at Op.

Expected Results:

- All DC-synchronized SubDevices transition smoothly into Op state.
- Working Counter (WKC) consistently equals Expected Working Counter, with no synchronization failures.
- When combined with a CiA 402 servo subsequently, the actual position trajectory produced by motion commands (CSP) is smooth and free of jitter.

Verification:

- Call `getWorkingCounter()` and `getExpectedWorkingCounter()` in the MainDevice's cyclic callback and compare; they should remain equal.
- Use an oscilloscope to observe the SubDevice's SYNC0 output pin (if provided); the waveform period should exactly match the configured SYNC0 Cycle Time.
- Switch to the ESC Registers tab and read DC-related registers (e.g., 0x0910 System Time, 0x098E SYNC0 Cycle Time) to confirm the actual values match the configured values.

Common Issues:

- Returns ECAT_ERR_DEVICE_NO_DC (-2003): the SubDevice does not support DC; disable DC or switch to an application that only requires free-running operation.
- Non-zero AL Status Code appears when transitioning to Safe-Op: typically SYNC0 Cycle Time is incompatible with MainDevice Cycle Time; adjust to be equal or an integer multiple.
- System time fails to converge (Device Count jumps wildly or error callbacks fire continuously after Start): check Global Shift Time, or reset it to 0 and let the MainDevice compute automatically.

> **Note** *DC settings are written to the SubDevice mailbox. To re-apply changes, transition back to Pre-OP first, then to OP.*

12.4 Scenario D — Servo Trial Run in PP Mode

Purpose: Use the CiA 402 tab to perform a short-stroke point-to-point trial run in Profile Position (PP) mode to verify drive parameters, PDO mapping, and tool control flow.

Prerequisites:

- The servo drive has completed identification, PDO configuration, and DC synchronization per Scenarios A, B, and C, and is currently in Op state.
- The machine is equipped with an Emergency Stop (E-Stop) and software limits; motor rotation has been verified not to cause mechanical collisions.
- The drive's motor resolution (6080h or 60EFh) and a safe small-stroke target position (e.g., within 1 revolution, converted to counts) are known.
- Start is ON; Begin is ON; DC is at Cycle time × 1.

Steps:

1. Switch to the CiA 402 tab.

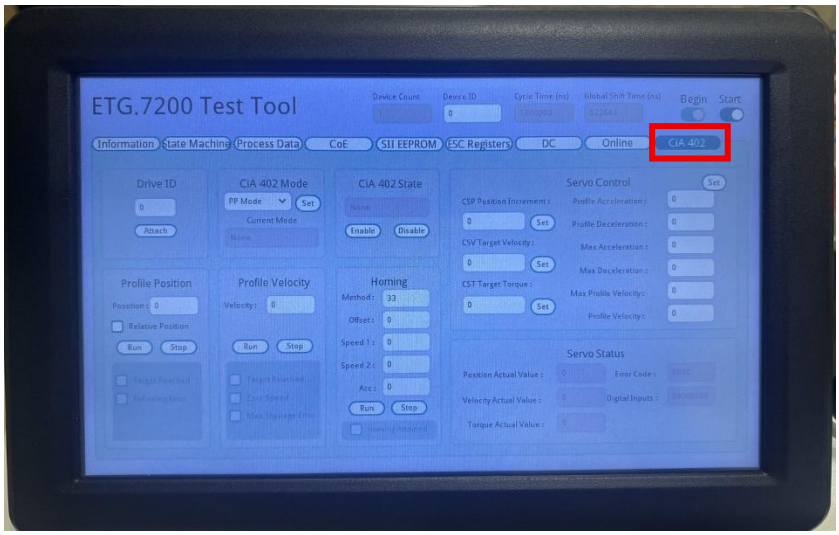


Figure 12.4-1 Initial screen after switching to the CiA 402 tab.

2. Enter the corresponding axis number into the Drive ID field (typically 0, unless a single SubDevice contains multiple axes).
3. Click the Attach button. On success, the button switches to a toggled state and displays Detach.

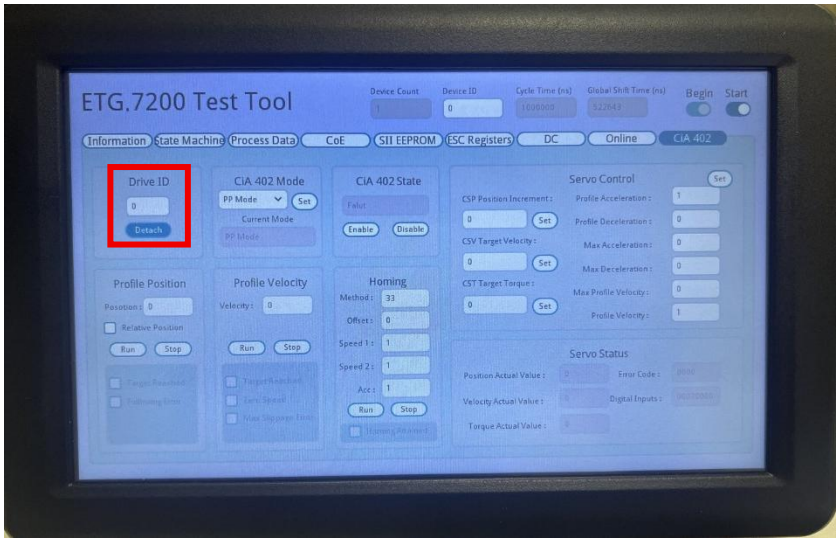


Figure 12.4-2 After clicking Attach, the button switches to Detach.

4. In the CiA 402 Mode panel, select Profile Position (pp) and click Set.

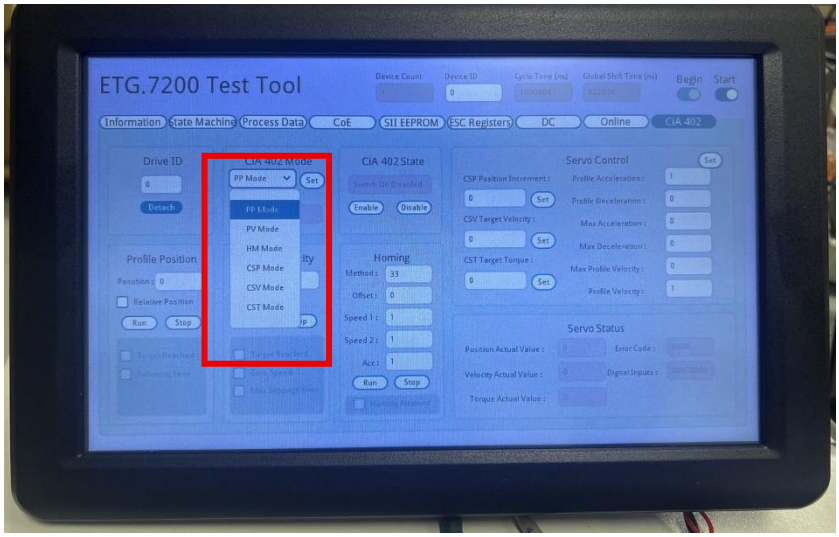


Figure 12.4-3 The CiA 402 Mode dropdown expanded, with six modes selectable: PP Mode / PV Mode / HM Mode / CSP Mode / CSV Mode / CST Mode.

The Current Mode field should show PP Mode.

5. Once the mode is set, in the CiA 402 State block, click the Enable button to switch the CiA 402 State to Operation Enabled. Verify that Current State shows Operation enabled and Error Code is 0. The motor should also be enabled.

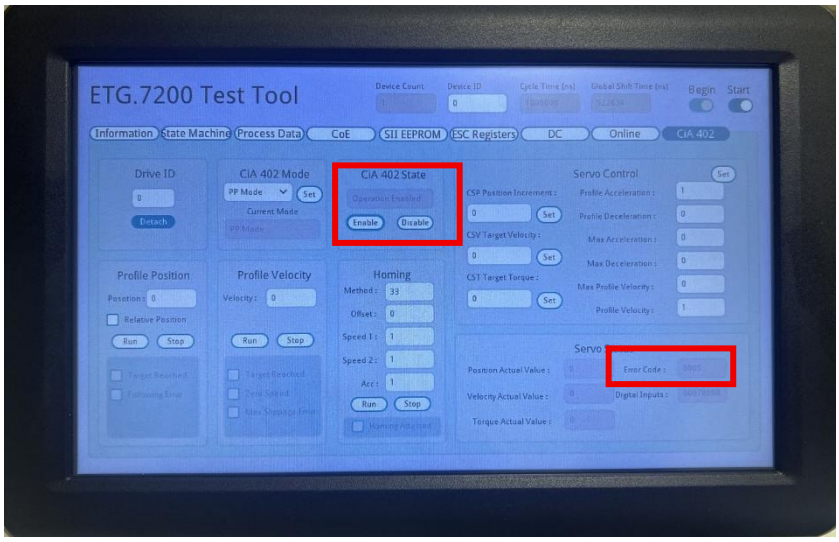


Figure 12.4-4 After clicking Enable, the CiA 402 State switches to Operation Enabled and the motor is energized.

6. In the Servo Control — Motion Parameters panel, set Profile Velocity, Profile Acceleration, and Profile Deceleration in sequence. Initial trial runs are recommended at low speed (e.g., profile velocity at 10% of rated motor speed). When done, click Set.

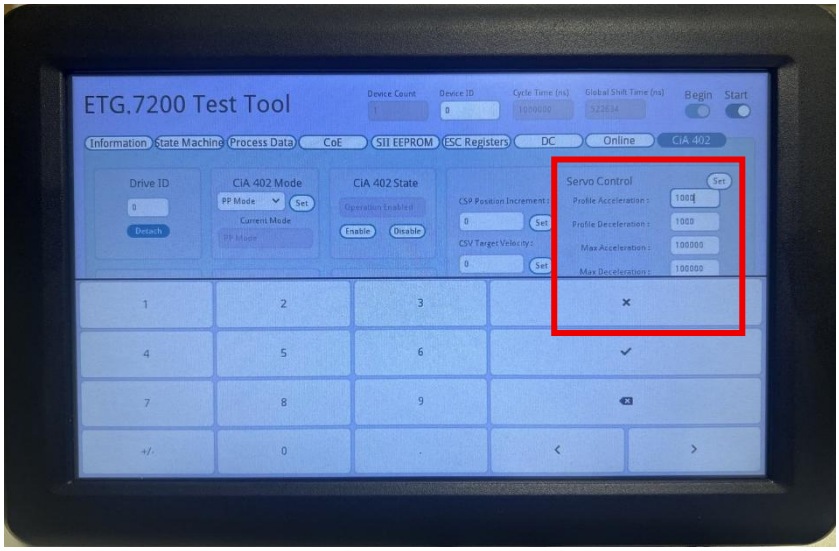


Figure 12.4-5 When entering motion parameters in Servo Control, the HMI brings up a numeric keypad to assist input.

7. In the Profile Position panel, enter Target Position; for the first trial run, a target within 1 motor revolution (in counts) is recommended.
8. Click the Run button to start motion.

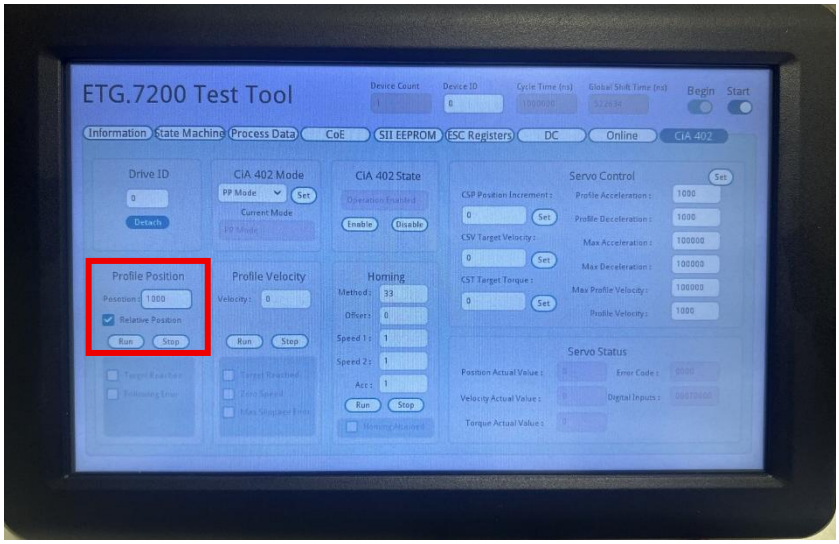


Figure 12.4-6 Profile Position and motion parameters are set, ready to press Run to start motion.

9. In the Servo Status block, observe Position Actual Value continuously increasing toward Target Position; the Target Reached flag is set to 1 when motion ends.

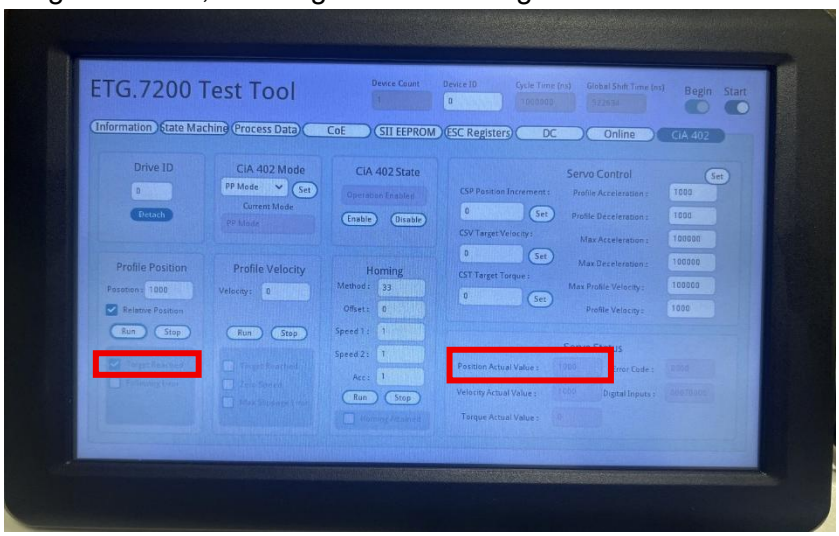


Figure 12.4-7 Run completed; Target Reached is checked, Position Actual Value = 1000.

Expected Results:

- The motor reaches the target position smoothly along the configured acceleration/deceleration curve, and Target Reached is set to 1 within the Position Window tolerance.
- During motion, Error Code remains 0 and AL Status Code remains 0x0000.
- After Disable, the motor releases output and can be turned manually.

Verification:

- Use `getPositionActualValue()` and `pp_IsTargetReached()` from Chapter 13 within the cyclic callback to record the actual trajectory and confirm target arrival.
- Use an external encoder or position sensor to compare actual displacement against Target Position.

Common Issues:

- Attach fails with ECAT_ERR_DEVICE_CIA402_ADD_FAIL (-2501): Drive ID does not match the axis number in the drive firmware; verify drive settings.
- After Enable, Current State stops at Switched On or shows Fault: read Error Code (603Fh) and handle per the drive error table (common causes: excitation abnormality, limit triggered, etc.).
- After Run, position does not change and Target Reached remains 0: verify Profile Velocity and Profile Acceleration are non-zero, and that Target Position differs sufficiently from the current position.
- Position jumps or stalls: verify PDO mapping includes 6040h/607Ah/6041h/6064h, or switch to CSP mode with Shift Time (Scenario C) for verification.

> **Note** *For first-time trials, limit the target position to within 1 revolution and be ready to press E-Stop or click Disable at any moment to guard against accidents from misconfigured parameters.*

12.5 Scenario E — Cyclic Synchronous Operation in CSP Mode

Purpose: Use the Cyclic Synchronous Position (CSP) mode on the CiA 402 tab, where the MainDevice writes a new target position to the drive every EtherCAT cycle. This scenario verifies the CSP communication chain, CspTargetPosition behavior, and CSP Position Increment control.

Prerequisites:

- The servo drive has completed identification and PDO configuration per Scenarios A and B; the PDO mapping includes at least 6040h, 6041h, 6064h, 607Ah.
- DC synchronization is completed per Scenario C; CSP mode is sensitive to DC jitter, and lacking DC typically causes uneven position output.
- The drive supports CSP: confirm via `getSupportedDriveModes()` or check the CSP option in the CiA 402 Mode dropdown.
- The machine is equipped with an Emergency Stop (E-Stop); once CSP mode is enabled, the drive follows target positions in real time and motion may be faster than PP.
- Begin is ON, Start is ON, DC is at Cycle time × 1, and the SubDevice is in Op state.

Steps:

1. Switch to the CiA 402 tab.
2. Enter the axis number (typically 0) in the Drive ID field and click Attach. On success, the button switches to Detach.
3. In the CiA 402 Mode panel, select CSP Mode and click Set. Current Mode should update to CSP Mode.

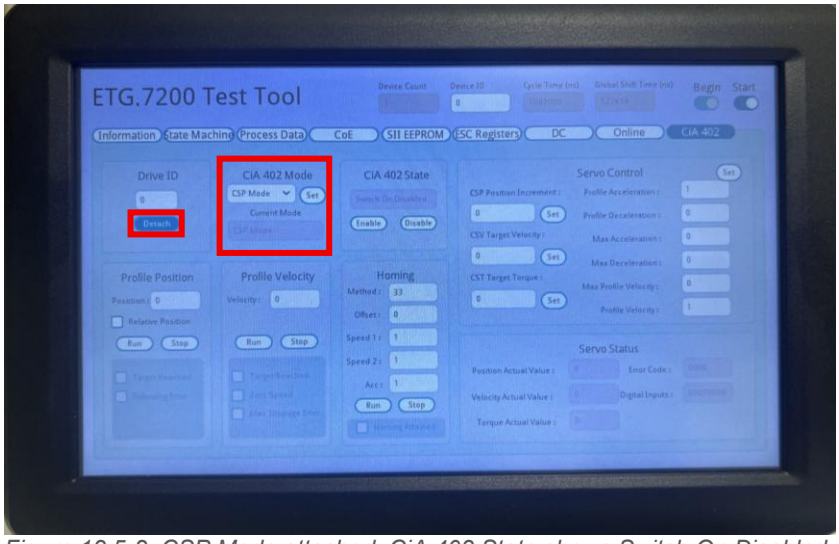


Figure 12.5-2 CSP Mode attached; CiA 402 State shows Switch On Disabled, awaiting Enable.

4. In the CiA 402 State panel, click Enable to switch the drive to Operation Enabled. At this point CSP Position Increment is 0; the motor has not moved but is energized. CSP positions are supplied by the MainDevice every cycle; Increment is added to (or subtracted from) the current position.

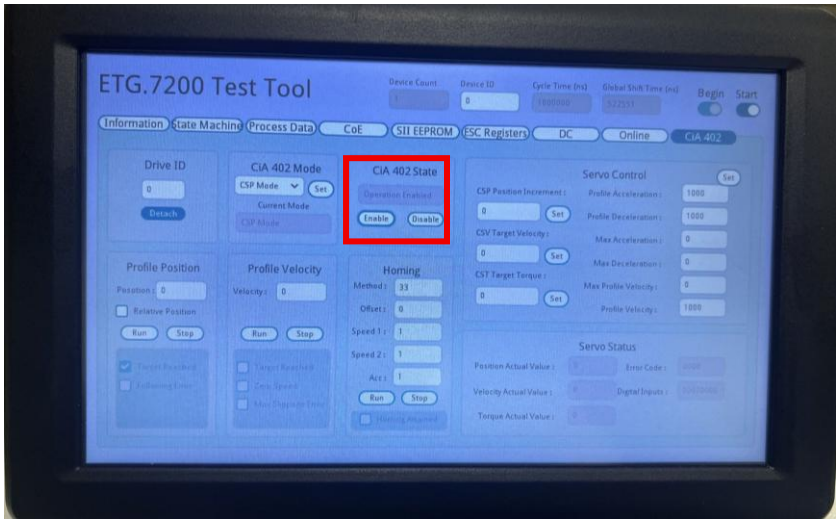


Figure 12.5-4 After clicking Enable, CiA 402 State shows Operation Enabled; Increment remains 0 and the motor is idle, ready.

5. Tap the CSP Position Increment field; the HMI brings up a numeric keypad. Enter the per-cycle position increment (positive values move forward, negative values move backward; this example uses 1 count/cycle), then press ✓ to confirm.

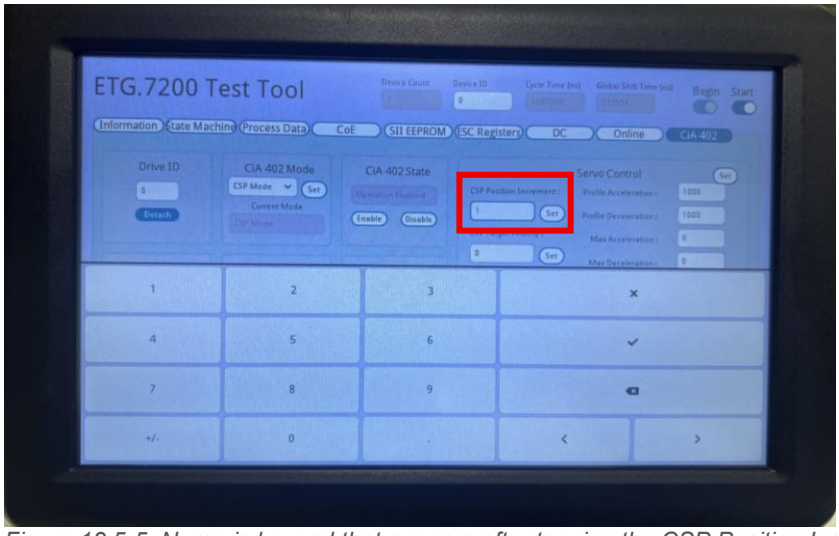


Figure 12.5-5 Numeric keypad that appears after tapping the CSP Position Increment field.

6. Press Set next to the Increment field; the MainDevice adds Increment counts to the target position each cycle. The motor begins continuous motion at a rate of approximately Increment × (1 / Cycle Time).

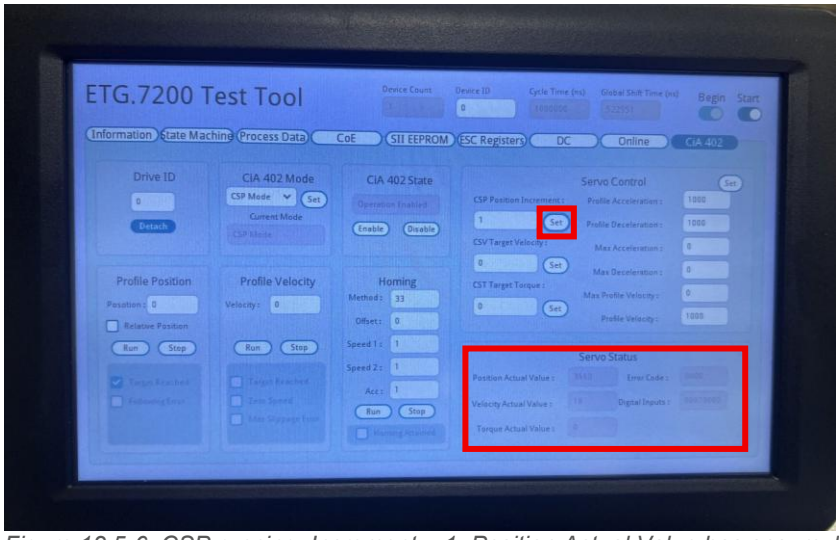


Figure 12.5-6 CSP running: Increment = 1; Position Actual Value has accumulated to 3553.

7. To stop, set CSP Position Increment to 0 and press Set; the target position is frozen at the last written value. Alternatively, click Disable to enter Switch On Disabled; the motor stops according to the drive's Disable Operation Option (typically a natural stop).

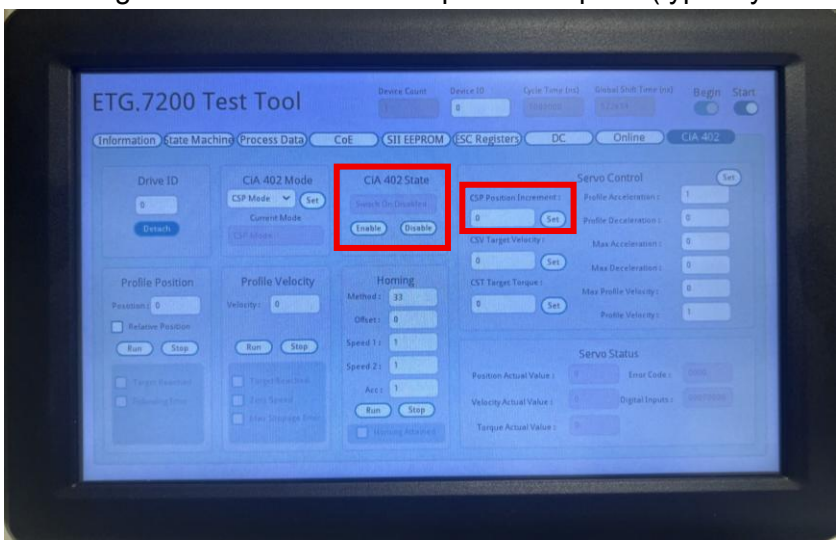


Figure 12.5-7 To stop, set CSP Position Increment to 0 and press Set; alternatively, click Disable to enter Switch On Disabled.

Expected Results:

- The motor runs continuously at a constant speed determined by CSP Position Increment.
- The difference between Position Actual Value and Position Demand Value (i.e., the Following Error) stays within the drive's configured Following Error Window.
- Error Code remains 0, AL Status Code remains 0x0000, and Working Counter equals Expected Working Counter.
- After Disable, the motor stops per configuration; subsequent Enable should not cause jumps (since CspTargetPosition was latched to the current actual position at Attach).

Verification:

- Use an oscilloscope or external encoder to observe actual motor displacement and verify a stable rate (= Increment ÷ Cycle Time).
- Vary the CSP Position Increment value or sign and observe Position Actual Value tracking in real time.
- Use `getFollowingErrorActualValue()` to confirm Following Error is within a reasonable range.

Common Issues:

- After Enable, the motor jumps suddenly or shows excessive Following Error: typically CspTargetPosition does not match the current actual position. This tool automatically latches Position Actual Value to CspTargetPosition on Attach; if jumps persist, Disable and re-Attach.
- Position Actual Value does not change: verify CSP Position Increment has been written via Set, the State is Operation Enabled, and the drive is actually excited; also check whether Following Error Window is too small, causing the drive to automatically transition to Fault.
- The rate doesn't match the calculated value: Cycle Time may be misconfigured, or the PDO is not updating each cycle; verify the MainDevice cycle using ESC Registers 0x0910 System Time.
- Fault occurs after Enable (non-zero Error Code): common causes are software limit exceeded, excessive Following Error, or missing DC synchronization causing interpolation failure. Read Error Code (603Fh), consult the drive error table, and verify Scenario C's DC synchronization is complete.

> **Note** *CSP mode is continuous; once enabled, the motor immediately moves. Always verify machine travel and E-Stop safety. Begin trial runs with a small Increment (e.g., 1–20 counts) and increase gradually.*

12.6 Scenario F — Object Dictionary Browsing and SDO Read/Write

Purpose: Use the CoE tab to perform SDO Upload (read) and SDO Download (write) for accessing the SubDevice's object dictionary; use OD Generation to import the full object dictionary structure for browsing in one go; and observe Emergency Messages for fault diagnosis. This scenario covers the most common acyclic data access methods used during drive commissioning, PDO presetting, and error troubleshooting.

Prerequisites:

- Scenario A is completed; SubDevice identification can be read from the Information tab.
- The SubDevice is in Pre-Op or higher — SDO communication requires Mailbox to be enabled; access is not possible in Init state.
- On the Information tab, confirm CoE is checked under Mailbox Protocol; if OD Generation is needed, Enable SDO Info must be checked under CoE Details.
- The Index / SubIndex of interest is known; obtainable from the vendor datasheet, ESI file, or generated automatically by OD Generation.

Steps:

(A) Browsing the Full Object Dictionary

1. When first entering the CoE tab, the Object Dictionary table and Emergency Message block are both empty.

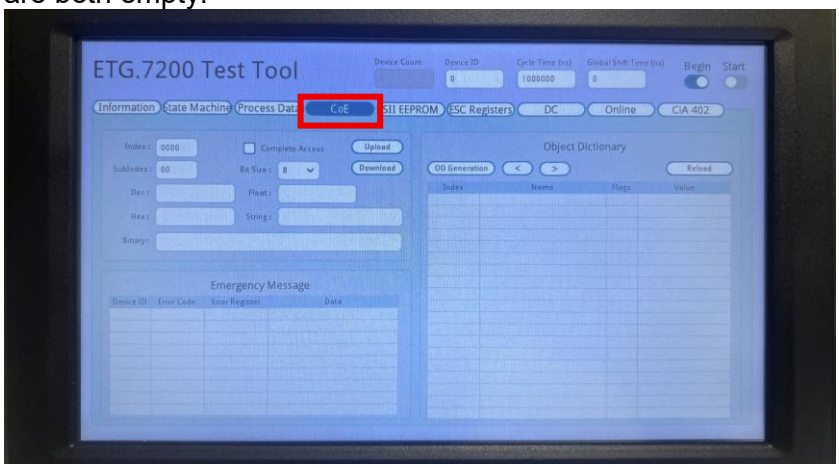


Figure 12.6-1 Initial CoE tab: Object Dictionary and Emergency Message are both empty.

2. Click the OD Generation button at the top right. The tool sequentially calls `getODlist()`, `getObjectDescription()`, and `getEntryDescription()` to read the SubDevice's full dictionary structure. Depending on the object count, this may take 1–10 seconds.

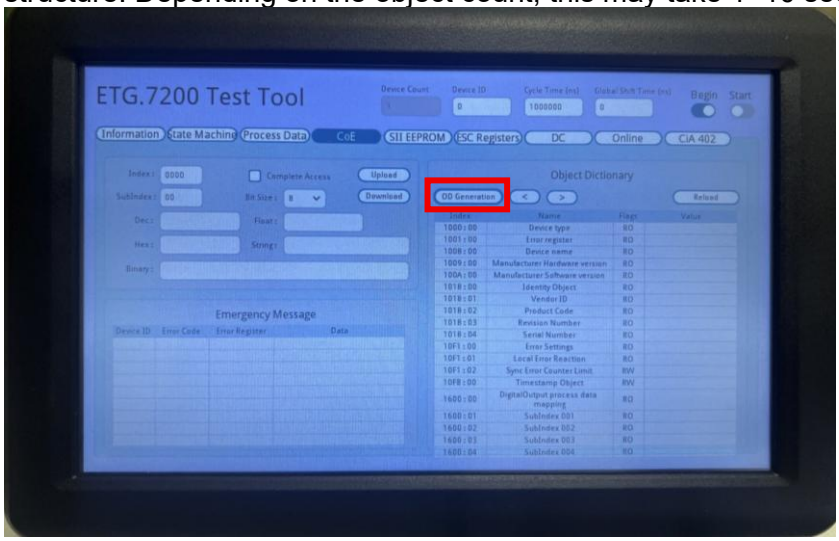


Figure 12.6-2 After OD Generation, Object Dictionary lists all Index / Name / Flags, but the Value column is still empty.

3. Click the Reload button. The tool issues SDO Upload for each object one by one and fills in the Value column.
4. Scroll or use the < / > buttons to page through and browse Index, Name, Flags (RO/RW), and Value columns; common objects such as 1000h Device Type, 1018h Identity Object, and 6064h Position Actual Value can all be inspected here.

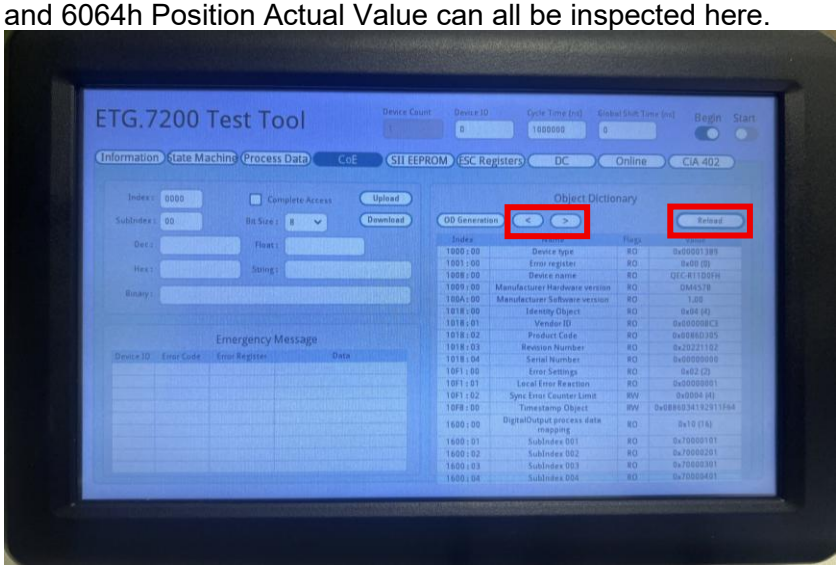


Figure 12.6-3 After Reload, Object Dictionary displays the current values of all objects, allowing direct inspection of RO/RW flags and contents.

(B) Reading a Single Object (SDO Upload)

1. In the SDO operation panel at the top left, fill in Index (e.g., 6041h, CiA 402 Statusword) and SubIndex.

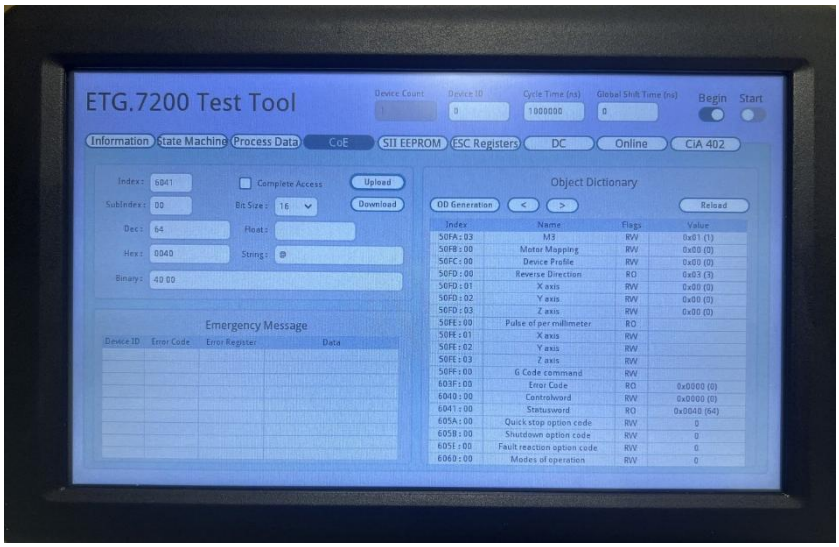


Figure 12.6-4 SDO Upload example: entering the object to read.

2. Set Bit Size — choose 8 / 16 / 32 / 64 from the dropdown (16 in this example). To read an entire compound object in one transfer, check Complete Access.
3. Click the Upload button. The tool calls `sdoUpload()` to retrieve the object's current value.

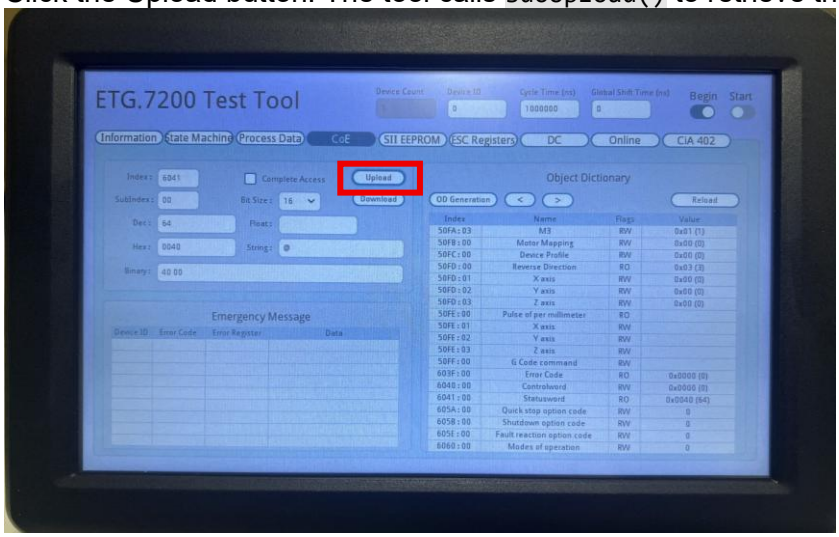


Figure 12.6-5 SDO Upload example: reading 6041h Statusword; Dec=64, Hex=0040, Binary=40 00. The Object Dictionary on the right also shows 6041:00 as 0x0040 (64).

4. The result is shown simultaneously in Dec, Hex, Float, String, and Binary formats for cross-reference as needed.

(C) Writing a Single Object (SDO Download)

1. Likewise, fill in Index / SubIndex / Bit Size. Commonly writable objects include 6065h Following Error Window, 6080h Max Motor Speed, and 605Ah Quick Stop Option Code.

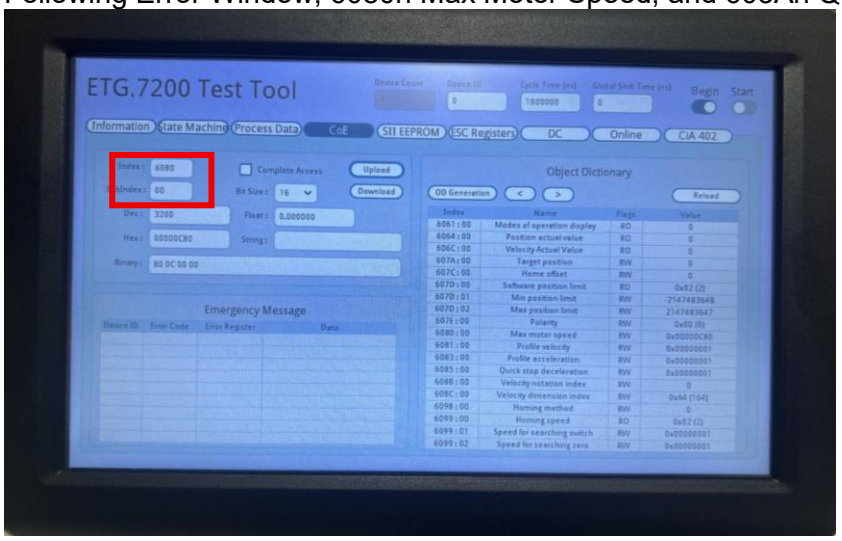


Figure 12.6-6 Before writing, Upload 6080h to obtain the old value: Hex=00000C80, Dec=3200.

2. Tap the Dec or Hex field; the HMI brings up a numeric keypad to enter the new value.

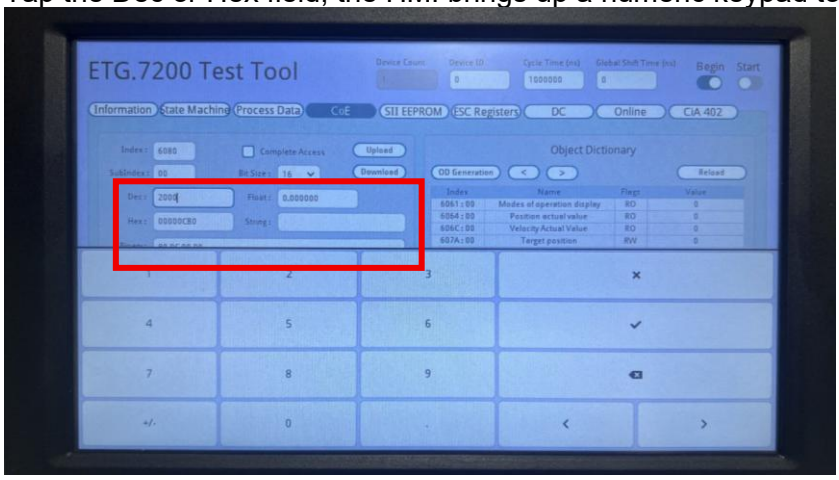


Figure 12.6-7 After tapping the Dec field, the keypad appears; entering new value 2000.

3. Click the Download button; the tool calls `sdoDownload()` to write, and a status dialog appears.

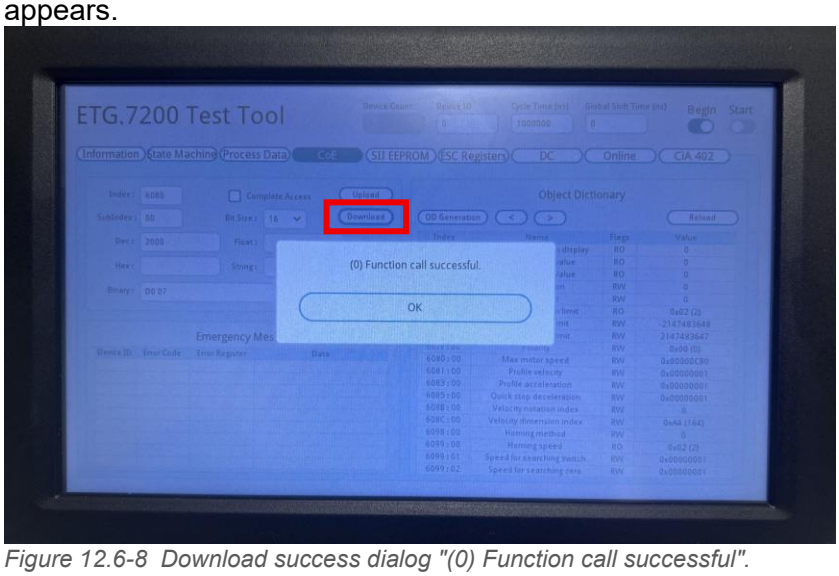


Figure 12.6-8 Download success dialog "(0) Function call successful".

4. Immediately click Upload to verify the write; the Hex field should now show 000007D0 (= 2000), and the value of 6080:00 in the Object Dictionary is also updated to 0x000007D0.

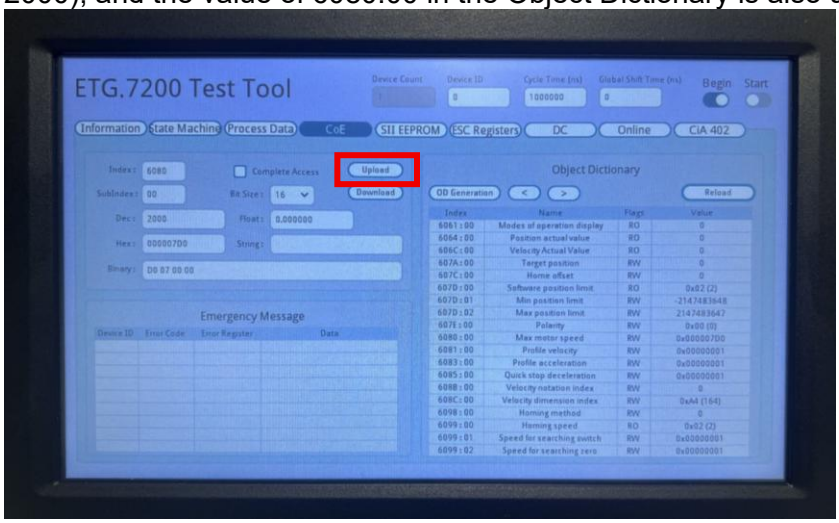


Figure 12.6-9 After writing, Upload again: Hex = 000007D0 = 2000; the Object Dictionary on the right shows 6080:00 updated synchronously.

(D) Observing Emergency Messages

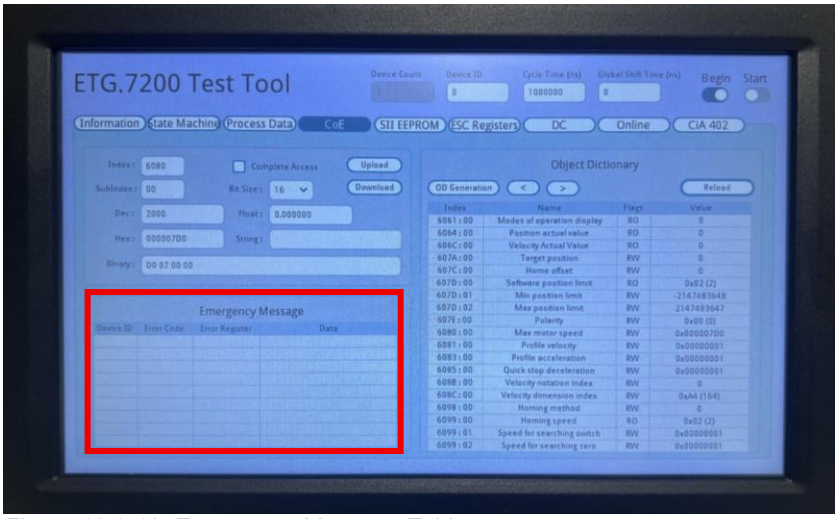


Figure 12.6-10 Emergency Message Table.

When the SubDevice detects an internal error (overcurrent, excitation failure, limit triggered, etc.), it automatically sends an Emergency Message via CoE; the tool inserts each message into the table below in real time. Each entry contains:

- Device ID — the SubDevice ID that originated the message.
- Error Code — CiA 301 standard error code (16 bits), defining the error category.
- Error Register — corresponds to object 1001h; the bit map indicates current error categories (generic, current, voltage, temperature, communication, vendor-specific).
- Data — 5 bytes of vendor-specific extension data, dependent on drive specifications.

When troubleshooting drive faults, first inspect the Error Code and Error Register in the Emergency Message; cross-referencing the drive datasheet pinpoints the cause.

Expected Results:

- OD Generation completes successfully; the Object Dictionary lists all SubDevice objects (typically 200–600 entries for a common drive).
- After Reload, the Value column for each object is correctly filled in.
- Single-entry Upload returns values consistent with what OD Generation + Reload displays.
- After a Download write, an Upload reads back a value equal to the value written.
- The Emergency Message table is empty when the SubDevice is normal; errors appear in real time.

Verification:

- Immediately Upload after Download to compare; if writing to NVM is needed, trigger storage via 1010h Store Parameters.
- Cross-reference the vendor datasheet's "default value", "valid range", and "access permissions" to ensure the written content is legal.
- Reproduce the same operations in the MainDevice program using `sdoUpload()` / `sdoDownload();` note that these two functions cannot be called within a cyclic callback (they block).

Common Issues:

- Upload returns ECAT_ERR_DEVICE_COE_TIMEOUT (-2412): the SubDevice has not entered Pre-Op, or Mailbox communication is abnormal. Confirm State Machine shows Pre-Op or higher first.
- Download returns ECAT_ERR_DEVICE_COE_ERROR (-2413): the object is read-only, the value is out of range, or Bit Size is incorrect. Check the Access (RO/RW) field and Value Range in the datasheet.
- OD Generation returns ECAT_ERR_DEVICE_COE_SDO_INFO_NOT_SUPPORT (-2401): Enable SDO Info is not checked under CoE Details; some legacy drives or third-party

vendors don't support it — fall back to manually entering Index from the datasheet for SDO Upload.

- Upload shows the old value after writing: writes must be stored in NVM to persist across restarts — trigger storage by SDO Downloading 1010h:01h = "save" (0x65766173); some objects only accept writes in specific states (e.g., Pre-Op, not Operation Enabled).
- Reload hangs or takes very long: the combination of many objects and a slow SubDevice. Manually Upload the objects you care about, or reduce the per-transfer Bit Size.

Note *Writing to the object dictionary changes SubDevice behavior in real time. Mistakes on critical objects such as PDO mapping (1600h/1A00h), Cycle Time, or Following Error Window can lead to runaway motion or loss of communication. Upload to back up the original value first, and avoid Download while the motor is Operation Enabled.*

13. EtherCAT API Reference

13.1 EtherCAT API

Functions:

Function Name	Description	Callback Available
Initialization-related functions		
`begin()`	Initialize the EtherCAT MDevice.	
`end()`	Deinitialize the EtherCAT MDevice.	
`isRedundancy()`	Check if the EtherCAT MDevice has cable redundancy enabled.	O
`libraryVersion()`	Get the EtherCAT MDevice library version.	O
`firmwareVersion()`	Get the EtherCAT firmware version.	O
`readSettings()`	Read the current EtherCAT MDevice settings.	
`saveSettings()`	Save the EtherCAT MDevice settings.	
Control-related functions		
`start()`	Start the EtherCAT MDevice.	
`stop()`	Stop the EtherCAT MDevice.	
`update()`	Update process data and handle acyclic commands.	O
`setShiftTime()`	Set the Global Shift Time for DC-Synchronous mode.	
`getShiftTime()`	Get the Global Shift Time for DC-Synchronous mode.	O
`getSystemTime()`	Get the system time of the current cycle.	O
`getWorkingCounter()`	Get the working counter for the current cycle.	O
`getExpectedWorkingCounter()`	Get the expected working counter.	O
Callback-related functions		
`attachCyclicCallback()`	Register a cyclic callback.	
`detachCyclicCallback()`	Unregister cyclic callback.	
`attachErrorCallback()`	Register an error callback.	
`detachErrorCallback()`	Unregister error callback.	
`attachEventCallback()`	Register an event callback.	
`detachEventCallback()`	Unregister event callback.	
`errGetCableBrokenLocation1()`	Get the cable broken location 1 in error callback.	O[1]
`errGetCableBrokenLocation2()`	Get the cable broken location 2 in error callback.	O[1]
`evtGetMasterState()`	Get the EtherCAT MDevice state in event callback.	O[2]
SubDevice information related functions		
`getSlaveCount()`	Get the number of SubDevices on the network.	O
`getVendorID()`	Get the vendor ID of the specified device.	O
`getProductCode()`	Get the product code of the specified device.	O
`getRevisionNumber()`	Get the revision number of the specified device.	O
`getSerialNumber()`	Get the serial number of the specified device.	O
`getAliasAddress()`	Get the alias address of the specified device.	O
`getSlaveNo()`	Find the sequence number of the matching EtherCAT SubDevice on the network.	O

- **Note 1**: This function can only be called in error callback.
- **Note 2**: This function can only be called in event callback.

13.2 SDO / PDO / ESC / SII / FoE API

Functions:

Function Name	Description	Callback Available
SubDevice information related functions		
`getVendorID()`	Get the vendor ID.	O
`getProductCode()`	Get the product code.	O
`getRevisionNumber()`	Get the revision number.	O
`getSerialNumber()`	Get the serial number.	O
`getAliasAddress()`	Get the alias address.	O
`getSlaveNo()`	Get the sequence ID on the EtherCAT network.	O
`getDeviceName()`	Get the device name.	O
`getMailboxProtocol()`	Get the supported mailbox protocol types.	O
`getCoEDetails()`	Get the details about CoE supported.	O
`getFoEDetails()`	Get the details about FoE supported.	O
`getEoEDetails()`	Get the details about EoE supported.	O
`getSoEChannels()`	Get the number of SoE channels supported.	O
`isSupportDC()`	Check if the EtherCAT SubDevice has DC supported.	O
PDO access functions		
`pdoBitWrite()`	Write 1-bit output process data.	O
`pdoBitRead()`	Read 1-bit input process data.	O
`pdoGetOutputBuffer()`	Get the memory pointer of output process data.	O
`pdoGetInputBuffer()`	Get the memory pointer of input process data.	O
`pdoWrite()`	Write multiple bytes of output process data.	O
`pdoWrite8()`	Write 8-bit output process data.	O
`pdoWrite16()`	Write 16-bit output process data.	O
`pdoWrite32()`	Write 32-bit output process data.	O
`pdoWrite64()`	Write 64-bit output process data.	O
`pdoRead()`	Read multiple bytes of input process data.	O
`pdoRead8()`	Read 8-bit input process data.	O
`pdoRead16()`	Read 16-bit input process data.	O
`pdoRead32()`	Read 32-bit input process data.	O
`pdoRead64()`	Read 64-bit input process data.	O
CoE communication functions		
`sdoDownload()`	Write multiple bytes of data to the object.	
`sdoDownload8()`	Write 8-bit value to the object.	
`sdoDownload16()`	Write 16-bit value to the object.	
`sdoDownload32()`	Write 32-bit value to the object.	
`sdoDownload64()`	Write 64-bit value to the object.	
`sdoUpload()`	Read multiple bytes of data from the object.	
`sdoUpload8()`	Read 8-bit value from the object.	
`sdoUpload16()`	Read 16-bit value from the object.	
`sdoUpload32()`	Read 32-bit value from the object.	
`sdoUpload64()`	Read 64-bit value from the object.	
`getODlist()`	Get a list of objects existing in the object dictionary.	
`getObjectDescription()`	Get the object description of the object.	
`getEntryDescription()`	Get the entry description of the object.	
FoE communication functions		
`readFoE()`	Read a file from the EtherCAT SubDevice.	
`writeFoE()`	Write a file to the EtherCAT SubDevice.	
DC configuration functions		
`setDc()`	Configure DC parameters.	
SII EEPROM access functions		
`writeSII()`	Write multiple bytes of data to the SII EEPROM.	
`writeSII8()`	Write 8-bit value to the SII EEPROM.	
`writeSII16()`	Write 16-bit value to the SII EEPROM.	
`writeSII32()`	Write 32-bit value to the SII EEPROM.	
`readSII()`	Read multiple bytes of data from the SII EEPROM.	

`readSII8()`	Read 8-bit value from the SII EEPROM.	
`readSII16()`	Read 16-bit value from the SII EEPROM.	
`readSII32()`	Read 32-bit value from the SII EEPROM.	

13.3 CiA 402 API

Functions:

Function Name	Description	Callback Available
Initialization-related functions		
`attach()`	Initialize the object of this EtherCAT SubDevice class.	
`detach()`	Deinitialize the object of this EtherCAT SubDevice class.	
`isStepper()`	Check if the EtherCAT SubDevice is a stepper motor drive.	O
Control-related functions		
`getCiA402Mode()`	Get the current mode of operation. (6061_h)	O[1]
`setCiA402Mode()`	Switch the mode of operation. (6060_h, 6061_h, 6502_h)	O[1, 2]
`getCiA402State()`	Get the current CiA 402 state. (6041_h)	O
`setCiA402State()`	Switch the CiA 402 state. (6040h, 6041_h)	O[2]
`enable()`	Enable the drive function and power on the motor. (6040_h, 6041_h)	O[2]
`disable()`	Disable the drive function and power off the motor. (6040_h, 6041_h)	O[2]
Operation-related functions		
`setTargetPosition()`	Set the target position. ($607A_h$)	O[1]
`setTargetVelocity()`	Set the target velocity. ($60FF_h$)	O[1]
`setTargetTorque()`	Set the target torque. (6071_h)	O[1]
`setProfileAcceleration()`	Set the profile acceleration. (6083_h)	O[1]
`setProfileDeceleration()`	Set the profile deceleration. (6084_h)	O[1]
`setMaxAcceleration()`	Set the max acceleration. ($60C5_h$)	O[1]
`setMaxDeceleration()`	Set the max deceleration. ($60C6_h$)	O[1]
`setMaxProfileVelocity()`	Set the max profile velocity. ($607F_h$)	O[1]
`setMotionProfileType()`	Set the motion profile type. (6086_h)	O[1]
`setPositionWindow()`	Set the position window. (6067_h)	O[1]
`setPositionWindowTime()`	Set the position window time. (6068_h)	O[1]
`setPositionOffset()`	Set the position offset. ($60B0_h$)	O[1]
`setSoftwarePositionLimit()`	Set the software position limit. ($607D_h$)	O[1]
`setFollowingErrorWindow()`	Set the following error window. (6065_h)	O[1]
`setPositionPolarity()`	Set the position polarity. ($607E_h$)	O[1]
`setVelocityWindow()`	Set the velocity window. ($606D_h$)	O[1]
`setVelocityWindowTime()`	Set the velocity window time. ($606E_h$)	O[1]
`setVelocityThreshold()`	Set the velocity threshold. ($606F_h$)	O[1]
`setVelocityOffset()`	Set the velocity offset. ($60B1_h$)	O[1]
`setMaxMotorSpeed()`	Set the max motor speed. (6080_h)	O[1]
`setVelocityPolarity()`	Set the velocity polarity. ($607E_h$)	O[1]
`setTorqueOffset()`	Set the torque offset. ($60B2_h$)	O[1]
`setMaxTorque()`	Set the max torque. (6072_h)	O[1]
`setPositiveTorqueLimit()`	Set the positive torque limit. ($60E0_h$)	O[1]
`setNegativeTorqueLimit()`	Set the negative torque limit. ($60E1_h$)	O[1]
`setQuickStopDeceleration()`	Set the quick stop deceleration. (6085_h)	O[1]
`setQuickStopOptionCode()`	Set the quick stop option code. ($605A_h$)	
`setShutdownOptionCode()`	Set the shutdown option code. ($605B_h$)	
`setDisableOperationOptionCode()`	Set the disable operation option code. ($605C_h$)	
`setHaltOptionCode()`	Set the halt option code. ($605D_h$)	
`setFaultReactionOptionCode()`	Set the fault reaction option code. ($605E_h$)	
`getErrorCode()`	Get the error code. ($603F_h$)	O[1]
`getSupportedDriveModes()`	Get the supported drive modes. (6502_h)	O[1]
`getMotorResolution()`	Get the motor resolution. ($60EF_h$)	
`getPositionActualValue()`	Get the position actual value. (6064_h)	O[1]
`getVelocityActualValue()`	Get the velocity actual value. ($606C_h$)	O[1]

getTorqueActualValue()	Get the torque actual value. (6077h)	O[1]
getCurrentActualValue()	Get the current actual value. (6078h)	O[1]
getPositionDemandValue()	Get the position demand value. (6062h)	O[1]
getPositionDemandInternalValue()	Get the position demand internal value. (60FCh)	O[1]
getPositionActualInternalValue()	Get the position actual internal value. (6063h)	O[1]
getAdditionalPositionActualValue()	Get the additional position actual value. (60E4h)	O[1]
getFollowingErrorActualValue()	Get the following error actual value. (60F4h)	O[1]
getVelocityDemandValue()	Get the velocity demand value. (606Bh)	O[1]
getTorqueDemandValue()	Get the torque demand value. (6074h)	O[1]
getDigitalInputs()	Get the digital inputs. (60FDh)	O[1]
Profile Position mode (pp) related functions		
pp_SetVelocity()	Set the profile velocity. (6081h)	
pp_SetAcceleration()	Set the profile acceleration. (6083h)	
pp_SetDeceleration()	Set the profile deceleration. (6084h)	
pp_SetMotionProfileType()	Set the motion profile type. (6086h)	
pp_Run()	Move to the target position. (6040h, 6041h, 607Ah)	
pp_IsTargetReached()	Check if the target position has been reached. (6041h)	O
pp_CheckFollowingError()	Check if the following error occurs. (6041h)	O
pp_Halt()	Pause the current operation. (6040h, 6041h)	
pp_Resume()	Resume the paused operation. (6040h, 6041h)	
Profile Velocity mode (pv) related functions		
pv_SetAcceleration()	Set the profile acceleration. (6083h)	
pv_SetDeceleration()	Set the profile deceleration. (6084h)	
pv_SetMotionProfileType()	Set the motion profile type. (6086h)	
pv_Run()	Move at a target velocity continuously. (6041h, 60FFh)	
pv_IsTargetReached()	Check if the target velocity has been reached. (6041h)	O
pv_CheckZeroSpeed()	Check if the speed is zero. (6041h)	O
pv_CheckMaxSlippageError()	Check if the maximum slippage error occurs. (6041h)	O
pv_Halt()	Pause the current operation. (6040h, 6041h)	
pv_Resume()	Resume the paused operation. (6040h, 6041h)	
Profile Torque mode (tq) related functions		
tq_SetTorqueSlope()	Set the torque slope. (6087h)	
tq_SetTorqueProfileType()	Set the torque profile type. (6088h)	
tq_SetMotorRatedCurrent()	Set the motor rated current. (6075h)	
tq_SetMotorRatedTorque()	Set the motor rated torque. (6076h)	
tq_Run()	Drive continuously at the target torque. (6041h, 6071h)	
tq_IsTargetReached()	Check if the target torque has been reached. (6041h)	O
tq_Halt()	Pause the current operation. (6040h, 6041h)	
tq_Resume()	Resume the paused operation. (6040h, 6041h)	
Homing mode (hm) related functions		
hm_SetHomeOffset()	Set the home offset. (607Ch)	
hm_SetHomingMethod()	Set the homing method. (6098h)	
hm_SetHomingSpeeds()	Set the homing speeds. (6099h)	
hm_SetHomingAcceleration()	Set the homing acceleration. (609Ah)	
hm_Run()	Initiate a homing operation. (6040h, 6041h)	
hm_IsAttained()	Check the status of the homing operation. (6041h)	O
hm_Stop()	Stop the homing operation. (6040h, 6041h)	
Function Group "Touch Probe" related functions		
enableTouchProbe1()	Enable the touch probe 1. (60B8h, 60B9h, 60D0h)	
enableTouchProbe2()	Enable the touch probe 2. (60B8h, 60B9h, 60D0h)	
disableTouchProbe1()	Disable the touch probe 1. (60B8h, 60B9h)	
disableTouchProbe2()	Disable the touch probe 2. (60B8h, 60B9h)	
isTouchProbe1ValueReady()	Check if a positive or negative edge has occurred on the touch probe 1 signal. (60B9h)	O[1]

`isTouchProbe2ValueReady()`	Check if a positive or negative edge has occurred on the touch probe 2 signal. ($60B9_h$)	O[1]
`readTouchProbe1Value()`	Read the touch probe position 1. ($60BA_h$, $60BB_h$)	O[1]
`readTouchProbe2Value()`	Read the touch probe position 2. ($60BC_h$, $60BD_h$)	O[1]
Low-level functions for mode-specific flow control		
`setHaltBit()`	Set the halt bit in the controlword. (6040_h)	O
`isTargetReached()`	Check if the target has reached. (6041_h)	O
`setModeSpecificBit4()`	Set the bit 4 in the controlword. (6040_h)	O
`setModeSpecificBit5()`	Set the bit 5 in the controlword. (6040_h)	O
`setModeSpecificBit6()`	Set the bit 6 in the controlword. (6040_h)	O
`checkModeSpecificBit12()`	Check the value of bit 12 in the statusword. (6041_h)	O
`checkModeSpecificBit13()`	Check the value of bit 13 in the statusword. (6041_h)	O

- **Note 1:** This function can be used in callback functions if the related object is mapped to PDO.
- **Note 2:** This function will ignore the timeout parameter and will not wait for the actual value to match the set value when used in a callback.

14. Error Code Reference Table

14.1 EtherCAT API Error Codes

For most functions, a return value less than zero indicates an error and represents an error code. If an error code exists, you can find the cause and corresponding corrective action below.

Definition	Code
ECAT_SUCCESS	0
ECAT_ERR_MODULE_INIT_FAIL	-100
ECAT_ERR_MODULE_GET_VERSION_FAIL	-101
ECAT_ERR_MODULE_VERSION_MISMATCH	-102
ECAT_ERR_MODULE_GENERIC_TRANSFER_INIT_FAIL	-103
ECAT_ERR_MASTER_DOWNLOAD_SETTINGS_FAIL	-200
ECAT_ERR_MASTER_SET_DEVICE_SETTINGS_FAIL	-201
ECAT_ERR_MASTER_GET_GROUP_INFO_FAIL	-202
ECAT_ERR_MASTER_GET_MASTER_INFO_FAIL	-203
ECAT_ERR_MASTER_GET_DEVICE_INFO_FAIL	-204
ECAT_ERR_MASTER_SET_GROUP_SETTINGS_FAIL	-205
ECAT_ERR_MASTER_MAPPING_INIT_FAIL	-206
ECAT_ERR_MASTER_INTERRUPT_INIT_FAIL	-207
ECAT_ERR_MASTER_ACTIVE_FAIL	-208
ECAT_ERR_MASTER_ENI_INITCMDS_FAIL	-209
ECAT_ERR_MASTER_NO_DEVICE	-210
ECAT_ERR_MASTER_ACYCLIC_INIT_FAIL	-300
ECAT_ERR_MASTER_ACYCLIC_REQUEST_FAIL	-301
ECAT_ERR_MASTER_ACYCLIC_BUSY	-302
ECAT_ERR_MASTER_ACYCLIC_TIMEOUT	-303
ECAT_ERR_MASTER_ACYCLIC_ERROR	-304
ECAT_ERR_MASTER_ACYCLIC_WRONG_STATUS	-405
ECAT_ERR_MASTER_GENERIC_SEND_FAIL	-400
ECAT_ERR_MASTER_GENERIC_RECV_FAIL	-401
ECAT_ERR_MASTER_NOT_BEGIN	-1000
ECAT_ERR_MASTER_WRONG_BUFFER_SIZE	-1001
ECAT_ERR_MASTER_REDUNDANCY_NO_DC	-1002
ECAT_ERR_MASTER_MEMORY_ALLOCATION_FAIL	-1003
ECAT_ERR_MASTER_OSLAYER_INIT_FAIL	-1004
ECAT_ERR_MASTER_NIC_INIT_FAIL	-1005
ECAT_ERR_MASTER_BASE_INIT_FAIL	-1006
ECAT_ERR_MASTER_CIA402_INIT_FAIL	-1007
ECAT_ERR_MASTER_SETUP_PDO_FAIL	-1008
ECAT_ERR_MASTER_SCAN_NETWORK_FAIL	-1009
ECAT_ERR_MASTER_START_MASTER_FAIL	-1010
ECAT_ERR_MASTER_CYCLETIME_TOO_SMALL	-1011
ECAT_ERR_MASTER_DUMP_OUTPUT_PDO_FAIL	-1012
ECAT_ERR_MASTER_CONFIG_DEVICE_FAIL	-1013
ECAT_ERR_MASTER_CONFIG_MAPPING_FAIL	-1014
ECAT_ERR_MASTER_WAIT_BUS_SYNC_TIMEOUT	-1015
ECAT_ERR_MASTER_WAIT_MASTER_SYNC_TIMEOUT	-1016
ECAT_ERR_MASTER_CYCLIC_START_FAIL	-1017
ECAT_ERR_MASTER_WRONG_BUFFER_POINTER	-1018
ECAT_ERR_MASTER_ENI_INIT_FAIL	-1050
ECAT_ERR_MASTER_ENI_MISMATCH	-1051
ECAT_ERR_MASTER_STOPPED	-1100
ECAT_ERR_MASTER_STARTED	-1101
ECAT_ERR_MASTER_NOT_IN_PREOP	-1102
ECAT_ERR_MASTER_NOT_IN_SAFEOP	-1103
ECAT_ERR_MASTER_NOT_IN_OP	-1104
ECAT_ERR_MASTER_II_TRANSITION_FAIL	-1200
ECAT_ERR_MASTER_IP_TRANSITION_FAIL	-1201
ECAT_ERR_MASTER_PS_TRANSITION_FAIL	-1202
ECAT_ERR_MASTER_SO_TRANSITION_FAIL	-1203

ECAT_ERR_DEVICE_NOT_EXIST	-2000
ECAT_ERR_DEVICE_NOT_ATTACH	-2001
ECAT_ERR_DEVICE_NO_MAILBOX	-2002
ECAT_ERR_DEVICE_NO_DC	-2003
ECAT_ERR_DEVICE_WRONG_INPUT	-2004
ECAT_ERR_DEVICE_MEMORY_ALLOCATION_FAIL	-2005
ECAT_ERR_DEVICE_VENDOR_ID_MISMATCH	-2006
ECAT_ERR_DEVICE_PRODUCT_CODE_MISMATCH	-2007
ECAT_ERR_DEVICE_NO_SUCH_FUNCTION	-2008
ECAT_ERR_DEVICE_FUNCTION_NOT_INIT	-2009
ECAT_ERR_DEVICE_BUSY	-2010
ECAT_ERR_DEVICE_TIMEOUT	-2011
ECAT_ERR_DEVICE_NO_DATA	-2012
ECAT_ERR_DEVICE_SII_READ_FAIL	-2100
ECAT_ERR_DEVICE_SII_WRITE_FAIL	-2101
ECAT_ERR_DEVICE_PDO_NOT_EXIST	-2200
ECAT_ERR_DEVICE_PDO_OUT_OF_RANGE	-2201
ECAT_ERR_DEVICE_FOE_NOT_SUPPORT	-2300
ECAT_ERR_DEVICE_FOE_REQUEST_FAIL	-2310
ECAT_ERR_DEVICE_FOE_TIMEOUT	-2311
ECAT_ERR_DEVICE_FOE_ERROR	-2312
ECAT_ERR_DEVICE_FOE_BUFFER_TOO_SMALL	-2313
ECAT_ERR_DEVICE_FOE_READ_FAIL	-2314
ECAT_ERR_DEVICE_FOE_WRITE_FAIL	-2315
ECAT_ERR_DEVICE_COE_SDO_NOT_SUPPORT	-2400
ECAT_ERR_DEVICE_COE_SDO_INFO_NOT_SUPPORT	-2401
ECAT_ERR_DEVICE_COE_BUSY	-2410
ECAT_ERR_DEVICE_COE_REQUEST_FAIL	-2411
ECAT_ERR_DEVICE_COE_TIMEOUT	-2412
ECAT_ERR_DEVICE_COE_ERROR	-2413
ECAT_ERR_DEVICE_CIA402_NOT_EXIST	-2500
ECAT_ERR_DEVICE_CIA402_ADD_FAIL	-2501
ECAT_ERR_DEVICE_CIA402_TYPE_MISMATCH	-2502
ECAT_ERR_DEVICE_CIA402_NO_MODE_SUPPORT	-2503
ECAT_ERR_DEVICE_CIA402_WRONG_MODE	-2504
ECAT_ERR_DEVICE_CIA402_MODE_NOT_SUPPORT	-2505
ECAT_ERR_DEVICE_CIA402_CHANGE_WRONG_STATE	-2506
ECAT_ERR_DEVICE_CIA402_WRITE_OBJECT_FAIL	-2507
ECAT_ERR_DEVICE_CIA402_NO_SUCH_TOUCH_PROBE	-2580
ECAT_ERR_DEVICE_CIA402_NO_SUCH_TOUCH_PROBE_SOURCE	-2581
ECAT_ERR_DEVICE_EOE_NOT_SUPPORT	-2600
ECAT_ERR_DEVICE_EOE_NO_SUCH_PORT	-2601
ECAT_ERR_DEVICE_EOE_TOO_MUCH_CONTENT	-2602
ECAT_ERR_DEVICE_EOE_BUSY	-2610
ECAT_ERR_DEVICE_EOE_REQUEST_FAIL	-2611
ECAT_ERR_DEVICE_EOE_TIMEOUT	-2612
ECAT_ERR_GROUP_WRONG_INPUT	-3000
ECAT_ERR_GROUP_NOT_ATTACH	-3001