Junolane MotionPLC Detailed User Manual

ZE V4.23 Document R1.23 Hardware 24Y and 32Y

Prepared on 25 October 2026

This manual is for operators, commissioning engineers and local administrators. It explains device identification, MCU connections, task editing and download, project templates, vision compensation, updates and troubleshooting. New users should read Chapters 1 to 5 first; programmers should also read Chapters 6 to 12.

This manual applies to Junolane MotionPLC V4.23 and covers machine-name management, Chinese and English switching, task programming, offline project templates and device maintenance. Before use, check the Host version, MCU version and 24Y or 32Y hardware model. After a software version change, use the manual for that version.

Document status V4.23 release. Win7 supports serial and network; BLE is available only on Win10/11. Published at the owner's request without actual-machine acceptance; target-system and hardware results are not claimed.

Applicable executable D:\Program Files\PLC-F103ZET\LYXMotionPLC.exe

Instructions include operating sequences, parameter meanings, effects, expected results and diagnostic checks. Example values explain the workflow; confirm travel, wiring and motion conditions for the actual machine before using them.

Reading guide

1 Scope and basic concepts

2 Installation startup and permissions

3 Device list and language switching

4 Connection channels and machine management

5 Machine window and configuration mode

6 Task editing and program data

7 Points and coordinates

8 Program instruction reference

9 Marks functions variables and timers

10 System parameters and motor operating mode

11 Vision configuration and compensation

12 Project task and section templates

13 Host automation programs

14 Monitoring and operation

15 Host and MCU updates

16 Data backup and recovery

17 Troubleshooting

18 Commissioning examples and handover

Appendix Version checks and revision record

In Word, hold Ctrl and click a chapter name to open that chapter. The Reading guide link in each page footer returns here.

For a first connection, identify the machine and select its channel, then choose which program to load in Chapters 2 to 5. Before writing, check points and settings, edit the task, and use Save / Download to choose database-only saving or saving with MCU download. Use Chapter 12 for reusable templates and offline template editing.

1 Scope and basic concepts

1.1 Host MCU and peripheral responsibilities

The Windows Host provides the device list, machine window, program editor, templates and configured database. The MCU executes downloaded tasks, handles I/O and pulse motion, and supplies real-time status and safety checks. Successful editing on the PC does not change the official MCU task until the corresponding download and readback have completed.

The current device center supports local COM, BLE Central and ESP TCP channels. An ESP or BLE module is a communication bridge; the MCU UID identifies the machine. Multiple channels to the same MCU are combined into one machine. Older architecture documents requiring a separate Backend relay do not describe this local version.

Object	Purpose	Identification
Host	Configuration, editing, monitoring, templates and local automation	Executable path and local account
MCU	Task execution, motion, I/O and heartbeat	MCU UID
Communication module	BLE or ESP bridge	BLE MAC or ESP endpoint
Machine registration	Names, approval and channels	Local MachineId associated with MCU UID

1.2 Three levels of program data

The Host editor, the database machine program linked by MCU UID and the MCU executable program are separate copies. Loading changes the editor; database saving updates the machine record; downloading updates the device. Templates are separate reusable records. Choose the intended source and save action explicitly, then verify the device when required.

1.3 Hardware and Windows support

Firmware is built separately for ZET 24Y and 32Y; select the matching model. Host has separate self-contained Win7 SP1 x64 and Win10/11 x64 builds. As confirmed by the user on 25 October 2026, Win7 supports serial and network communication without BLE. BLE Central is available in the Win10/11 build.

24Y output indexes are Y0 to Y23; 32Y indexes are Y0 to Y31. Input indexes are X0 to X39 and motors are M0 to M7. Index names do not replace the terminal wiring diagram. Older devices without reported extension capabilities use the 24Y-compatible range; a displayed extended index does not prove that the machine has that output.

The integrator must verify wiring, home switches, drive enables and the hardwired emergency stop. A software emergency stop depends on the communication link and does not replace a hardware emergency-stop circuit. Confirm the work area and reset conditions before remote operation.

Win7 release acceptance requirements

Win7 BLE testing is not applicable under the user-confirmed V4.23 scope. On Win10/11, the BARROT Bluetooth 5.4 Adapter, USB VID_33FA / PID_0010, still requires driver, discovery, GATT notification/read/write, heartbeat, reconnection and USB replug acceptance. This does not establish Win7 BLE support.

Test the final outer Win7 Setup on clean Win7 SP1 x64: offline installation, first launch, original-account login, templates and automation, 100/125/150 percent DPI, serial/network (no BLE), camera/native libraries, automatic upgrade and failed-update rollback. Repeat after reboot and run for at least two hours. Use the Win7 self-contained runtime and offline prerequisites without installing developer tools. No Win7 runtime evidence is available yet. The complete matrix is in Docs/V4.23 发布前测试.md in the repository.

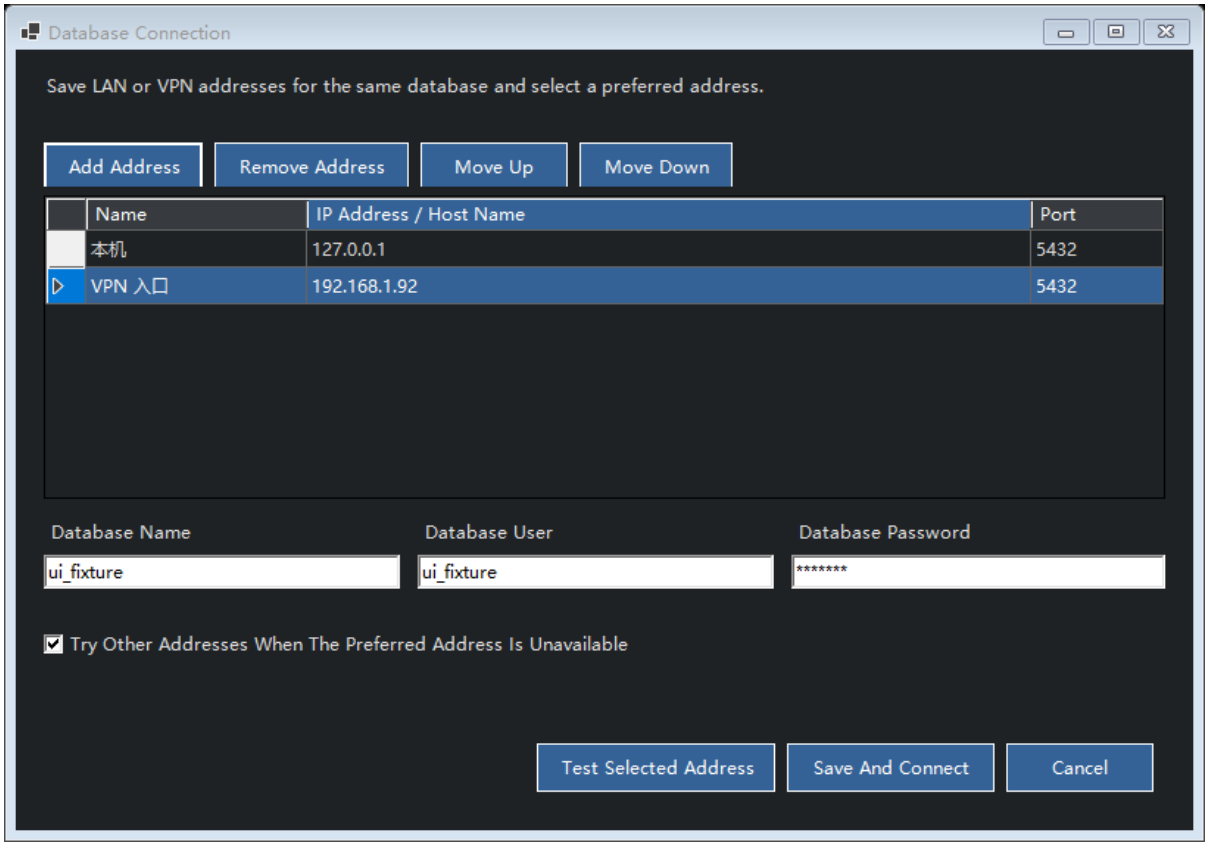
The official PostgreSQL 18 installer test-platform list omits Win7. The existing 18.6 installer is not proof of local Win7 database compatibility; verify its service, installer and restore tools separately. A remote database connection does not replace local one-click installation acceptance. Microsoft has ended modern .NET support for Win7. Record the runtime, OS patches and prerequisites actually tested. References: www.postgresql.org/download/windows/ and learn.microsoft.com/dotnet/core/install/windows.

2 Installation startup and permissions

2.1 Starting and checking the installation

1. Open LYXMotionPLC.exe from the operating installation. The verified local directory is D:\Program Files\PLC-F103ZET.

2. On first launch, enter the PostgreSQL server IP or host name, port (normally 5432), database name, username and password in Database Connection. Use Test Connection, then Save and Connect; login is available only after a successful connection. Cancelling setup exits the application. Later launches reuse the saved configuration; a connection failure opens the repair dialog. Select the server containing your existing machine data. This dialog does not import an old SQLite database or create a replacement database automatically.



Example test addresses only; enter the actual database server address.

3. Check total devices, online devices, pending approvals and BLE status. Zero counts are normal before devices have connected.

4. Check the language, machine name and Device ID before opening the corresponding HOST window.

Save changed machine programs with Save / Download before closing. Save as template creates a separate reusable template. Keep the Data directory and the server database: local Data contains connection settings, preferences and retained legacy files; PostgreSQL contains accounts, machines and program/template data. A software update must preserve both. Website and FTP accounts are separate from Host and database credentials.

PostgreSQL one-click preparation and existing accounts

Install in three steps: (1) the independent complete offline database kit, (2) the matching Windows Host Setup, (3) the ZET 24Y or 32Y application firmware. The database kit is delivered privately with authorized original data and credentials; it is not embedded in each Host Setup. Preserve existing databases during upgrades. Public Host and matching manual firmware downloads become available only after release acceptance.

Preserve the original Host application users, password hashes, roles, permissions and templates. For new local database installations, the user has authorized a new database connection password in the supplied database.json; the new postgres role uses the same installation password. Existing databases and credentials are not changed. The connection password is separate from Host login passwords and is not printed in this manual.

For a new installation, compare schema, template IDs/names/categories/contents and program data with the snapshot inventory. The verified materials contain 33 tables and seven templates; use the inventory accompanying any later snapshot. After reboot the service should start automatically and the Host reconnect with original credentials. Reinstallation and upgrades must retain the original database, passwords and edited data. Restore failures and port conflicts must be reported.

2.2 Roles and button permissions

Role or permission	Default use	How to check
Operator	Machine operation permission	Authorized operation and monitoring
Administrator	Machine operation and configuration	Parameters, tasks and templates
Super administrator	Protected highest-level identity	User management and protected account controls
Machine operation	Run, resume, pause and related entries	Also depends on connectivity and MCU state
Machine configuration	Approval, renaming, channels, parameters and templates	Controls are disabled without permission

Roles can be customized; these defaults do not prove the current account has a specific permission. A grey button may also mean an offline machine, missing MCU ID, unapproved or occupied channel, or an action rejected by the MCU state. Check these conditions before changing account permissions.

2.3 User management

A super administrator opens User Management from the device list to inspect accounts and roles. Use the available create, edit, role assignment and enable-state controls. Verify the resulting buttons with the intended account. Treat the protected super administrator differently from ordinary accounts. Do not put passwords in project notes, template descriptions or this manual.

3 Device list and language switching

3.1 Device list layout

The top area contains the Device List title, BLE switch/status and language controls. Cards show total devices, online devices and pending approvals. The toolbar includes Connect & Load Program, Switch Channel, Rename Machine, Linkage Programs, Database Maintenance and Program & Template, with approval, deletion and user-management controls according to permissions. Each machine row shows status, Device ID, name, counters, channel, ownership, MAC, RSSI and run controls. Device ID is the MCU UID; changing a module MAC or machine name does not change it. The list fits and centers within the current display working area, excluding the taskbar. Database Connection occupies its own row, aligned with and as wide as the first connection or firmware action below it. Other action buttons remain compact in one row.

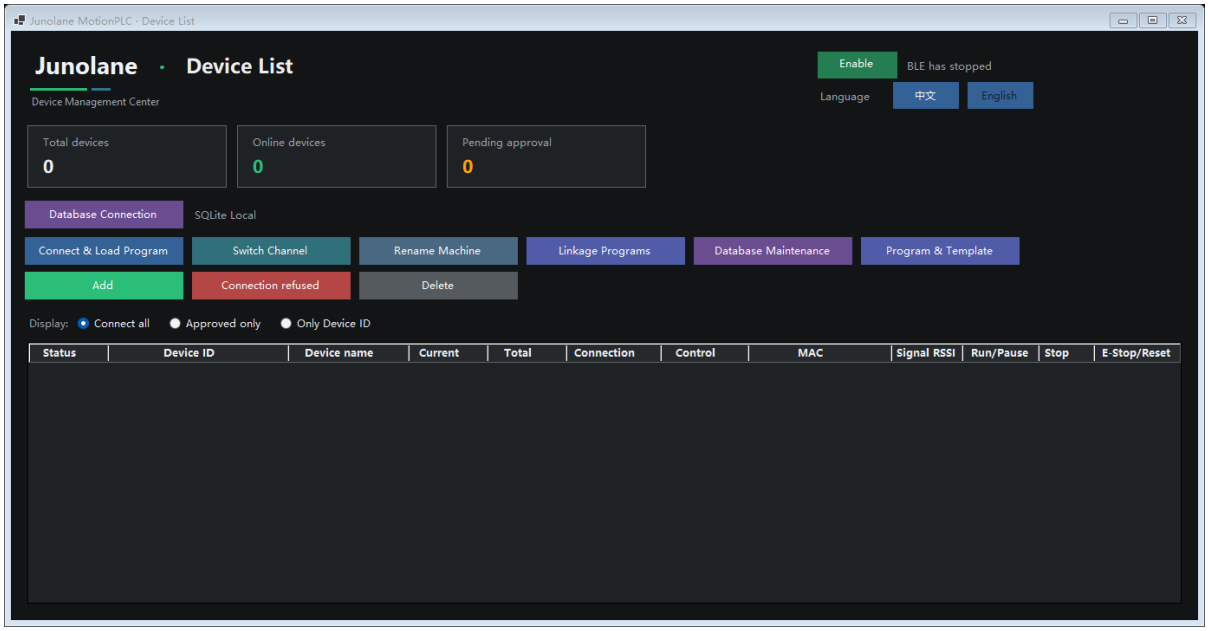


Figure 1 Current device list with no connected MCU in an isolated example

The device list shows Device Management Center, Device ID and Only Device ID. Figure 1 records the connections at capture time; device and connection counts change during operation.

3.2 Switching Chinese and English

1. Click the Chinese button labelled 中文 or the English button at the upper right. The active language button is disabled; the other is available.

2. Check the title, cards, toolbar and headers. Already-open machine windows follow the shared language. Opening a machine may restore its saved language; check Menu > Language after opening.

3. Check the dynamic BLE text. Figure 1 shows BLE has stopped in the isolated offline example. When enabled, its count follows actual BLE connections, independently of the registered-device count.

4. Switch back and check the corresponding Chinese status. The selection is saved in local UI settings and reused after restart.

The language buttons retain their own names so users can switch back from an unfamiliar language. User-authored machine names, Marks, comments, function names and dictionary text keep their original content; the language switch does not translate them.

3.3 Status words and action buttons

English status	Meaning
Idle	Online with no currently running task
Running	Executing a task or motion
Paused	Paused with a recoverable execution context
Stopped	Task ended following normal stop
Offline	No online channel; live counters show a dash
Emergency stopped	Software emergency-stop latch; use the prescribed reset
Fault	Inspect error codes and channel status
Online	Communication is online; check the heartbeat for task state
Pending approval	An authorized user must verify and approve the machine
Approved / connecting	Registration is approved; connection and identification are still completing
Rejected	The connection or registration has been rejected
Waiting for MCU ID	MCU identity has not been obtained; a bridge address alone does not identify the machine
Waiting for Firmware	Bootloader confirmed; select Write Firmware to recover
Writing Firmware	Firmware transfer in progress; duplicate writing is disabled

The Status column describes the machine. Run, Pause and Stop are commands that will be sent when clicked. Idle means the machine is online with no task currently running.

A status can include a numeric suffix, such as Idle-1 or Offline-2. It comes from the locally stored device-authorization record: 0 means locked, 1 means unlocked, and 2 means no record was obtained or the record could not be read. The word describes operation or connection state. The suffix is not an online count, task number or output count, and does not replace current permissions or MCU responses.

3.4 Filters sorting counters and RSSI

At startup, registered devices can appear before discovery and MCU identification finish. Figure 1 is an isolated example with no connected MCU and zero counts. Registered and online counts describe different things; an initial online count of zero alone does not establish a fault. Peripheral means a Bluetooth peripheral device.

Connect all applies the broad list filter. Approved only shows registered approved machines. Only Device ID shows rows with an identified MCU. Registered offline machines remain manageable. BLE peripherals without an MCU identity are candidates and may not appear in the main machine list.

Click sortable headers to change order. Current and Total are sorted numerically. Current is calculated from total count and the reset baseline. Offline machines or incompatible heartbeat data show a dash rather than zero. RSSI is measured in dBm: values closer to zero usually indicate a stronger received signal, but do not prove MCU wiring or protocol compatibility.

3.5 Common device list buttons

Use this reference to identify the same controls after changing language. Their effects do not change. Availability also depends on account permissions, the connection and machine state.

Button	Purpose
Connect & Load Program	Open the selected machine and choose its program source
Switch Channel	Select a communication channel for the machine
Rename Machine	Change the registered machine name
Linkage Programs	Open PC automation program management
Database Maintenance	Back up or restore local data
Program & Template	Browse machine programs, project templates and task templates
User Management	Manage local accounts and roles
Delete	Delete registration or a connection within the confirmation scope
Stop BLE	Stop the BLE communication service
Run	Send a run command
Pause	Pause the current task while retaining a resumable execution context
Resume	Continue a paused task when the machine state permits it
Stop	Send a normal stop command
E-Stop	Send a software emergency-stop command
Reset	Request an MCU system restart, potentially followed by configured automatic homing
Write Firmware	Open recovery and the application BIN chooser; confirm before writing
Writing Firmware…	Disabled during transfer to prevent a duplicate action

Stop BLE interrupts communication and must not be used as a machine stop. Use the current run-state controls to resume a task or release an emergency-stop latch. The device Reset action restarts the MCU; restoring factory editor content instead uses the explicit Empty template workflow in Section 12.5.

4 Connection channels and machine management

4.1 Connecting a machine for the first time

1. Power the MCU and bridge according to the wiring plan. BLE requires a supported Windows adapter; COM requires the correct driver and wiring; ESP requires correct WiFi and Host connection settings.

2. Wait for the Host to identify the MCU UID. A MAC, COM name or TCP endpoint alone does not establish machine identity.

3. Select a pending MCU UID and click Add (Approve). Check its identity and name, then confirm. An unknown MCU stays pending until this manual action.

4. Once the machine is available, click Connect & Load Program. Confirm the machine name and choose its MCU program, database program or a template before editing.

BLE starts by default and local COM ports are also scanned. Stop BLE controls BLE Central only; it does not command the machine to stop. Use the row or machine-window Stop command for that purpose. Click BLE status text to inspect candidates that do not yet have an MCU ID.

4.2 Selecting and changing channels

The same MCU may have direct physical, Wi-Fi and BLE channels, combined by MCU identity. MAC identifies a replaceable BLE module. When Connect & Load Program requires channel selection, check its type, endpoint and MCU UID.

1. Select the machine. Save and close any open machine HOST window to release its channel.

2. Click Switch Channel and choose the available automatic recommendation, Wifi, BLE or direct physical connection. Unavailable choices are disabled or produce a reason.

3. Click Connect & Load Program again, confirm the machine name and load source, and check the channel and advancing Up connection time.

4. After interruption, resolve and confirm the channel and write result before retrying a download through another channel.

COM numbers depend on wiring and the PC. UART registration uses MCU logical roles; do not permanently describe one UART number as the LCD or ESP port. Changing a channel does not change the MCU UID or create an independent task set for the same MCU.

4.3 Machine names and BLE module replacement

1. Select a registered machine with an MCU UID and click Rename Machine.

2. Enter a non-empty name of at most 100 characters in the name dialog.

3. Check the old and new names in the confirmation, save and verify the updated row.

4. Names are persisted against stable machine identity, including offline machines. Renaming does not change MachineId or automation references.

Approve each MCU UID once. Its machine record persists across BLE, Wi-Fi or COM reconnection and module changes. An old MAC cannot approve an unknown UID; verify and approve that UID manually. New FFF0 modules are probed immediately at startup. A BLE link bound before Host opened actively reconnects after releasing its old session, without awaiting another advertisement. Probes time out after 10 seconds, with at most two failure retries; later broadcasts can retry. The same approved UID needs no new approval. The grid is read-only; use Rename Machine. Double-click an online approved machine to choose Connect and load. An existing window for the same MCU is activated with its current edits; pending rows open identity details.

4.4 Deleting registration and restoring defaults

Delete manages a registered machine or unidentified connection. Registered offline machines can be deleted. This is separate from erasing MCU programs. Read the stated confirmation scope; reconnection may require approval again.

To restore factory values in the editor, explicitly load the built-in Empty template as described in Section 12.5. Review tasks, points and settings, then choose database-only saving or saving with MCU download. Loading alone neither restarts the MCU nor immediately changes its program or the machine database record.

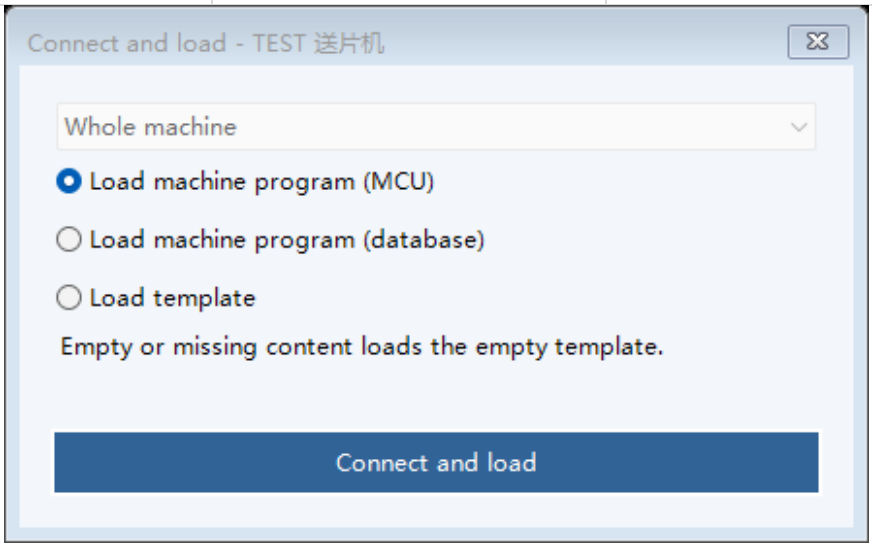
5 Machine window and configuration mode

5.1 Opening a machine HOST window

Select an approved online machine and click Connect & Load Program, or double-click its row. Check the machine name and choose the MCU program, database program or a template. An existing Host window for the

same MCU is activated with its current edits. Use Program & Template in the device list to browse stored machine programs and templates.

Source	Loaded content	What to expect
Load machine program (MCU)	The identified device's current program	Read with progress; no database program overwrite
Load machine program (database)	Machine program linked by MCU UID	Channel changes do not change ownership
Load template	A selected reusable template	Confirm first, then choose from the template list



Connect and load example with the machine name in the title.

The machine window has four main tabs: Task, Vision Coordinates, System Monitor and System Settings. It fits the current display working area without covering the taskbar. Controls reflow as the window changes; smaller windows allow scrolling to the lower System Monitor motor and I/O controls. Points are within Task, and the display-name dictionary is within System Settings. Menu at the left of the bottom status bar opens task editing and manual controls. Menu is 50 percent wider than its natural label width and remains correctly sized after language changes. The bar uses short mode, channel, connection-time and result labels; hover for the full explanation.

Up hh:mm:ss is the elapsed connection time for this machine HOST window. It starts at 00:00:00 on the first valid heartbeat and advances as new heartbeats arrive. It is not the MCU uptime or task execution time. Changing language preserves the counter; closing and reopening the window starts a new counter. The tooltip retains the full Duration wording.

Before a valid heartbeat, after disconnection or after the configured heartbeat timeout, the display is Up -------. The default timeout is 3 seconds. UI refresh alone cannot advance connection time; a recovered heartbeat starts a new session. Firmware recovery and Bootloader mode also display dashes.

The status bar keeps compact outcomes. Database saved means only the database was saved; Data saved follows successful saving and MCU download. The small progress window shows the current database, transfer and verification action. Read its error details if any stage fails; a successful earlier stage does not establish complete success.

Display	Meaning	Next action
MCU: RUN / CFG / OFF	Run / configuration / offline	Hover for mode and owner

Display	Meaning	Next action
BLE / WiFi / COM3/U4	Current channel	Confirm the target connection
Up 00:01:23	Connection time for this window	Dashes mean unavailable
Reset	Blue counter-reset button	Clears counters, not the program
Database saved	Database-only saving completed	MCU may still contain older content
Data saved	Saving and MCU transfer completed	Proceed with required device checks
Database saved; MCU download failed	Database committed, device write incomplete	Read the error and check device contents
Check incomplete	Comparison could not finish	Resolve database/communication/data errors
T0 RUN / PAUSE / STOP	Task 0 running / paused / stopped	Hover for the complete task name
FW written; version unconfirmed	Firmware write passed; runtime not confirmed	Reconnect and verify the MCU version

5.2 Operation mode and configuration mode

Operation mode allows normal operation, manual commands and queries. MCU parameter and official-storage writes require configuration ownership. The Host stops the machine and requests configuration mode through the software workflow. The MCU checks idle state and source authority before granting ownership. Heartbeat, queries, stop and emergency stop remain available, while ordinary motion, resume and homing are restricted.

Use Menu > Enter Configuration and Debug Mode or Menu > Exit Configuration and Debug Mode to request or release configuration ownership. Check MCU: RUN, MCU: CFG or MCU: OFF first. The owner port can appear as /U1 to /U5, for example MCU: CFG/U4. Editing controls can be disabled without permission or ownership; opening System Configuration and Debug does not itself grant ownership.

1. Confirm a safe mechanical state and inspect the operation/configuration indication.

2. For a write, follow the prompts to stop and obtain Host configuration ownership. Resolve busy, denied or occupied ownership instead of repeatedly clicking Download.

3. Check readback and completion messages after writing. Exit configuration mode owned by this Host, then verify operation conditions.

4. If data was written but exiting configuration failed, preserve the result and fix ownership or connectivity. Do not assume all writes failed or start the machine immediately.

The MCU arbitrates TouchLCD and physical buttons. Configuration is primarily managed by the Host; merely connecting an LCD does not grant ownership. Ordinary actions suppressed during configuration are not replayed afterwards. Diagnose BUSY, DENY and queue FULL using mode and long-command state.

5.3 Loaded data and saving on exit

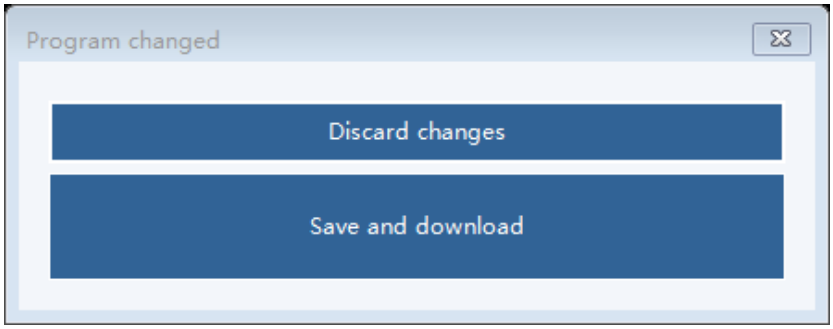
HOST loads the operation catalog and the selected program source. Expanding an operation category, selecting a Task, reopening its editor or reconnecting the same workspace displays cached content. It does not reread all tasks or clear unsaved edits. To refresh explicitly, use Program > Load program, choose the scope and MCU or database source, then confirm. A failed initial load returns to the same source choices.

An unchanged program closes without a modification prompt. After edits, Program changed offers Discard changes or Save / Download. Escape or closing the prompt returns to editing. Discard changes neither saves the

database nor downloads a program. The save dialog offers database-only saving or database saving with MCU download. Closing saves the whole machine; failure or cancellation keeps the Host open.

Save as template is the explicit reusable-template action. Saving a template in an online machine window does not save that machine program or clear its unsaved-change state. Loading MCU/database content establishes the editor baseline; a task-only save leaves unsaved edits in other tasks pending. Language changes and ordinary UI refreshes do not themselves count as program edits.

Database work, MCU transfer/readback, editor updates and any required release of configuration ownership remain inside progress with the current action displayed. Progress closes after all steps finish. Releasing configuration ownership on exit is communication cleanup, not a program download. Exit after disconnection or firmware reconnection has a bounded wait; failure restores the main window and a handling prompt rather than reporting unfinished work as successful.



Changed-program close choices; closing this prompt returns to editing.

5.4 Database and MCU consistency check

Click Program > Consistency check while connected. It reads the database machine program and MCU content and compares components, task rows and parameters. It does not compare unsaved Host edits, write either program, stop tasks or change device mode. The current editor and its selection remain intact.

Identical content produces only the message Database and MCU programs are identical. The differences window opens only for mismatches and lists component, row/field, database value and MCU value. A missing database program, communication failure or invalid content gives Check incomplete. Equivalent numeric spellings and bilingual task-caption differences do not cause false mismatches; Host-only vision data is excluded. Database/MCU reads, comparison and result preparation finish before the final message appears.

This is an explicit read action, not a heartbeat refresh. After finding differences, decide which copy to retain before loading or saving. Database-only saving can intentionally leave the database and MCU different.

6 Task editing and program data

6.1 Task slots and editor layout

There are ten ordinary slots, Slot 0 to 9, plus the instruction-test Slot 10. Each task supports functions F0 to F15. The selector shows names and updated execution states. Editing affects the selected slot, not every task or machine. Slot 10 serves instruction testing and function debugging and must not be treated as unconditionally unused storage. Its state meanings are the same as ordinary tasks, as shown below.

6.1.1 Instruction test execution states

Task 10, Instruction Test, uses the same state text and colors as ordinary tasks; it no longer shows a permanent Special label. Selecting or deselecting it does not change its execution state. Instruction Mismatch remains the higher-priority warning. The selector name still identifies the test slot.

State	Meaning	Next action
Running	Task is executing	Pause or stop as permitted
Paused	Execution context is retained	Resume when conditions permit, or stop
Stopped	Stopped or awaiting a new start after reset	Start again when the machine is ready
END Completed	Task reached its normal end	Check the result; start again if required
Error	Task execution failed	Inspect the error and log
N/A	No executable program in this task	Load or edit a task first

Open Menu > System Configuration and Debug. Task editing/debugging and motion controls now share one narrow floating window, with manual movement below task actions. The two-column window has five groups: Program, Current Task, All Tasks, Point matrix and Manual move. It uses the machine data already loaded; opening the tool does not force another MCU read.

Choose an operation in Operations, fill its parameters and use Append or Insert. Selecting an existing row shows that operation name in the parameter title. Summaries use dictionary names with their identifiers, for example Variable1 Feed flag == 2, or Y12 Feed valve = Closed. Comparison symbols remain; redundant Condition/Alarm prefixes and separator punctuation are omitted. Length is image-storage size, not execution time.

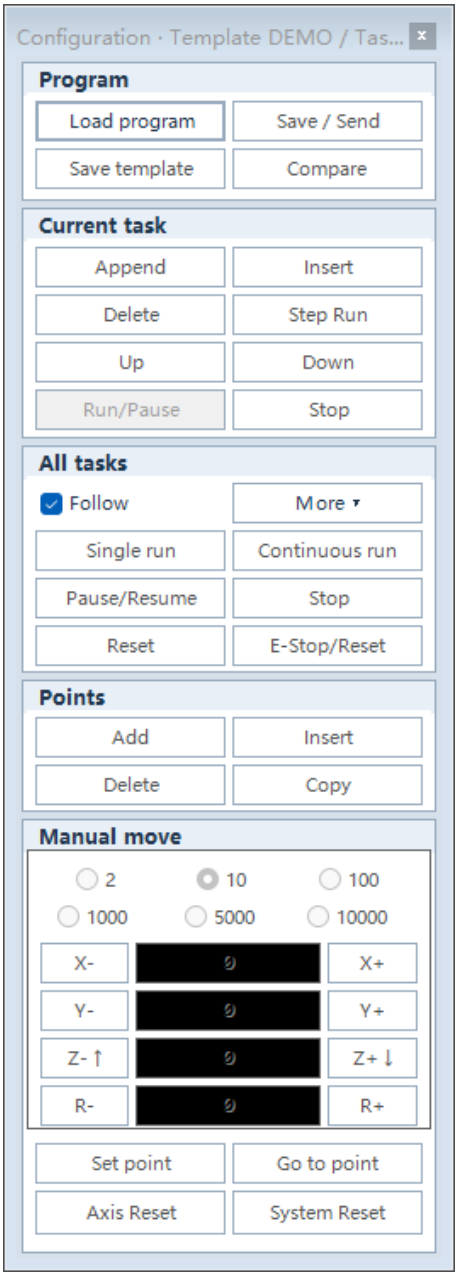
Program provides Load program, Save / Send, Save template and Compare. Current Task provides Append/Insert, Delete/Step, Up/Down and Run/Pause/Stop. All Tasks places Follow on the left and More on the right, followed by Single run/Continuous run, Pause/Resume/Stop and Reset/E-Stop/Reset. More retains System Initialize, E-Stop / System Initialize and View database program. Initialization is no longer a configurable physical-button action. Point matrix retains Add, Insert, Delete and Copy.

Entry	Scope	Action
Load program	Whole machine / Current task / Point matrix	Choose MCU, database or template source
Save / Send	Chosen scope	Database only, or database plus MCU
Save template	Explicit template scope	Save reusable content separately
Compare	Database and MCU	Read-only content comparison
More	All Tasks	Initialize, E-Stop / Initialize, View database program
Reset (status bar)	Counters	Clear counters; distinct from system reset

The tool opens near the upper-right corner of the current screen. Its title identifies the machine/template and current task. Two-column buttons keep readable text spacing; height is limited to the screen working area, with scrolling when a smaller display needs it. Closing the tool hides it; reopening reuses the window and coordinate display.

Manual move offers 2, 10, 100, 1000, 5000 and 10000-pulse steps, signed X/Y/Z/R buttons and a central coordinate display. Z- ↑ indicates up and Z+ ↓ down; confirm physical directions and wiring. Set point, Go to point, Axis Reset and System Reset are below. The merged tool preserves motion commands, permission checks and configuration-mode restrictions; unavailable offline or unowned actions remain disabled. Host homing is blocked while any task from 0 through 10 is running or paused. Stop the tasks and confirm their states first.

On the first opening, an editor with no valid selection selects its first operation. Each Task remembers its own selected row for this Host session. Switching back, reopening or reloading preserves that row and its parameters. A missing remembered row falls back to the first available row; an empty task shows Operation Parameters with no stale values. Function editing keeps a separate selection and restores the parent Task selection when closed. Changing language translates the title without moving the selection.



Offline layout example. The image retains the former Configuration title; the current window is System Configuration and Debug. Manual move is below task controls; availability depends on connection, mode and permissions.

6.2 Creating and changing a task

1. Select a task and check its existing rows. When replacing data, use Load program with Current task and the required MCU, database or template source.

2. Choose an instruction in the tree. Inspect parameter hints; incompatible entries may appear red or cannot be added.

3. Enter and submit real parameters. To modify an existing operation, select its row first rather than changing only the pending default operation.

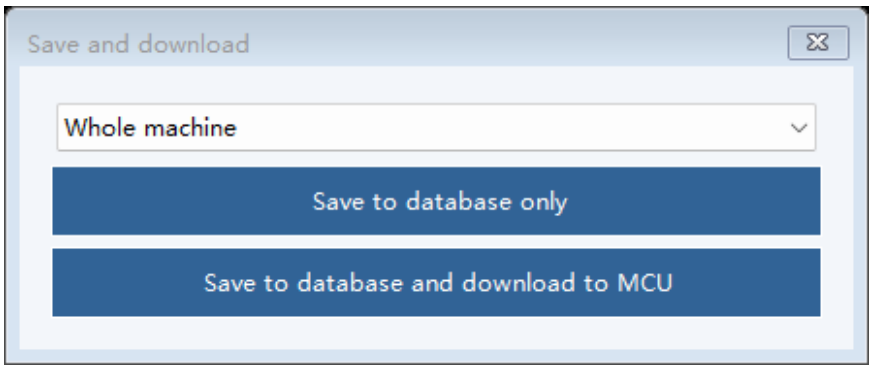
4. Append to the end or Insert at a selected position. Use move up/down to arrange execution order.

5. Write Comments and legal target Marks. Update jump references before deleting a referenced Mark row.

6. Check summaries, Marks, functions and image size. Use Save / Download with Current task; choose database-only saving to retain edits before download.

6.3 Saving machine programs and downloading

Click Program > Save / Download and choose Whole machine, Current task or Point matrix. Save to database only stores that scope under the current MCU UID without writing to the MCU. Save to database and download to MCU first commits the database data, then writes and verifies the MCU. It never saves a template. An offline-template window instead uses this entry to select a target for project download; see Section 12.3.



Choose the save scope and whether to write the MCU.

1. Confirm the machine name, MCU UID and scope. Use Whole machine to save all task, dictionary, point and machine-setting edits together. A task-only save preserves other tasks and merges only that task's bilingual names; a point-only save preserves task data.

2. For MCU download, check the channel and safe stopped state. Parameter, target Mark, hardware, instruction support and image-size checks run before transfer. Resolve reported errors before proceeding.

3. Choose one of the two save actions. The database transaction must succeed before the first MCU write. A newer database revision cannot be overwritten by a stale editor; reload and reconcile the changes if a revision conflict is reported.

4. Follow the progress window through database saving, MCU writing, text/dictionary work and readback. Database saved; MCU download failed means the database copy was saved but the device still requires checking. Do not treat that partial result as Data saved.

5. After completion, use Consistency check or an explicit Load program from MCU to confirm device contents. Exit configuration mode and follow the approved low-speed or single-step commissioning procedure.

Where firmware supports it, the Host first compares verified MCU storage fingerprints to avoid repeated transfers. Unchanged program, point, remark and dictionary images skip their matching writes; changed images are transferred. Older firmware retains the readback-verification path. Skipping redundant transfers does not omit database saving or verification; completion still waits for all required work.

Each task program image, including functions, is limited to 2048 bytes; the point-table image is limited to 4096 bytes. MCU display remarks are limited to eight displayed characters and 32 UTF-8 bytes. When download detects overlong remarks, choose Yes to truncate and continue or No to return to editing; check that the remaining text is meaningful. Program transfer, project-text transfer and dictionary transfer have separate results: preserve a successful program write and investigate a later text/dictionary failure rather than assuming that all data failed.

6.4 Functions and standalone tests

Use the function-node menu to edit, rename or clear a function. Function operations use the normal validation rules but cannot include END or another CALL_FUNCTION; nested calls are unsupported. The function returns when its body ends. Marks have separate scope and timers belong to the caller task. Standalone function debugging uses Slot 10 and its backup/restore workflow; read the prompts first.

Instruction testing provides download, read, run, resume, pause, stop and single-step for small programs. Stop a running or paused test as required before editing. Homing-repeatability results appear in a separate Test Results window rather than occupying the status bar. Elapsed (Host) is measured from sending the test command until the Host receives the MCU result; it includes communication time and is not an MCU motion-only duration.

7 Points and coordinates

7.1 Points and point arrays

Open the point area in the Task main tab for label, index and X/Y/Z/R. Point actions are in Menu > System Configuration and Debug > Point matrix; capture and positioning are in Manual move below. The adjacent array fields provide start/end XY and point count. Tasks reference point indexes, so review references before deleting or reordering. Labels do not replace indexes. Coordinates are pulses.

1. Perform required homing and check position and soft limits.

2. Select a point and use System Configuration and Debug > Manual move > Set point to capture the current coordinates, or enter X/Y/Z/R manually.

3. Use add, insert, delete and copy. To generate an array, enter start/end XY and point count.

4. Check indexes, labels and every position. In Program > Save / Download choose Point matrix, then database-only saving or database saving with MCU download. Use Save as template > Point matrix for a reusable point template.

5. System Configuration and Debug > Manual move > Go to point produces real motion. Confirm operating mode, workspace and channel before using it.

7.2 Offsets and coordinate meanings

The X/Y/Z/R offset fields and Apply +/Apply - edit coordinate sets. Back up first and check the affected points and results. Relative motion adds an increment to the current position; absolute motion uses a target coordinate. Repeated relative moves accumulate and can exceed travel.

Pulse equivalence converts physical travel: linear axes use nm/pulse and R uses pulses per degree. Recheck coordinates and limits after changing equivalence, microstepping or transmission. The user defines safe_z in the machine coordinate system; the software does not infer obstacles.

8 Program instruction reference

This chapter explains the 37 operations in the current V4.23 catalog. Timer offers five actions in one catalog entry, rather than five separate instructions. Use matching Host and MCU firmware. Before download, the Host checks Conditional Wait, user error code and timer-control capabilities, including operations inside functions. Defaults illustrate parameters rather than machine-specific values. CW means clockwise and CCW counter-clockwise. Compensation IDs are 0 to 7; 4294967295 (0xFFFFFFFF) disables compensation.

Compilation and download also depend on hardware, system settings, motor operating mode, program context and MCU validation. Historical aliases retained for importing older records are not the current add-instruction catalog; for example, inspect imported pick/place records separately.

8.1 Interpolated Absolute Move to Point

Instruction code MOVE_POINT_ID

Interpolated absolute move to the selected point.

Parameter	Default	Meaning
point_index	0	Point-table index. Absolute moves use target coordinates; relative moves use increments. The referenced point must exist.
compensation_id	4294967295	MCU compensation ID 0 to 7; 4294967295 disables compensation. Verify the source and current result.

Operation check Create and check the point and compensation first; repeated execution targets the same coordinates.

8.2 Interpolated Absolute Move to Coordinates

Instruction code MOVE_XYZR

Interpolated move to explicit absolute X/Y/Z/R.

Parameter	Default	Meaning
x	0	X target or increment in pulses. Absolute targets must satisfy configured travel.
y	0	Y target or increment in pulses. Absolute targets must satisfy configured travel.
z	0	Z target or increment in pulses. Check the safe height and complete path.
r	0	R target or increment in pulses, subject to R-axis enable and equivalence.
compensation_id	4294967295	MCU compensation ID 0 to 7; 4294967295 disables compensation. Verify the source and current result.

Operation check Check travel and the complete path; confirm R pulse equivalence when enabled.

8.3 Interpolated Relative Move to Coordinates

Instruction code MOVE_RELATIVE

Interpolated X/Y/Z/R increment from current position.

Parameter	Default	Meaning
x	0	X target or increment in pulses. Absolute targets must satisfy configured travel.
y	0	Y target or increment in pulses. Absolute targets must satisfy configured travel.

Parameter	Default	Meaning
z	0	Z target or increment in pulses. Check the safe height and complete path.
r	0	R target or increment in pulses, subject to R-axis enable and equivalence.
compensation_id	4294967295	MCU compensation ID 0 to 7; 4294967295 disables compensation. Verify the source and current result.

Operation check Repeated execution accumulates movement. Limit loop counts and final positions.

8.4 Interpolated Relative Move to Point

Instruction code MOVE_RELATIVE_POINT

Use point-table components as relative increments.

Parameter	Default	Meaning
point_index	0	Point-table index. Absolute moves use target coordinates; relative moves use increments. The referenced point must exist.
compensation_id	4294967295	MCU compensation ID 0 to 7; 4294967295 disables compensation. Verify the source and current result.

Operation check The point is not an absolute destination. Check signs and repetition count.

8.5 Safe Absolute Move to Point

Instruction code SAFE_MOVE_POINT

Safe-move sequence to an absolute point with safe_z.

Parameter	Default	Meaning
point_index	0	Point-table index. Absolute moves use target coordinates; relative moves use increments. The referenced point must exist.
safe_z	0	Safe Z coordinate in pulses used by the selected safe-move sequence.
compensation_id	4294967295	MCU compensation ID 0 to 7; 4294967295 disables compensation. Verify the source and current result.

Operation check Check Z direction and clearance. The name does not guarantee collision avoidance.

8.6 Safe Absolute Move to Coordinates

Instruction code SAFE_MOVE_XYZR

Safe-move sequence to explicit absolute coordinates.

Parameter	Default	Meaning
x	0	X target or increment in pulses. Absolute targets must satisfy configured travel.

Parameter	Default	Meaning
y	0	Y target or increment in pulses. Absolute targets must satisfy configured travel.
z	0	Z target or increment in pulses. Check the safe height and complete path.
r	0	R target or increment in pulses, subject to R-axis enable and equivalence.
safe_z	0	Safe Z coordinate in pulses used by the selected safe-move sequence.
compensation_id	4294967295	MCU compensation ID 0 to 7; 4294967295 disables compensation. Verify the source and current result.

Operation check Verify target coordinates, safe_z and the machine origin direction.

8.7 Safe Relative Move to Point

Instruction code SAFE_MOVE_RELATIVE_POINT

Safe relative move using point-table increments and safe_z.

Parameter	Default	Meaning
point_index	0	Point-table index. Absolute moves use target coordinates; relative moves use increments. The referenced point must exist.
safe_z	0	Safe Z coordinate in pulses used by the selected safe-move sequence.
compensation_id	4294967295	MCU compensation ID 0 to 7; 4294967295 disables compensation. Verify the source and current result.

Operation check Calculate the target from current position; checking the stored point alone is insufficient.

8.8 Safe Relative Move to Coordinates

Instruction code SAFE_MOVE_RELATIVE

Safe relative move using explicit increments and safe_z.

Parameter	Default	Meaning
x	0	X target or increment in pulses. Absolute targets must satisfy configured travel.
y	0	Y target or increment in pulses. Absolute targets must satisfy configured travel.
z	0	Z target or increment in pulses. Check the safe height and complete path.
r	0	R target or increment in pulses, subject to R-axis enable and equivalence.
safe_z	0	Safe Z coordinate in pulses used by the selected safe-move sequence.
compensation_id	4294967295	MCU compensation ID 0 to 7; 4294967295 disables compensation. Verify the source and current result.

Operation check Check the calculated target, avoidance sequence and accumulated loop positions.

8.9 Single-Axis Absolute Move to Point

Instruction code MOVE_AXIS_ABSOLUTE_POINT

Absolute single-axis move using that component of a point.

Parameter	Default	Meaning
axis	X	Selected X, Y, Z or R axis. The operating mode must support coordinate axes.
point_index	0	Point-table index. Absolute moves use target coordinates; relative moves use increments. The referenced point must exist.
compensation_id	4294967295	MCU compensation ID 0 to 7; 4294967295 disables compensation. Verify the source and current result.

Operation check Other components are not move targets; verify the single-axis path.

8.10 Single-Axis Absolute Move to Coordinate

Instruction code MOVE_AXIS_ABSOLUTE

Move one coordinate axis to an absolute pulse position.

Parameter	Default	Meaning
axis	X	Selected X, Y, Z or R axis. The operating mode must support coordinate axes.
pulse_count	0	Target/increment for coordinate moves, or fixed-pulse quantity in RUN_MOTOR.
compensation_id	4294967295	MCU compensation ID 0 to 7; 4294967295 disables compensation. Verify the source and current result.

Operation check The value is a machine-coordinate target, not a travel distance.

8.11 Single-Axis Relative Move to Point

Instruction code MOVE_AXIS_RELATIVE_POINT

Single-axis relative move using a point component.

Parameter	Default	Meaning
axis	X	Selected X, Y, Z or R axis. The operating mode must support coordinate axes.
point_index	0	Point-table index. Absolute moves use target coordinates; relative moves use increments. The referenced point must exist.
compensation_id	4294967295	MCU compensation ID 0 to 7; 4294967295 disables compensation. Verify the source and current result.

Operation check Add the stored increment to current position to check the resulting target.

8.12 Single-Axis Relative Move to Coordinate

Instruction code MOVE_AXIS_RELATIVE

Move one coordinate axis by a pulse increment.

Parameter	Default	Meaning
axis	X	Selected X, Y, Z or R axis. The operating mode must support coordinate axes.
pulse_count	0	Target/increment for coordinate moves, or fixed-pulse quantity in RUN_MOTOR.
compensation_id	4294967295	MCU compensation ID 0 to 7; 4294967295 disables compensation. Verify the source and current result.

Operation check Validate signs, current position and final travel.

8.13 Request Vision Compensation

Instruction code REQUEST_ROI_OFFSET

Request Host processing of an ROI and update its compensation.

Parameter	Default	Meaning
roi_id	0	Vision ROI or compensation ID 0 to 7; require valid configuration and results.

Operation check Require the vision service, camera and valid ROI. Missing results are not successful zero compensation.

8.14 Relative Compensation Move

Instruction code OFFSET_COMPENSATION

Move the selected axis by the chosen ROI compensation.

Parameter	Default	Meaning
axis	X	Selected X, Y, Z or R axis. The operating mode must support coordinate axes.
roi_id	0	Vision ROI or compensation ID 0 to 7; require valid configuration and results.

Operation check Obtain a valid result first and check direction and pulse units.

8.15 Arc XY

Instruction code ARC_XY

XY arc defined by end position, Z, center offsets and direction.

Parameter	Default	Meaning
x	0	X target or increment in pulses. Absolute targets must satisfy configured travel.

Parameter	Default	Meaning
y	0	Y target or increment in pulses. Absolute targets must satisfy configured travel.
z	0	Z target or increment in pulses. Check the safe height and complete path.
i	0	X center offset relative to the current position, in pulses.
j	1000	Y center offset relative to the current position, in pulses.
direction	CW	CW or CCW. Check the complete path and travel independently.

Operation check i/j are relative center offsets, not absolute center coordinates. Check radii and full travel.

8.16 Three-Point Arc XY

Instruction code ARC_THREE_POINT_XY

XY arc through explicit start, intermediate and end points.

Parameter	Default	Meaning
start_x	1000	Explicit three-point arc start X in pulses; must match MCU current position.
start_y	0	Explicit three-point arc start Y in pulses; position there before execution.
via_x	2000	Arc intermediate X in pulses; avoid coincident/degenerate or collinear points.
via_y	1000	Arc intermediate Y in pulses.
end_x	1000	Arc end X in pulses.
end_y	2000	Arc end Y in pulses.

Operation check Avoid collinearity and invalid integer center/radius geometry. Host checks travel; MCU also checks the current start.

8.17 Center-Radius Circle XY

Instruction code CIRCLE_XY

Complete XY circle from absolute center and radius.

Parameter	Default	Meaning
center_x	2000	Absolute circle-center X in pulses.
center_y	2000	Absolute circle-center Y in pulses.
radius	1000	Circle radius in pulses, greater than zero.
direction	CW	CW or CCW. Check the complete path and travel independently.

Operation check Current position must be on the circle; its complete boundary must fit XY travel.

8.18 Ellipse XY

Instruction code ELLIPSE_XY

XY ellipse from relative center, semi-axes, rotation and direction.

Parameter	Default	Meaning
z	0	Z target or increment in pulses. Check the safe height and complete path.
i	-1000	X center offset relative to the current position, in pulses.
j	0	Y center offset relative to the current position, in pulses.
major_radius	1000	Ellipse major semi-axis in pulses, at most 1000000 and not smaller than the minor semi-axis; the axis ratio cannot exceed 10:1.
minor_radius	500	Ellipse minor semi-axis in pulses, at least 10; this is a semi-axis length, not a full diameter.
rotation_angle_x1000	0	Ellipse rotation in degrees multiplied by 1000, from -360000 to 360000; 90000 means 90 degrees.
direction	CW	CW or CCW. Check the complete path and travel independently.

Operation check Use semi-axis lengths and angle multiplied by 1000. Check the complete rotated path and start.

8.19 Multi-Axis Home

Instruction code HOME

Home coordinate axes in the selected A to I sequence. Select single-stage or three-stage mode, then choose the homing order.

Parameter	Default	Meaning
Mode / home_mode	2	Single-stage or Three-stage. Single-stage uses each axis search rate; three-stage uses search, release and latch. Requires the matching ZE MCU firmware.
Order / home_order	DEFAULT	DEFAULT uses the saved machine order; A to I select an explicit sequence. The selected mode applies to every axis in the sequence.

Legacy templates and MCU home instructions are converted to Mode and Order. Legacy mode 0 becomes Three-stage; a missing order uses the machine default. Review each axis's homing rates and the order before downloading the converted program.

Operation check Z completes the selected homing cycle before the next group. In (XY) and (XYR), each axis independently runs the selected cycle. A completed axis stops its own pulses; the group waits until every participating axis has finished. A failed axis stops the group and prevents later groups.

If X starts on its home switch, X first releases that switch while Y and enabled R begin searching. Confirming a home switch or its release requires a continuous stable signal for at least 5 ms; a rebound restarts confirmation.

Stop ends the current homing cycle. A new start is rejected while that cycle is still exiting; wait for it to finish and retry. Restart begins a new cycle.

8.20 Single-Axis Home

Instruction code RESET_AXIS

Home one selected X, Y, Z or R axis using single-stage or three-stage mode. Speed selection is separate from this motion command.

Parameter	Default	Meaning
Axis	X	Selected X, Y, Z or R axis. The operating mode must support coordinate axes.
Mode / home_mode	1	1 searches only; 2 searches, releases and latches. Both use the selected axis profile.

Operation check Mode 1 uses only this axis search rate and completes without motion on a stable home switch. Mode 2 releases an initially active switch, then runs all three stages. Unified Reset speed is removed. Legacy RESET and HOME programs are converted on import/readback and must be recompiled and downloaded.

8.21 System Initialize

Instruction code SYSTEM_INITIALIZE

Stop tasks and restore startup state for the current operating mode.

No user numeric parameters.

Operation check Coordinate mode includes multi-axis homing; independent mode stops motors and resets corresponding I/O, variables and state. Motion may occur.

8.22 Motor Homing Repeatability Test

Instruction code TEST_PRECISION

Home and repeatedly travel the chosen axis/motor to measure repeatability.

Parameter	Default	Meaning
axis	X	Selected X, Y, Z or R axis. The operating mode must support coordinate axes.
test_pulse_count	10000	Travel pulse quantity for repeated homing-position tests.
test_count	10	Number of repeated tests, from 1 to 100; this causes repeated real motion.

Operation check Choose XYZR in coordinate mode or an available independent motor. This is motion, not a static query.

8.23 Set Speed/Accel

Instruction code SET_SPEED_HZ

Change runtime linear/arc speed and acceleration; zero keeps the corresponding current value.

Parameter	Default	Meaning
speed_hz	0	Pulse frequency in Hz; zero semantics depend on the instruction.
accel_steps_s2	0	Linear interpolation acceleration in pulse/s².

Parameter	Default	Meaning
arc_speed_hz	0	Arc interpolation frequency in Hz.
arc_accel_steps_s2	0	Arc interpolation acceleration in pulse/s².

Operation check Use Hz and pulse/s² correctly. This does not persist system settings to Flash.

8.24 Set Axis Homing Stage Speeds

Instruction code SET_HOME_STAGE_SPEEDS

Set the selected axis runtime search, release and latch speeds without moving it. Every task starts with no overrides; called functions use the caller task settings.

Parameter	Default	Meaning
axis	X	Choose X, Y, Z or R; rates do not affect other axes.
stage1_rate_hz	0	Search speed; zero selects the saved default.
stage2_rate_hz	0	Release speed; zero selects the saved default.
stage3_rate_hz	0	Latch speed; zero selects the saved default.

Operation check Each value is 0 to 64000 Hz. New instructions use zero to clear that stage override and select its saved per-axis default. Overrides apply only within the current task and do not write Flash. Unmarked old templates and old binary instructions remain legacy: their zero still means unchanged.

8.25 Set Motor Speed

Instruction code SET_MOTOR_SPEED

Set an independent motor frequency.

Parameter	Default	Meaning
motor_index	4	Independent motor index; coordinate mode restricts this set to M4 through M7.
speed_hz	4000	Pulse frequency in Hz; zero semantics depend on the instruction.

Operation check The RUN_MOTOR system has a 4000 Hz maximum. Select only motors allowed by the mode.

8.26 Motor Run

Instruction code RUN_MOTOR

Start an independent motor continuously or for a fixed pulse count.

Parameter	Default	Meaning
motor_index	4	Independent motor index; coordinate mode restricts this set to M4 through M7.
run_mode	continuous	continuous or fixed_pulses mode.

Parameter	Default	Meaning
pulse_count	0	Target/increment for coordinate moves, or fixed-pulse quantity in RUN_MOTOR.
acceleration_steps_s2	32000	Independent motor start acceleration in pulse/s².

Operation check Design a STOP_MOTOR/sensor termination for continuous operation. Fixed mode requires pulse count and acceleration.

8.27 Motor Stop

Instruction code STOP_MOTOR

Stop the selected independent motor with specified deceleration.

Parameter	Default	Meaning
motor_index	4	Independent motor index; coordinate mode restricts this set to M4 through M7.
deceleration_steps_s2	32000	Selected motor stop deceleration in pulse/s².

Operation check This is a single-motor command, distinct from system Stop and Emergency stop.

8.28 Run Indexer

Instruction code RUN_INDEXER

Drive an indexer object and use a sensor input for positioning.

Parameter	Default	Meaning
target	M0	Indexer drive object; choose a permitted motor or Y output for the operating mode.
input_index	X0	Condition operand or indexer sensor input. The indexer uses physical X inputs.

Operation check Verify that the sensor reaches the expected state. Legal drive objects depend on motor operating mode.

8.29 Output IO

Instruction code IO_WRITE

Set the selected Y open/closed and delay before advancing.

Parameter	Default	Meaning
io_index	0	Y output index. Coordinate-mode programs reserve Y0 through Y3 for direction.
io_value	0	Choose the displayed open/closed state; do not infer wiring polarity from 0 or 1.
delay_ms	0	Delay in milliseconds; DELAY_MS blocks the current task.

Operation check Check actual wiring. Coordinate-mode Y0 to Y3 are reserved direction outputs.

8.30 Delay

Instruction code DELAY_MS

Keep the current task at this row until the delay expires.

Parameter	Default	Meaning
delay_ms	100	Delay in milliseconds; DELAY_MS blocks the current task.

Operation check Other tasks can run. It neither creates nor starts T0 to T3.

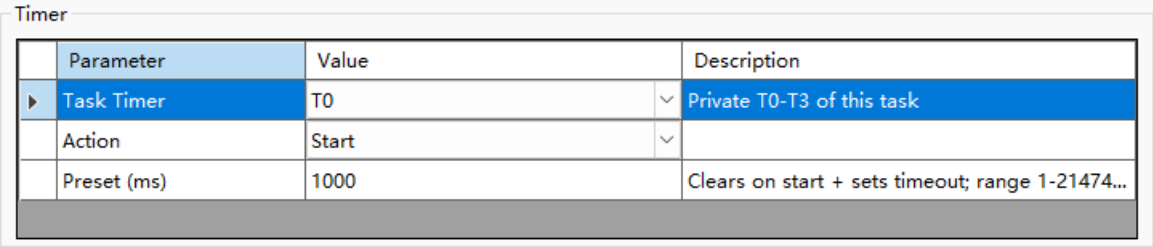
8.31 Timer

Instruction type key TIMER_START Display name Timer

Choose T0 to T3 and an Action: Start, Pause, Resume, Stop or Reset. Each action immediately advances to the next row without blocking the task. Only Start uses a preset in milliseconds.

Parameter	Default	Meaning
timer_id	T0	Current-task private timer T0 to T3.
action	START	Start / Pause / Resume / Stop / Reset
preset_ms	1000	Start only: 1 to 2147483647 ms; hidden for other actions.

Operation check Start counts from zero and clears the previous expiration state. Pause freezes the value; Resume continues only a paused timer and excludes paused time. Stop prevents Resume until a new Start. Reset clears and stops while retaining the last preset. Expiration does not automatically jump, fault or stop a task. See Section 9.2 for the action matrix.



Timer parameters with Action Start; presets are hidden for other actions.

8.32 Set Variable

Instruction code SET_VAR

Assign, add to or subtract from V0 to V18; system_count adds system counts.

Parameter	Default	Meaning
variable_index	0	Tasks can write V0 to V18 or select system_count to add system counts. V19 is task-read-only motion mode and cannot be written by SET_VAR.
assign_mode	=	For ordinary variables, = assigns, + adds and - subtracts. system_count always adds; it does not assign or subtract.
value	0	Numeric assignment/comparison value within the current validation range.

Operation check Tasks cannot write V19. Define ownership/order for shared variables; system_count is not an ordinary variable.

8.33 Jump

Instruction code JUMP

Unconditional jump to a Mark in the current scope.

Parameter	Default	Meaning
target_mark		Target Mark within this scope. Conditional jumps can also target the current row.

Operation check The Host resolves its location during download; cross-function/task targets are invalid.

8.34 Conditional Jump

Instruction code JUMP_IF_INPUT

Evaluate once: true jumps, false advances.

Parameter	Default	Meaning
input_index	X0	Condition operand or indexer sensor input. The indexer uses physical X inputs.
compare_source	XSTATE_CLOSE	Second operand source or manual numeric value; internal codes are shown as meaningful states.
condition	=	Equals, not equals, less/greater than, or less/greater than or equal. Internal not-equal encoding is <>.
target_mark	Current row	Target Mark within this scope. Conditional jumps can also target the current row.

Operation check A current-row target repeats while true and advances when false. Conditional Jump has no built-in timeout. Use Conditional Wait when false must wait and timeout must pause all tasks and latch an alarm output.

8.35 Conditional Wait

Instruction code WAIT_FOR_SIGNAL

Choose Variables & Flow > Conditional Wait. Its two operands and dropdowns match Conditional Jump, with no target Mark or row. Select a user error code from 128 to 255 already defined in the data dictionary, including

when timeout is zero. System codes cannot be selected; distinct machine faults use their own descriptions instead of a generic wait-timeout code.

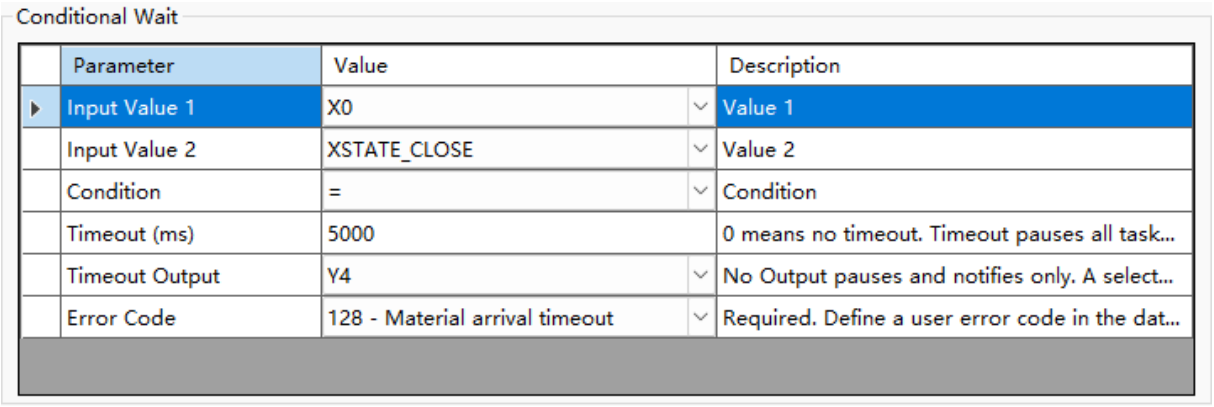
Parameter	Default	Meaning
input_index	X0	First operand: input, variable, output or current-task timer value/state, as offered by the dropdown.
compare_source	XSTATE_CLOSE	Second operand or manual numeric value. A manual value is unsigned 32-bit.
condition	=	=, <>, <, >, <= or >=. The row advances only when the comparison is true.
timeout_ms	0	0 to 2147483647 ms. Zero waits indefinitely without a timeout alarm.
timeout_output	Y4	No Output, or Y4-Y31 on 32Y / Y4-Y23 on 24Y. No Output still records the timeout and pauses tasks; it does not change a Y output.
timeout_error_code	Required	Defined user code 128-255 only; mandatory even with timeout 0.

True advances immediately. False keeps only this task at the same row until the condition becomes true or the timeout is reached; other tasks and communication can continue. If the condition is true at the timeout boundary, the satisfied condition takes precedence.

On timeout the MCU pauses all running tasks, retains the waiting row and records the selected user error code. A chosen Y output stays active for an external lamp or sounder; No Output leaves every Y unchanged. The Host shows codes with dictionary descriptions and offers Keep Paused or Continue. Enter, Escape and closing the prompt keep the pause. With no Host, the MCU still pauses and records the error; an external alarm requires a selected Y output.

Continue clears this kind of timeout error and its held output, resumes tasks affected by the pause and retries the same row with a fresh interval. It neither bypasses the comparison nor starts stopped/completed tasks. Without a Host, resolve the cause and use the configured physical Continue input. The MCU uses internal fault types to permit recovery; a user-selected number cannot change fault severity or override other blocking faults. A shared output stays active while another timeout holds it.

The Host checks firmware support before task/project writes, including waits inside functions. No Output and user error codes each need matching firmware capabilities. Update unsupported firmware first. Legacy wait records without a valid user code need an explicit selection before saving again. Conditional Wait does not consume T0 to T3. Its summary omits repeated Timeout and Alarm labels, for example Variable1 Feed flag == 0 None No Output 128 Material arrival timeout; None means unlimited waiting.



Conditional Wait example: 5000 ms, Y4 and user code 128 Material arrival timeout. Description columns are widened for print readability; use machine-specific values.

8.36 Call Function

Instruction code CALL_FUNCTION

Run a defined function of this task and return at its body end.

Parameter	Default	Meaning
function_id	0	Current-task function F0 to F15; it must exist.

Operation check Functions F0 to F15 cannot contain END or nested calls. They share caller context/timers. Check Slot 10 for separate debug.

8.37 Task End

Instruction code END

End the normal task instruction sequence.

No user numeric parameters.

Operation check Arrange output and motor cleanup explicitly; do not substitute END for a designed independent-motor stop.

9 Marks functions variables and timers

9.1 Marks and Comments

A Mark is a jump target; Comments explain intent without affecting execution. A Mark allows at most eight Unicode display characters and 32 UTF-8 bytes. Use letters, digits and underscores; the first character cannot be a digit. Chinese letters are valid. Marks must be unique within their scope, ignoring case.

The current-row name and CURRENT, CURRENT_ROW and SELF are reserved conditional-jump targets, not ordinary Marks. Renaming a Mark updates same-scope references. Referenced Marks cannot simply be cleared or deleted. Main programs and functions resolve their own scope; cross-scope jumps are invalid.

A current-row Conditional Jump repeats while its condition is true and advances when false. Conditional Wait has the opposite waiting rule: false waits and true advances. For a direct wait until X0 closes, choose Conditional Wait with X0 = Closed. See Section 8.35 for timeout and resume behavior.

9.2 Task timers

Each task has private timers T0 to T3. Tn.Value is accumulated milliseconds and Tn.Timeout indicates expiration. Same-numbered timers in different tasks are independent; functions use caller timers. In the unified Timer instruction, only Start uses a preset from 1 to 2147483647 ms. Other actions do not take a preset.

Action	Value and preset	Next step
Start	Value/Timeout cleared; new preset	Counts immediately from zero
Pause	Reading frozen	Resume continues this count
Resume	Keeps count; excludes paused time	Only resumes a paused timer
Stop	Value/Timeout frozen	Requires a new Start

Action	Value and preset	Next step
Reset	Value/Timeout zero; preset retained	Stopped until Start

Timing uses the MCU 1 ms system tick; actual instruction response also depends on task scheduling. Pausing a task does not freeze its timers. Only the timer Pause action freezes its value, and Resume excludes the frozen interval. Stopping or restarting a task clears all its timers. An unstarted timer has zero Value and Timeout; an expired value is latched at its preset.

1. Add Timer with T0, Action Start and a 1000 ms preset. Old projects without an action field retain Start behavior.

2. Add work to perform during timing. Add Pause for the same T0 to freeze it and Resume to continue it. Task pause and timer Pause are separate operations.

3. Use Conditional Jump to compare T0.Timeout with not expired and target the current row while true; expiration advances the row. Compare T0.Value when accumulated time is needed.

4. Use Start for a new timing cycle. Stop retains the reading and prevents Resume; Reset clears it and stops. Start-only records retain their previous compatibility. Stop, Pause, Resume and Reset require timer-control firmware support, which the Host checks before download.

DELAY_MS blocks only the current task at a row. Task timers provide non-blocking elapsed/expiration values. Conditional Wait waits on a comparison and optionally pauses all tasks on timeout; it does not consume T0 to T3. The Host automation TIMER delays the PC interpreter. To inspect MCU T0.Value directly, use System Monitor > Task Timers as described in Section 14.3; copying the value into V variables is unnecessary.

9.3 Functions and variables

A task supports 16 functions F0 to F15. A function does not create an independent task or separate timer set. The 20 global variables V0 to V19 are shared across tasks. Distinguish assignment/add/subtract and define write ownership in concurrent programs.

Task SET_VAR instructions can write V0 to V18. V19 holds motion mode: 0 means system startup, 1 means S single mode and 2 means C continuous mode. Tasks can read or compare V19 but cannot change it with SET_VAR. Task-read-only does not mean every debugging entry is prevented from changing the value.

Ordinary variables are unsigned 32-bit integers from 0 to 4294967295. Subtraction floors at zero; addition wraps if it exceeds the maximum, so check the boundary before adding. For example, subtracting 5 from V0=2 gives zero, not a signed result of -3.

When system_count is selected, SET_VAR adds value to the system total count. Download forces addition even if assign_mode in a draft is = or -. This does not directly assign or clear the total. Keep the resulting production total at or below 2147483647 and verify Current, Total and the reset baseline.

Keep a complete template and database backup before importing older projects. Current downloads use the supported baseline project representation. Old firmware may lack project-text readback or Conditional Wait support; use the matching firmware when prompted. Changing a display label or manually changing a template format number does not make an unsupported instruction compatible.

10 System parameters and motor operating mode

10.1 Configuration areas

Open System Settings. Its tabs are Common Parameters, Interpolation Parameters and Non-Interpolation Parameters, alongside the display-name dictionary. Common Parameters includes physical buttons and the M0-M3 mode. Read/write controls apply to the selected area. To retain a reusable copy, use Program > Save as template > Current parameter section. Read and back up before editing, then resolve validation messages before writing.

Group	Examples	Check
General	Axis maximum pulses, R enable, home, auto-home, vision	Actual travel and switch wiring
Linear interpolation	Start/max frequency, acceleration, pause deceleration	Separate Hz and pulse/s²; meet pause rules
Arc interpolation	Main-axis maximum speed and acceleration	Load and arc requirements
Pulse equivalence	Linear nm/pulse, R pulses/degree	Recheck after microstep/transmission changes
Independent motors	Motor frequencies and homing profile	M4 to M7 are the same physical motors in both modes

The Host requires integer pause deceleration greater than linear acceleration and at least the software minimum. Pulse equivalence and arc speed/acceleration also have positive-value rules. Zero defaults in runtime instructions do not imply that persistent configuration fields may be zero.

Acceleration-table capacity is checked when saving interpolation settings. The table has 4096 entries. If a speed/acceleration pair exceeds that capacity, the warning gives both the maximum speed at the entered acceleration and the minimum acceleration at the entered maximum speed, using the configured start rate. Choose No to adjust the settings or Yes to retain them. Within capacity, no warning appears. The warning does not change the firmware or guarantee motor performance; an abrupt speed change can still require mechanical and motion-profile checks.

10.1.1 XYZR homing defaults

In Interpolation Parameters, edit each XYZR row: Search Hz, Release Hz, Release pulses and Latch Hz. Release pulses is the additional outward movement after the home switch is stably released. Enter 0 to 65535 pulses per axis; 0 preserves the previous behavior. Save/download persists the three speeds, release distance and default order. Runtime speed instructions override the speeds only.

During three-stage homing, search until the switch triggers, release until the switch is stably open, move the configured additional release pulses, then latch slowly and set zero. The same release distance also applies when a three-stage cycle starts with the switch already triggered. Single-stage homing ignores this distance. Old configuration and templates default missing release distances to 0. New Host requires MCU homing profile version 2 before any system-setting write; update the matching 24Y or 32Y application firmware first. Old profile readback remains readable and reports zero additional release pulses.



Parentheses mean simultaneous movement, not shared speed. Every axis completes its selected homing cycle; the next group waits for all axes. Z always runs first. A disabled R axis is skipped in multi-axis homing; an enabled R without its home/limit input configured causes failure before motion. An explicit single R reset also requires R to be enabled.

Order	Sequence	Execution
A	Z/X/Y/R	Sequential
B	Z/X/R/Y	Sequential
C	Z/Y/X/R	Sequential
D	Z/Y/R/X	Sequential
E	Z/R/X/Y	Sequential
F	Z/R/Y/X	Sequential
G	Z/(XY)/R	Grouped parallel
H	Z/R/(XY)	Grouped parallel
I	Z/(XYR)	Grouped parallel

Example: SET_HOME_STAGE_SPEEDS for Z = 1200, 600, 300, then RESET_AXIS Z mode 2 uses those three rates. Setting the three values back to zero restores Z defaults. HOME order G completes Z, then X and Y independently together, then R. Pause retains homing stages; resume continues them. Stop/emergency stop terminates the active group and prevents later groups.

These software defaults do not prove switch wiring, direction or mechanical performance. This revision is compiled for 24Y and 32Y; physical homing, loaded drives, pause/resume and emergency-stop behavior still require acceptance on the actual machine before operation.

10.2 Changing motor operating mode

The UI names are Non-Interpolation Mode and Interpolation Mode. Independent motor mode in this manual means Non-Interpolation Mode: M0 to M7 are independent motors. Coordinate mode means Interpolation Mode: M0 to M3 become X/Y/Z/R, while M4 to M7 remain independent. Switching changes legal targets and task compatibility.

1. Connect the target MCU and back up tasks and settings.

2. Open System Settings > Common Parameters and use the switch in the M0–M3 operating-mode group. Inspect the confirmation and follow the write prompts.

3. Follow required MCU/Host restart prompts, then read Slot 0 to 10 and functions.

4. Fix red incompatible instructions and download the corrected tasks. All relevant content must be compatible before run, resume and step.

Independent mode cannot execute XYZR coordinate operations. Coordinate-mode RUN_MOTOR, STOP_MOTOR and SET_MOTOR_SPEED use M4 to M7 only. Ordinary program I/O cannot write Y0 to Y3. Stop, emergency stop and queries retain their safety entries.

10.3 Data dictionary

The dictionary assigns useful names to inputs, outputs, motors, UARTs, programs, variables and errors. The current version supports separate Chinese and English names. Check the corresponding column when editing. Names do not change physical indexes, wiring or internal instruction codes.

Error definitions separate system and user codes. The generic Conditional Wait timeout system entry is no longer offered. Program execution, motion limits and counter storage retain internally defined handling. Conditional Wait instead selects specific user descriptions such as Material arrival timeout or Cylinder sensor timeout. User codes are decimal 128 to 255; the list shows the decimal ID and localized description without obsolete hexadecimal placeholders.

Select an existing user code in the data dictionary and edit both language descriptions, or use Add Error Code with an unused number and Chinese/English descriptions. Delete Error Code removes only unreferenced user codes; system definitions cannot be deleted. Error descriptions and program remarks remain part of machine data saved to the database and MCU. Unchanged verified content avoids redundant reads/writes without reducing stored content.

The compact heartbeat carries up to eight error codes. Severity, source and recovery rules remain internal to the MCU. When active-record capacity is full, new records are discarded while existing records remain, so this list is not a complete history. Continue clears recoverable Conditional Wait errors and their held outputs; task/system faults retain their internal recovery restrictions.

10.4 Main system fields and physical buttons

Field or setting	Unit/type	Entry and verification
Start/end frequency	Hz	Check start versus maximum; Hz is not millimetres per second
Linear acceleration	pulse/s²	Match load and drive capability
Pause deceleration	pulse/s²	Integer, at least 500000 and greater than linear acceleration
XYZR home defaults	12 rates in Hz; 4 release distances in pulses	Search, release and latch rates: 1-64000 Hz. Extra release distance: 0-65535 pulses per axis; default 0.
Default home order	A to I	Used by DEFAULT instructions and machine homing/initialization. Single-stage RESET uses that axis stage 1 speed; there is no separate Reset frequency.
Post-home initialization position	Enable and XYZR pulses	The corresponding sequence uses this target; check additional motion
Run-end position	Enable and XYZR pulses	Check end-of-task positioning and travel
Axis maximum X/Y/Z/R	Pulses	Match permitted mechanical travel; not a replacement for hard limits
Axis maximum frequencies	Hz	Check interpolation and each axis capability
Arc frequency/acceleration	Hz and pulse/s²	Check the complete curve and main-axis requirements
R enable and R limit	Switches	Enabling R and enabling its travel limit are separate

Field or setting	Unit/type	Entry and verification
Automatic multi-axis home on startup	Switch	Reset may immediately cause homing
Linear pulse equivalence	nm/pulse	Match transmission and microstepping
R angle pulse equivalence	Pulses/degree	Internally encoded at 1000 scale; enter the displayed units
Protocol and trace port	Logical UART	Communication and trace have different roles; verify wiring
Independent default frequency	Hz	RUN_MOTOR system permits 1 to 4000
Independent motor home settings	Enable, frequency, travel, post-home position	Check every motor/switch; M4 to M7 share parameters across modes
Vision enabled	Switch	Requires valid Host camera, ROI and calibration

Physical system buttons now provide ten functions: continuous run, stop, count reset, pause, resume, reset, continuous run/pause/resume, emergency stop/reset, pause/resume and single run. Configure input and momentary or maintained button type against actual wiring and dropdown restrictions. System Initialize and E-Stop/System Initialize have been removed from physical-button configuration; use System Configuration and Debug > All Tasks > More. Updating the Host or reading configuration does not rewrite old MCU mappings. An explicit system-parameter save or download disables those two old input assignments.

11 Vision configuration and compensation

11.1 Camera preparation and ROI

Vision requires camera drivers and supported acquisition hardware. Confirm device enumeration, continuous image acquisition and exposure before creating ROIs. Enabling vision does not prove lighting, mounting or calibration.

1. Prepare the camera driver and acquisition environment, then open Vision Coordinates. Capture a frame to inspect it, or open the live display to check image updates.

2. Create an ROI with a valid ID, region and detection settings.

3. Confirm pixel-to-pulse scale or affine calibration and check compensation direction using known displacement.

4. Use Program > Save as template > ROI to retain the inspection configuration. Align ROI IDs with REQUEST_ROI_OFFSET and compensation IDs.

5. Observe valid results before testing compensation motion under safe conditions.

The live-display control opens an image window after camera installation and acquisition setup. Use the ROI list, Add ROI, Load ROI and Evaluate ROI to manage and inspect regions; save through Program > Save as template > ROI. Move ROI commands actual motion and is unnecessary for viewing an image or checking an ROI.

11.2 Pixel scale and affine calibration

Pixel-to-pulse scale converts image displacement to motion pulses. Affine calibration maps image coordinates to machine coordinates. Recalibrate after mounting, lens, view or mechanical-coordinate changes. Host settings store ROI/calibration; task download does not reproduce camera installation on another machine.

The affine area has V2W, V2M and W2M tabs for vision-to-world, vision-to-machine and world-to-machine transforms. Check the selected direction and Vx/Vy, Wx/Wy and Mx/My coordinates before adding point pairs, calculating, testing or using matrix templates. Keep point pairs and matrices matched to the chosen transform.

Project templates include Host vision configuration, but a target still requires matching cameras, wiring and mounting. Verify ROI images and coordinate direction when reusing a template across machines.

11.3 Vision instruction order

REQUEST_ROI_OFFSET requests results; OFFSET_COMPENSATION applies an axis offset. Some move instructions reference the compensation array through compensation_id. IDs range from 0 to 7; 0xFFFFFFFF disables compensation. Distinguish missing results, detection failure and a valid zero result. Check the status bar and communication log for errors.

11.4 ROI detection rules and results

An ROI stores ID, name, image X/Y, width/height, pixel-to-pulse Mx/My and detection rules. Image coordinates and machine pulses use different units. Establish a stable image, region and reference before tuning filters. These are rule-model initial values; saved ROIs or sample configurations may differ. Use the currently read values.

Rule	Initial value	Meaning
Area filter enabled	true	Apply minimum/maximum area
Minimum/maximum area	200 / 5000	Contour area range; tune for target and image scale
Circularity filter enabled	true	Apply minimum circularity
Minimum circularity	0.5	$4\pi A/P^2$; closer to 1 is more circular
Aspect-ratio filter enabled	false	Apply minimum/maximum ratio
Minimum/maximum aspect	0.5 / 2.0	Shape aspect-ratio range
Shape match enabled	true	Match expected shape
Expected shape	Rectangle	Circle, triangle, rectangle, polygon, irregular or unknown
Angle filter enabled	false	Apply minimum/maximum angle
Minimum/maximum angle	175 / 180 degrees	Pass/fail angle range
Standard center X/Y	0 / 0	Offset reference, not a direct pass criterion
Standard angle	0 degrees	Angle-offset reference, not a direct pass criterion

Results include area, circularity, aspect, angle, angle offset, shape, center and pass/fail reason. Confirm the correct object contour before recording the reference. Do not disable every filter merely to obtain a pass. A detection pass does not verify compensation direction or machine travel.

12 Project task and section templates

12.1 Machine programs and templates

Type	Content	Entry
Single task	One task and its functions	Load into the currently selected task
Whole machine	Slot 0 to 10, points, dictionary, MCU settings and Host vision	Device-list program template library
Section	The selected configuration area	Save as template > required scope; use the matching load entry

Program & Template uses a TreeView on the left and a detail editor on the right. Its default client area is 900 by 520, matching Linkage Programs. Machine programs and template programs are separate groups; templates contain whole-machine and single-task groups. Initially no program or template is selected and the detail editor is blank and disabled. Select a leaf to display its ID, name, type, category, Task, data state, save time and description.

Template/Machine ID and Template/Machine Name are shared fields. A custom template ID contains 1 to 96 characters after trimming, excludes control characters and must be unique against other template IDs and registered MCU IDs without regard to case. New templates still receive MT/ST defaults, which can then be edited. A machine ID always remains its MCU UID. Classification, Task, state and save time are not manually editable.

Edit a template ID, name and description, or a machine name and description. Save Changes persists metadata in one transaction and rejects duplicate IDs and stale edits. Revert Changes restores the selected item. Switching, refreshing or closing offers save, discard or cancel. Metadata changes retain internal UUID, ownership, CRC and program contents and send no MCU download. The built-in Empty template remains protected from deletion and overwrite.

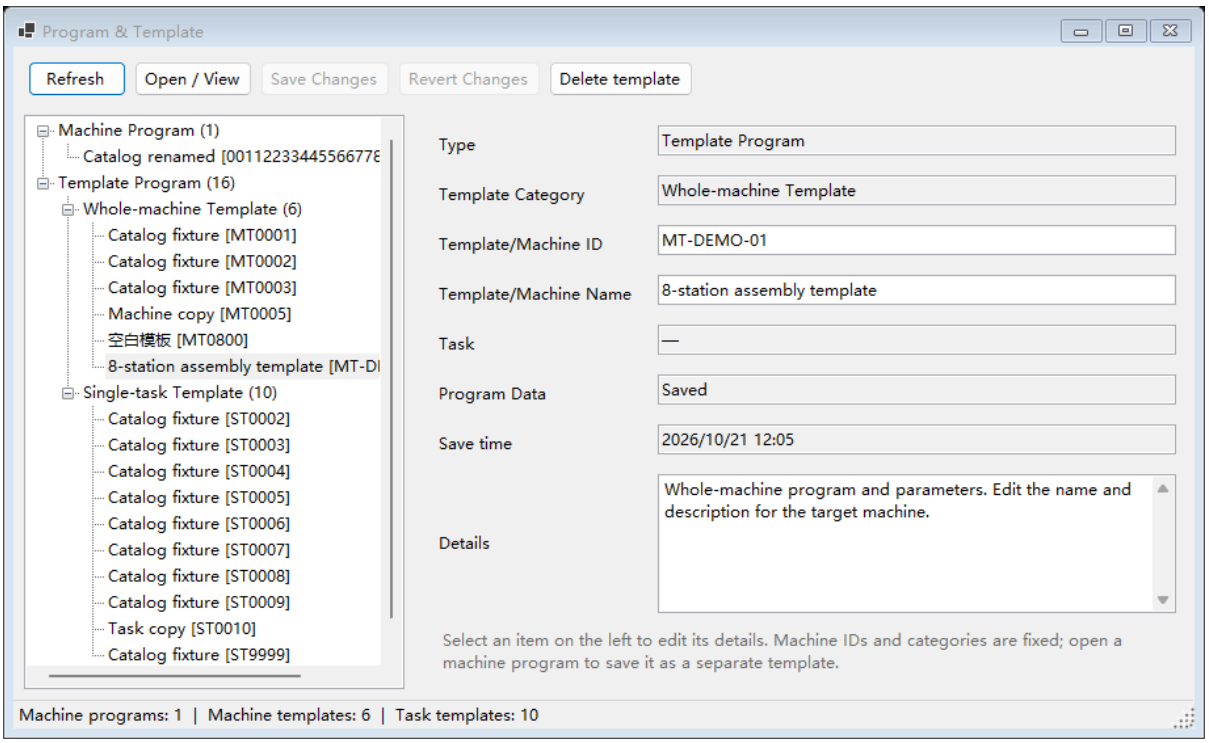


Figure 2 Program & Template with tree groups detail editing and a bottom status bar using isolated fixture data

A machine program cannot be reclassified as a template. Open its read-only viewer and choose Save As Whole-machine Template, or select a specific Task tab then Save As Single-task Template. Enter a name and description to create a new template UUID. The source machine program, MCU ID and classification remain intact. The complete raw-data tab is not a Task selection.

12.2 Editing an offline project

1. Open Program & Template, expand Template Program then Whole-machine Template, select the required entry and check its custom ID and name.

2. Click Open / View or double-click the whole-machine template. Check the offline-template indication in the title. Machine programs and single-task templates instead open a read-only viewer; this action does not connect to or write the MCU.

3. Edit tasks, points, dictionary and configuration. Offline editing/saving is available; ordinary MCU read/write, run and debug are disabled.

4. Use Program > Save as template > Whole machine. A unique name creates a record; an existing name selects confirmed overwrite. In the loading workflow, the second window uses the same name/list/description layout, with read-only fields and Load/Cancel buttons. Saving a template keeps it separate from the database machine program.

12.3 Downloading a project to a machine

1. Verify template content, hardware and motor operating mode, and back up the target project.

2. In the offline whole-machine template window, use Save / Download to open the target-download workflow, then select the intended online machine.

3. Choose an identified machine with an approved online channel. Check name, MCU UID and channel; source-template name alone is insufficient.

4. Read the full overwrite confirmation. The operation writes Slot 0 to 10, functions, comments, points, dictionary and MCU system configuration, and updates Host machine configuration.

5. The software stops, obtains configuration ownership, writes and reads back, then exits configuration mode. Inspect warnings and read the target again.

All relevant tasks are prevalidated before transfer. Communication failure can still leave incomplete state. Read back and handle reported results after errors; passing prevalidation does not prove all project data was written.

12.4 Saving and deleting templates

Program > Save as template offers Whole machine, Current task, Point matrix, Data dictionary, Current parameter section, ROI, Affine calibration, Pixel/pulse ratio and Test task. An existing whole-machine name selects confirmed overwrite; a unique name creates a record. Overwrite retains the identity and public ID and rejects stale revisions. The built-in Empty template can be renumbered, but its contents cannot be overwritten and its record cannot be deleted. Save edited contents under a new name.

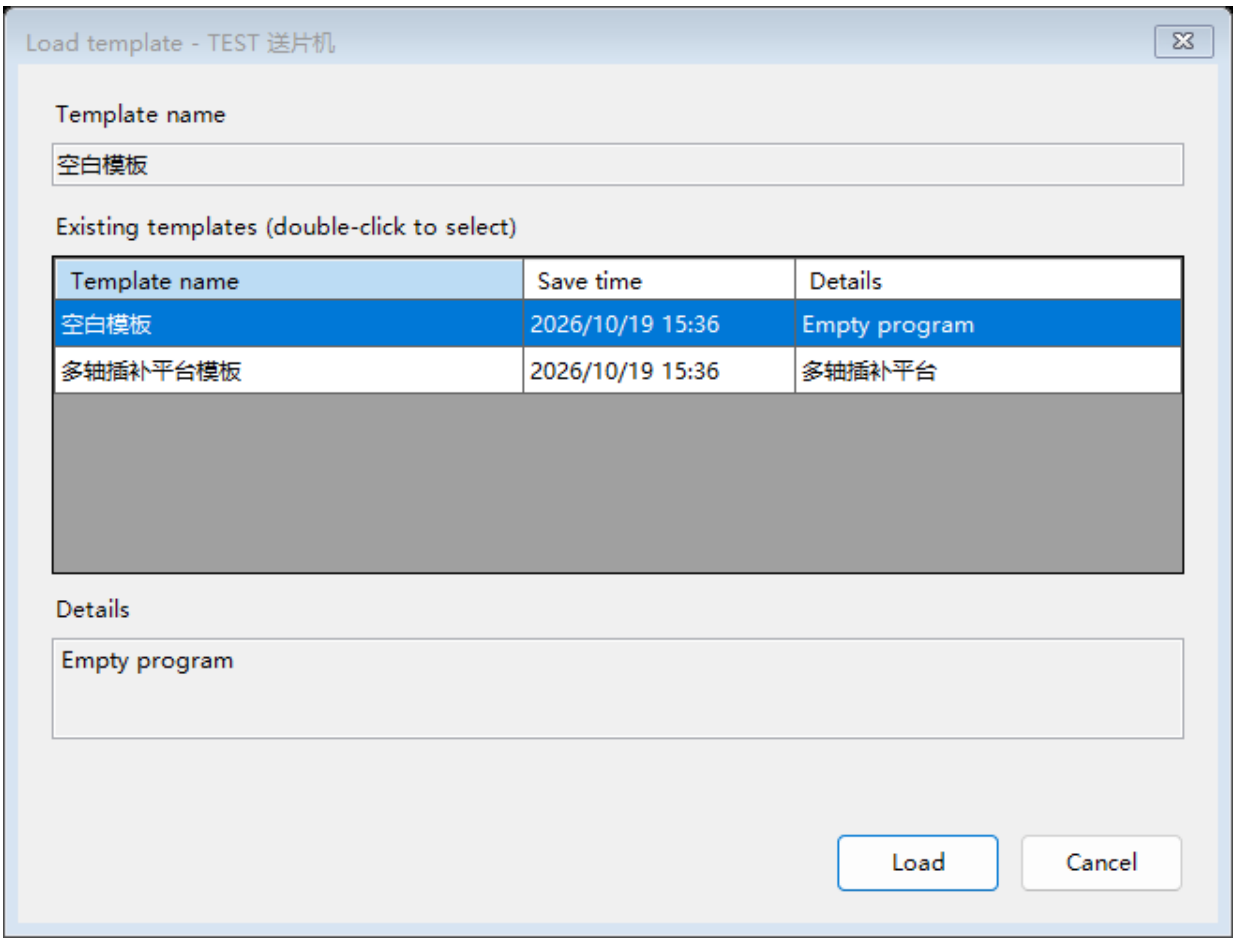
12.5 Choosing templates and empty programs

Choose Load template in the program-source dialog and confirm. Only then does HOST read template names, saved times and descriptions and open the separate list. Its title includes the machine name. Select a row and click Load, or double-click it. The name and description are read-only in this loading mode; it cannot overwrite a template.

Cancelling or closing the list returns to the source dialog without loading anything. The selected record is identified by its ID, so matching names cannot select another record. Whole-machine templates are available for the relevant scope; task and point scopes also include their matching template types.

Explicitly selecting the built-in Empty template restores this Host's factory values in the editor: empty Slots 0 to 10, functions, remarks, points and user names, plus default motion/button settings, pixel/pulse ratios, ROI and affine calibration. Review and explicitly save or download before changing the machine database or MCU. ESP, Wi-Fi, MCU identity, the template library and history are retained. Automatic empty fallback after missing database data, a deleted template or confirmed empty MCU content preserves device settings; it is separate from this explicit restore action.

A timeout, incompatible format or corrupt nonempty program/dictionary is a load failure, not an empty program. A failed load must not replace the editor with blank content. Loading alone does not write the MCU or automatically save the machine program. Use a distinct template name if you edit the built-in empty contents.



Template-loading example using the save-as list layout. Names are example records.

13 Host automation programs

13.1 Where Host automation runs

Click Linkage Programs in the device list to open the PC automation editor. Its interpreter runs on the Host and sends machine commands; it does not download its script as an MCU task. It cannot continue after the PC exits and does not automatically resume at the interrupted row after restart.

The taller instruction tree and parameters occupy the left side; the current automation program is on the right. The toolbar order is Append, Insert, Up, Down, Delete. Double-click still appends an instruction; its former on-screen hint was removed. Loading, validation and execution messages use the bottom StatusStrip. The grid starts with Mark and ends with Comments. Machine names are displayed while stable MachineId values are stored, so renaming does not retarget the script.

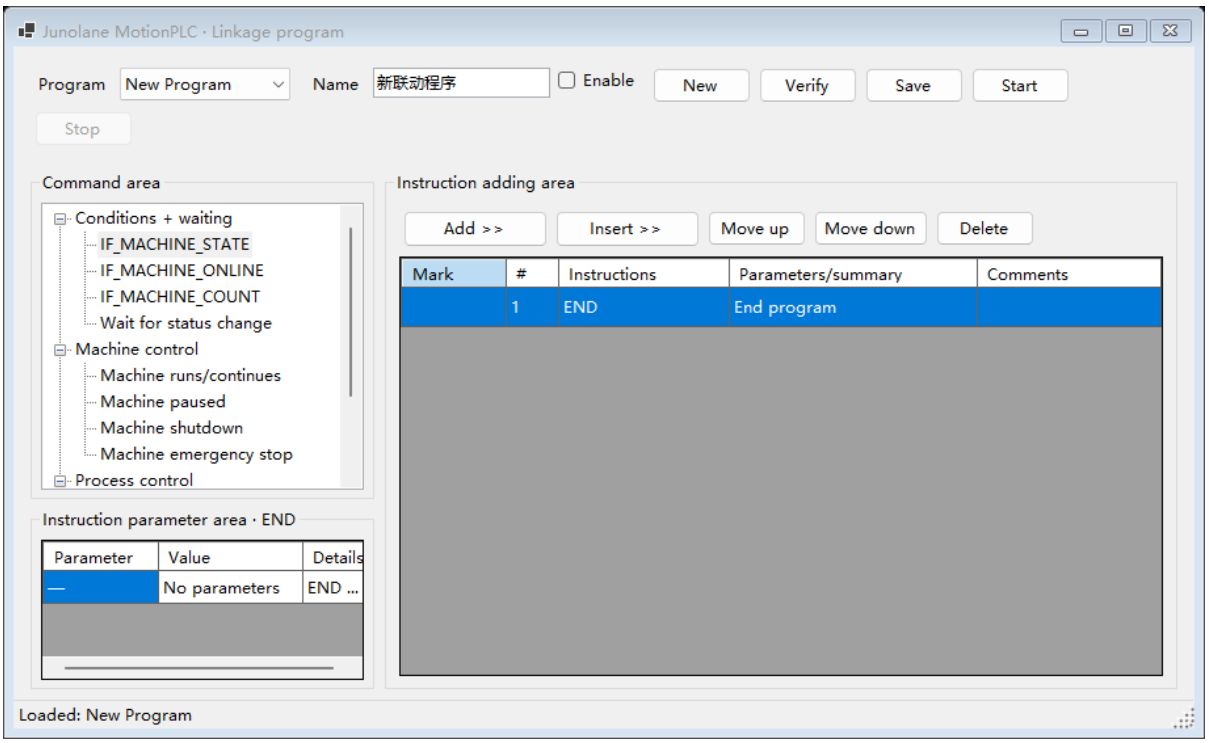


Figure 3 Automation layout with Append and Insert before Up Delete last and a bottom status bar with execution disabled

13.2 Automation instructions

Code	Purpose	Check
IF_MACHINE_STATE	Conditional state jump	Stable-time condition and target Mark
IF_MACHINE_ONLINE	Conditional online/offline jump	Distinguish channel state from run state
IF_MACHINE_COUNT	Compare current or total count	Check online/live count availability
WAIT_MACHINE_STATE_CHANGE	Wait for a change from the entry state	Avoid repeated triggers from a persistent state
MACHINE_RUN	Run or resume machine	Sends a real start command
MACHINE_PAUSE	Recoverable pause	Keeps resume context
MACHINE_STOP	Normal stop	Ends tasks
MACHINE_ESTOP	Software emergency stop	Depends on communication
TIMER	Delay the interpreter	Greater than zero, at most 24 hours
JUMP	Jump to script Mark	Target must exist
END	End the script	Does not restart automatically

13.3 Saving starting and stopping

1. Create the program, select registered machines and complete Marks, conditions and timers.

2. Save and explicitly enable execution. New scripts default to execution disabled.

3. Confirm target states and start manually. Inspect the log/current row and command acknowledgements.

4. Stop and wait for interpreter cancellation. This does not prove all controlled machines have stopped; check each independently.

Automation TIMER delays are greater than zero and at most 24 hours, and do not resume after cancellation/Host exit. These differ from MCU TIMER_START. Scripts and run/row events are stored in the configured database; AutomationBackups preserves program-version snapshots.

14 Monitoring and operation

14.1 Run pause stop emergency stop and reset

Action	Result	Follow-up
Run/resume	Start an idle task or restore pause context	Check emergency inputs, then confirm new heartbeat and task row
Pause	Decelerate while preserving context	Resume only when conditions permit
Stop	Decelerate and end tasks	Cannot resume the old pause context
Emergency stop	Request an immediate abort and latch; execution may be delayed during synchronous homing	Confirm the MCU result and machine state; an ACK alone is insufficient
Reset	Request MCU system restart and clear the corresponding emergency-stop latch	Require execution confirmation before reconnection; automatic homing may follow startup
Multi-axis reset	Home axes without restarting the MCU	Stop every running or paused task 0 to 10 first
System Initialize	Stop tasks and restore startup state without rebooting	Use More; requires confirmed interpolation mode and idle long-command state

Device-list Reset requests an MCU restart; after execution is confirmed, wait for a new handshake and heartbeat. Power-on multi-axis automatic homing uses the complete three-stage profile and saved A to I order when enabled. Host multi-axis homing and System Initialize in More use machine defaults but have different conditions and effects, as explained in Sections 14.1.3 and 14.1.4. To restore factory editor content, explicitly load the system Empty template in Section 12.5, review it, then save or download.

The row Run/Pause action changes with machine state. Starting after Stop differs from resuming after Pause. Normal stop requires compatible firmware; unsupported commands fail explicitly and do not silently fall back to emergency stop.

Normal stop acknowledgement is followed by new-heartbeat confirmation of stopped/idle state. The MCU finishes deceleration and ends tasks, with its timeout safety fallback. If the Host receives acceptance but no state confirmation within its limit, inspect the actual machine and log.

14.1.1 Emergency input checks before starting

Before a Host run, single-cycle run, resume, single-step or System Initialize action, the Host reads the emergency-input configuration stored in the MCU and checks the most recent valid heartbeat inputs. The same check covers jogging, homing, motor and precision tests, and machine starts from the device list and automation. Heartbeat inputs have communication and sampling delay. They are not an independent sample at the instant of a click and do not replace a hardwired emergency circuit.

The Host checks the emergency mappings actually returned by the MCU. A pressed input blocks a start or initialization and identifies its X number. Compatibility checks still include the old E-Stop/System Initialize input;

removing its new configuration control does not erase an existing MCU mapping. Release the switch and wait for an updated input state before retrying. If reset is still required, follow its existing procedure. An interlock rejection does not trigger an automatic program download or start retry.

If the emergency configuration cannot be read or recognized, the Host reports that it cannot verify the configuration. If a configured emergency input has no valid input heartbeat, or the communication timeout has elapsed, wait for heartbeat recovery and check the connection, MCU version and configuration read. If the MCU explicitly reports that neither emergency input is configured, this preflight does not require an input heartbeat. System Initialize still requires a fresh machine runtime snapshot. This exemption does not establish that a physical emergency circuit is fitted.

Emergency preflight does not block Stop, Pause or System Reset; their own connection, permission and current-operation requirements still apply. System Initialize is protected and must also meet Section 14.1.4. These Host checks do not themselves require reflashing, but configuration-read and runtime-control compatibility with older firmware must be checked separately.

14.1.2 Task display after a reset

After a successful MCU reset confirmation or an MCU reset notice, the Host refreshes all eleven tasks: populated tasks show Stopped, empty tasks show N/A, and execution-row markers are cleared. While waiting for a stopped state or reconnection, queued pre-reset RUN heartbeats cannot restore the old running display or highlight. After a new application connection, fresh valid heartbeats update the display, including tasks started from the physical panel.

A rejected reset, or an ACK without MCU execution confirmation, is not shown as a successful reset. An unconfirmed result is unknown: it proves neither a completed restart nor that nothing executed. The Host does not close, reconnect or clear task states merely because confirmation is missing. Reconnection follows a successful reset result or an MCU reset notice. Check Reset complete against fresh valid heartbeats and the actual machine. System Initialize likewise cannot be reported complete merely because it was sent or acknowledged.

14.1.3 Task checks before multi-axis homing

Multi-axis reset is a homing action, not an MCU restart. The Host checks every task from 0 through 10. If any task is running or paused, it asks for Stop first and identifies the affected task numbers. Instruction-test task 10 receives the same protection as ordinary tasks, regardless of which task is selected.

Stop the tasks and wait for the MCU heartbeat to confirm stopped states before homing. An immediate Stopped repaint does not release homing while the latest actual heartbeat still reports a paused task. Another homing action is also blocked during system initialization. Do not resume execution merely to bypass the homing guard.

14.1.4 System initialization access and protection

Use System Configuration and Debug > All Tasks > More > System Initialize or E-Stop / System Initialize. Initialization does not reboot the MCU. It stops all tasks and motors, clears paused execution context, restores variables and I/O to startup state, and homes axes in interpolation mode. Programs, points, counters and system configuration are retained. Read the confirmation because mechanical motion may occur.

Ordinary running or paused tasks, including task 10, may enter this stop-then-initialize flow. The rule requiring all tasks to stop before standalone homing therefore does not apply unchanged. Initialization still requires the same connected MCU, motion authorization, MCU-confirmed interpolation mode, valid configuration ownership, released emergency inputs, a fresh heartbeat and no unfinished homing or other long command. Unknown or non-interpolation mode, missing or stale runtime status, and busy operations produce an explanation and block initialization.

Repeated clicks do not queue another initialization. New run, resume and homing actions are protected while Stop remains available. Waiting for initialization completion does not monopolize the synchronous command channel. If the connection changes, the earlier result cannot mark the new connection initialized. Only an MCU execution-success result refreshes all eleven task states and execution rows.

E-Stop / System Initialize sends Stop first and initializes only after an MCU execution-success result confirms that stop. Later initialization permission, mode or busy checks do not prevent the stop request from being sent. An unconfirmed stop reports an unknown result, preserves unconfirmed task states and prevents subsequent initialization.

Existing firmware can defer a software stop while synchronous homing occupies its command task. Priority transmission by the Host does not prove that a motor has stopped, and an ACK confirms receipt only. Without execution confirmation, inspect the machine under the site safe-stop procedure. Do not repeatedly initialize or treat a display message as proof of a mechanical stop.

These restrictions apply to Host initialization controls. For the SYSTEM_INITIALIZE program instruction and its operating-mode behavior, see Section 8.21. Do not apply the Host control interpolation-mode restriction directly to that program instruction.

14.2 Monitoring and manual outputs

Open System Monitor to inspect global variables, task-stack watermarks, Pulse ISR, motors and X/Y states. Task Usage and Task Timers share the upper-right monitor region as tabs; switching tabs does not compress the I/O controls. Task usage describes Host program-image storage, while MCU statistics require their explicit read action. The variable column Mode is read-only V19; V0 to V18 are ordinary variables. A dash means unavailable, not zero.

System Monitor protects controls according to the operating mode confirmed by the MCU. Interpolation mode disables Y0 to Y3 and M0 to M3 while still displaying direction and running states. Non-interpolation mode allows M0 to M3 to start and stop tests in the same way as M4 to M7. Until the MCU mode is confirmed, Y0 to Y3 and M0 to M3 remain protected; changing the editor draft alone does not release them. Other supported Y outputs and M4 to M7 retain their controls, and disabled controls still receive heartbeat updates.

Control	Interpolation	Non-interpolation
Y0 to Y3	State display only	Manual toggles available
M0 to M3	State display only	Continuous test / stop
Other supported Y outputs	Manual toggles available	Manual toggles available
M4 to M7	Continuous test / stop	Continuous test / stop

In non-interpolation mode, M0 to M7 toggle continuous motor tests. The first click starts continuous pulses in the positive direction at that motor's configured default speed in Hz. The next click stops its pulses immediately. Releasing the mouse does not stop the motor, and no pulse count or travel distance is specified. These tests start and stop directly without acceleration or deceleration; programmed motion follows its own instruction parameters.

PWM0 to PWM3 provide the M0 to M3 pulse signals; Y0 to Y3 are their direction signals. A Y click toggles its output level without generating pulses. Physical direction depends on wiring and drive settings. Stop before changing direction to avoid an abrupt reversal. Every M-button test start sets the positive direction again. Active tasks or a busy motion engine cause firmware to reject a new test start.

Single-step, move-to-point, homing and repeatability tests cause motion. Confirm travel, switches, drives, speed, permission and mode. Diagnose errors/timeouts before retrying motion commands.

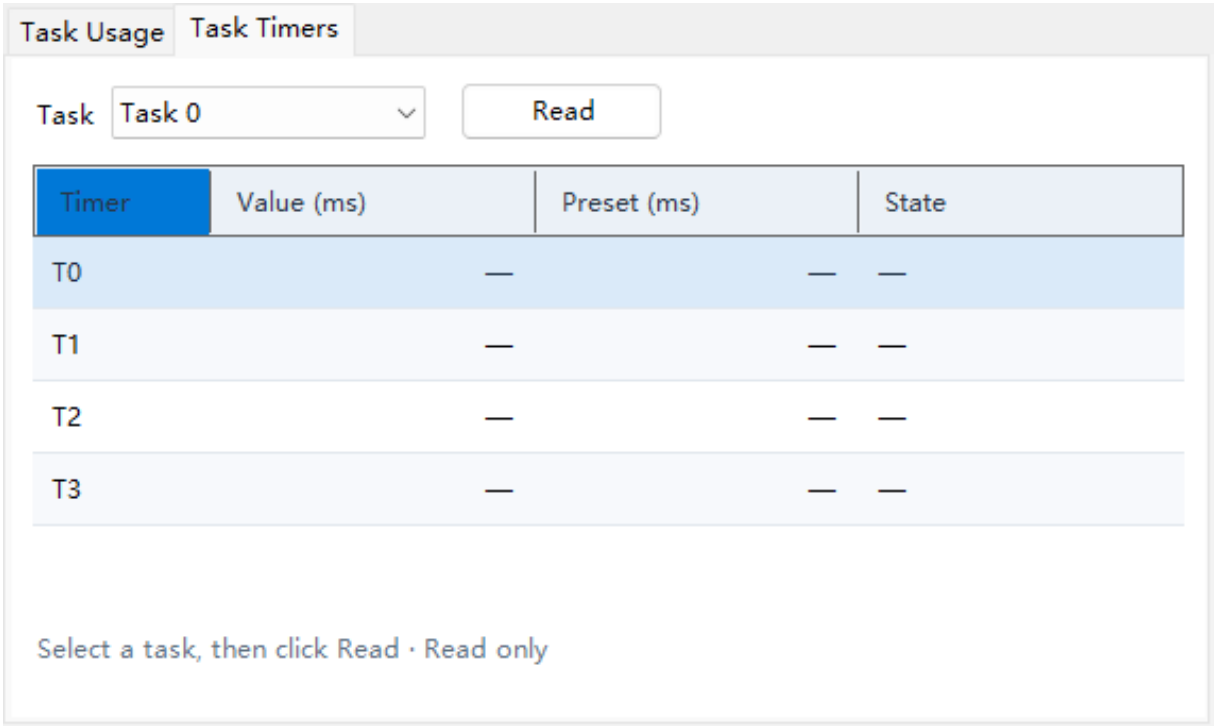
Task Usage lists slot, task name, used bytes, capacity and status. Slots 0 to 9 are ordinary tasks, Slot 10 is the test slot, and the point table is separate. Each task image has a 2048-byte capacity; the point table has 4096 bytes. These values describe storage, not runtime RAM or CPU load. If a point entry shows Local .../MCU ..., compare the values separately; the display alone does not establish a successful download.

14.3 Reading task timer values

Open System Monitor > Task Timers in the upper-right tab area. Select Task 0 to Task 10 and click Read. T0 to T3 show Value (ms), Preset (ms) and State: Idle, Timing, Expired, Paused or Stopped. Values come from the MCU rather than a Host stopwatch or V-variable copy. Paused and stopped rows retain their frozen readings.

Read requests exactly one snapshot for the selected Task. Opening the view, receiving heartbeats and changing the Task selector do not read automatically. Each Task keeps its own last snapshot and read timestamp; switching back restores that display without clearing or resetting any MCU timer. An unread Task shows dashes. Read at MM-dd HH:mm:ss identifies a snapshot rather than a live counter.

If reading fails, the last snapshot is retained and the failure is shown. If communication is busy, retry Read later; it does not queue behind a program download. A result received after changing the selector is stored for the originally requested Task. Disconnecting or changing the MCU session clears Host snapshots to prevent old-device readings from appearing on a new connection; it does not send a timer-reset command. Unsupported firmware prompts an MCU update.



Task Timers before the first manual read. Dashes mean no snapshot is available.

14.4 Logs and errors

Logs capture connections, commands, read/write and errors; new heartbeat confirms state. Record time, MCU UID, channel, action, full error and task state. A historical success is not proof of the latest action.

15 Host and MCU updates

15.1 Host updates

Use an official installer or signed update package. Save edits and close the running Host first. Preserve Data for V4 updates. Historical V3 migration uses a dedicated Downloader workflow; do not apply old updater assumptions to this version.

1. Back up programs, templates and database and confirm the running path.

2. Install the official Host update package and verify its version, signature and completion. Menu > Check for Updates > Automatic Update and Manual Firmware Update are MCU firmware actions, not the Host installer.

3. Restart from the same installation and verify machine registration, language, database machine programs and templates.

4. Reconnect and check heartbeat/readback. Host update does not imply MCU firmware update.

15.2 MCU application firmware

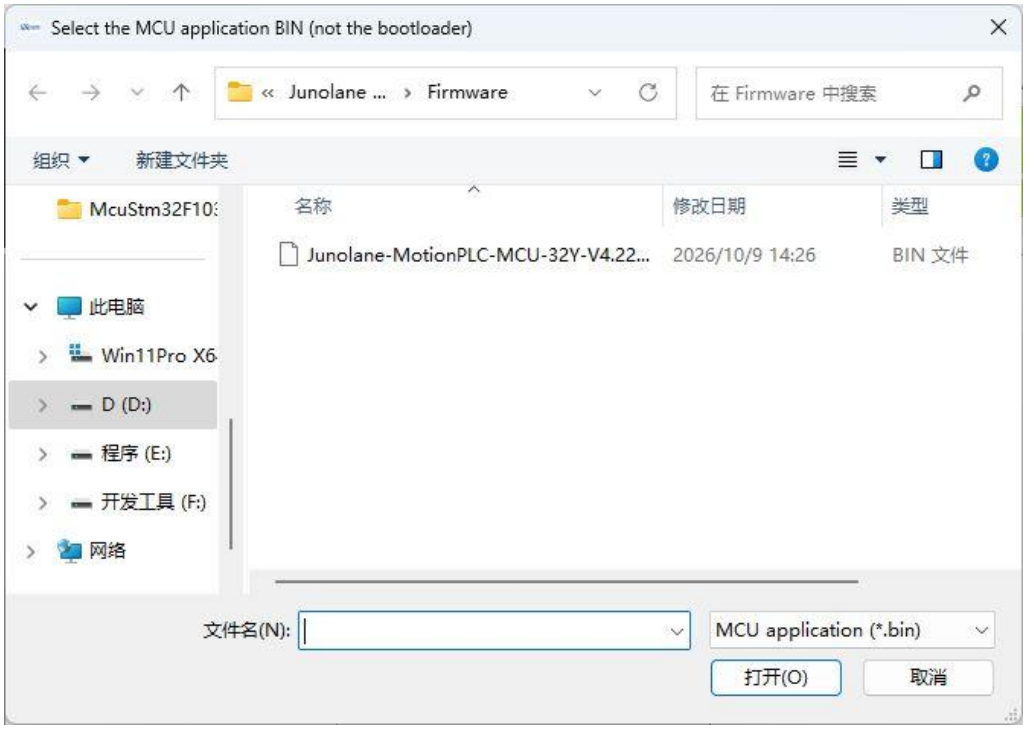


Figure 4 Manual Firmware Update opens the MCU application BIN chooser; Windows controls follow the operating-system language

1. Select the correct machine and channel, confirm MCU identity and 24Y/32Y hardware, and back up user data. A confirmed Bootloader shows Waiting for Firmware; choose Write Firmware to recover. An eligible saved offline machine without confirmed Bootloader information may still offer Connect & Load Program (Upgrade). Keep power stable and select the matching application BIN.

2. For a local application BIN, use Menu > Check for Updates > Manual Firmware Update. The chooser specifies MCU application BIN, not Bootloader. Automatic Update uses the matched signed MCU package and can begin firmware writing; use it only after completing the preparation above.

3. Do not operate the machine or switch channels during transfer. If ENTER_BOOTLOADER has no complete response, Host does not repeat ENTER; it queries read-only Bootloader INFO at most 20 times. A negative ACK or RESULT stops entry. No valid INFO or missing Bootloader write protection stops the update before BEGIN or DATA.

4. Rediscover, claim and handshake over the original channel, then verify the application version and continuing valid heartbeats. Waiting for Firmware clears and the connection button returns to its normal action. A BLE connection or reappearing COM port alone does not establish application recovery.

5. Read tasks, parameters and text and verify version/content. A reappearing COM port alone is insufficient.

The application-update flow supports complete-page resumption. Bootloader, application Firmware and Host are separate components. Application update does not replace initial MCU flashing. An incomplete entry response does not prove clock damage or require Bootloader reflashing. Recovery is not permission to interrupt power or bypass write protection.

15.2.1 Firmware write states and interrupted transfers

For a normal online application, Connect & Load Program opens the ordinary program-source chooser. A confirmed Bootloader changes Status to Waiting for Firmware and Connection to the channel plus the same state. The button becomes Write Firmware, and the BLE summary includes the number waiting. Before confirmation, BLE may show Waiting for MCU ID; missing heartbeats alone do not establish Bootloader mode.

Write Firmware opens recovery and the MCU application BIN chooser. Recovery searches Updates and Firmware in the installation directory for the original image with exactly the length and CRC32 recorded by the Bootloader. A match preselects its directory and filename. A different image with the same version label is not automatically selected as the original.

If no match is found, select the application BIN for the correct hardware manually. Writing starts only after file selection, hardware checks and upgrade confirmation. Canceling the chooser does not start writing. To resume the original transfer, use the same image; its name or version label alone is insufficient.

During writing, Status and Connection show Writing Firmware. The button shows Writing Firmware… and is disabled to prevent duplicate actions. Do not switch channels or command motion. If an unsuccessful update leaves the MCU in Bootloader, enter recovery again. The same flow applies when the BLE transport remains connected.

After transfer, verify integrity checking, the run command, application version and continuing valid heartbeats. A valid application identity or heartbeat clears the waiting state; a short or malformed heartbeat cannot clear it. An intact Bootloader and working recovery route can recover an interrupted transfer over BLE without using ST-Link solely because of the interruption. Determine any need for hardware servicing from the actual responses.

15.3 Choosing 24Y and 32Y firmware

24Y and 32Y identify hardware output models, corresponding to Y0 to Y23 and Y0 to Y31. They do not describe 24-byte or 32-byte packets. Read the model reported by the MCU and verify the physical PCB. Do not infer hardware from the device name, Bluetooth module name or firmware file size.

An online update manifest that supports model selection can provide both 24Y and 32Y firmware. The Host selects the package matching the MCU handshake model and checks downloaded size, SHA-256, CRC and version. Use the same model package throughout the update. Confirm Host and MCU updates separately.

If a 32Y machine reports that the channel provides 24Y firmware only, that channel has no eligible matching package. Use a publisher-supplied local BIN explicitly intended for 32Y, or wait for an online release providing 32Y and a compatible Host. Do not rename a 24Y image to bypass checks. For an unknown model or conflicting markings, verify the PCB and firmware source first. The local cross-model entry is only for first-time migration after confirming the physical PCB; follow its confirmation prompt.

15.4 Completion criteria

Require correct CRC, compliant running MCU version, the expected MCU UID, a ready application session on the original channel and sustained heartbeats, plus valid readback. Progress at 100% or CRC success alone does not confirm the application is running. CRC does not validate mechanical settings or all task behavior. Inspect registration and the original channel if recovery is incomplete.

16 Data backup and recovery

16.1 Backup scope

The installed Data directory is D:\Program Files\PLC-F103ZET\Data unless LYXMOTIONPLC_DATA_DIR specifies another location. Data/database.json selects the PostgreSQL server and its identity. Preserve the local directory and any original SQLite source/backup files; server data is not contained in a copy of that directory.

Content	Location	Purpose
Database and connection settings	PostgreSQL server plus Data/database.json; original SQLite source retained	Accounts, machines, tasks, templates and server backup
Project snapshots	ProjectBackups grouped by MCU UID	Per-machine recovery
Automation versions	AutomationBackups grouped by ProgramId	Script history
Firmware/updates	Firmware and Updates under installation	Update/recovery material
Accounts	Configured PostgreSQL account storage	Login and permission recovery

16.2 Complete backup procedure

1. Save changed machine programs to the database and retain templates as needed. Record MCU UID, hardware and Host/MCU versions.

2. Close Host normally and ensure database writing has stopped.

3. Create a native PostgreSQL backup of the configured database and verify it before migration or recovery. Also retain the complete Data directory and firmware/configuration backups in a dated location. For an old SQLite source, preserve any active WAL data together with the source file. Copying Data alone is not a PostgreSQL server backup.

4. Record original executable and data paths and check backup size/readability.

5. Restore to a test copy or use the approved recovery workflow. Check registration/templates/drafts before MCU readback.

16.3 Recovery and another computer

Keep a copy of current data before recovery and use compatible versions. MCU UID associates machines, while COM numbers, adapters, networks and endpoints may differ on another PC. Compare readback with images, drafts and templates before deciding to download recovery content. Do not overwrite another active installation without inspection.

Uninstall, full-chip erase or chip replacement may retain different data. Ordinary Host/application updates are not full-chip erase. A new MCU has a new UID and needs registration as a new machine.

17 Troubleshooting

17.1 No target machine

Check MCU power, bridge, COM driver or BLE adapter and inspect candidates. A MAC does not prove MCU handshake.

17.2 BLE connected without MCU ID

Inspect candidates, module-to-MCU UART wiring, baud rate and protocol. A Bootloader may not announce an application MCU ID. After allowing time for application identity, a BLE connection with a registered MAC receives a read-only Bootloader INFO query. Only a valid reply establishes Waiting for Firmware. An unidentified device, missing heartbeat or GATT connection alone is insufficient. MAC matching selects a recovery route; it does not authenticate the MCU. Duplicate owners or routes, changed MAC, closed channels and PendingApproval do not qualify. New devices still require manual approval. Recovery allows manual firmware writing while motion and ordinary template download remain unavailable.

17.3 Multiple channels for one machine

Verify MCU UID, close its HOST window, select the channel and check binding to the correct MCU.

17.4 Connect and load control disabled

Check operation permission, UID, approval, online channel and current selection. Offline machines cannot open real-time control.

17.5 Rename disabled

Check configuration permission and stable identified registration. Names must be non-empty and at most 100 characters.

17.6 Red task rows or tree entries

Check motor operating mode and incompatible operations; fix all relevant slots/functions and download.

17.7 Saved but still running the old task

Database-only saving does not change the MCU. Check the chosen save action, machine UID and scope. If database saving succeeded but download failed, retain the database copy and inspect device readback. Use Consistency check to compare the database and MCU; it does not compare unsaved Host edits.

17.8 Conditional jump stays on the row

True current-row conditions repeat; false advances. Check input/Timer state and condition direction.

17.9 Timer expires without stop

TIMER_START sets Timeout only. Add explicit conditions and stop/handling logic.

17.10 Normal stop fails

Check firmware support, new heartbeat and actual motion state; follow the reported reason.

17.11 Configuration ownership or BUSY

Check idle state, another owner and long-command completion.

17.12 Duplicate or missing Mark

Check case-insensitive uniqueness, reserved names, length, characters and function scope.

17.13 Program or text capacity exhausted

Check the 2048-byte slot including functions, 4096-byte point image and separate compressed text capacity.

17.14 Circle or three-point arc rejected

Check collinearity, integer geometry, radius, complete travel and MCU start position.

17.15 Offline template run disabled

Expected for ordinary run/debug controls. Save the template explicitly, then use Save / Download to select an identified online target for the separate download workflow.

17.16 Vision offset direction wrong

Check ROI, mounting, pixel scale/affine calibration; observe before approved low-speed testing.

17.17 COM returns after update but reads timeout

Check Status and Connection first. For Waiting for Firmware, follow Section 15.2 and prefer the original BIN used by the interrupted transfer; interruption alone does not establish Bootloader damage. After confirmed CRC/RUN, wait for discovery, claim and handshake on the original channel and valid application heartbeats. Power-cycle only after transfer has ended. If power-up works but RUN repeatedly fails, Bootloader servicing with ST-Link may be needed; application BIN updates cannot replace it. Back up Flash and option bytes first because changing protection may erase Flash. Preserve original RDP, USER, DATA and unrelated WRP values, change only Bootloader write protection, and verify the recovered version, heartbeat and data.

17.18 Chinese user names in English UI

User-authored text is preserved. For Chinese inside standard dynamic messages, record full text and installed version.

17.19 Unexpected database/template contents

Check executable directory, data override and whether Data came from another version.

17.20 32Y update reports 24Y-only firmware

Check the MCU model and channel in Chapter 15. Use an explicitly matching 32Y release; do not flash or rename 24Y firmware.

17.21 Status ends in -0 -1 or -2

See the local authorization-record suffixes in Chapter 3. Read the suffix separately from operation state and online counts.

A useful fault report includes executable path, versions, MCU UID, hardware, channel, steps and full error. It does not require login passwords, production keys or a whole sensitive configuration directory.

17.22 Conditional Wait timed out

When all running tasks pause, read the error code and dictionary description, then locate the Conditional Wait rows that reference it. Check operands, sensor wiring and timeout. The compact heartbeat carries codes rather than a complete task/row diagnosis. Keep Paused is the default. After resolving the cause, Continue or the configured physical Continue input clears this timeout error/output and retries the same row with a fresh interval. A false condition can time out again.

17.23 Timer readings do not change

The monitor displays the last manually read snapshot. Click Read to update and check its timestamp. Task selection changes only the displayed Task snapshot. A dash means unread/unavailable; a retained timestamp after failure is not a new reading. See Section 14.3.

17.24 Host reopens with paused status or a rejected handshake

Closing or reopening the Host does not reset the MCU or automatically resume a task. If Paused appears immediately, inspect tasks 0 through 10 and fresh heartbeats. The MCU may have retained an earlier paused context. Do not resume the machine merely to establish a connection.

An ordinary paused task with no unfinished long command permits normal connection. Older MCU firmware may reject a Host handshake while homing remains active, even when BLE is connected and heartbeats arrive. The matching updated MCU application permits that handshake during active long-command states. The handshake itself does not clear pause, resume tasks or replace the active homing context.

A Host-only update cannot remove the older MCU restriction. Follow Chapter 15 with the matching 24Y or 32Y application firmware, then verify connection and task states. Connection does not by itself authorize all parameter or program reads while homing remains unfinished. If synchronous homing still occupies the MCU command task, handshake execution can also wait; immediate connection is not guaranteed for every busy state. Investigate under the site safe-operation procedure rather than automatically stopping, resuming or resetting to conceal a connection failure.

18 Commissioning examples and handover

18.1 A first I/O task

1. Open an approved online machine with its MCU program selected. Choose a confirmed replaceable empty task and retain a backup.

2. Add IO_WRITE with an allowed Y, explicit open/closed state and appropriate delay.

3. Add DELAY_MS such as 100 ms, another IO_WRITE restoring the original state, and END.

4. Use Save / Download > Current task > Save to database and download to MCU. Check completion and readback.

5. Single-step only with isolated actuators or site approval, observe I/O and then follow production commissioning.

This example describes sequence construction and does not assign a mechanical function to a Y output. Never copy an unknown output number into live operation.

18.2 Waiting for a condition with a timeout

First define user code 128 as Material arrival timeout with both Chinese and English descriptions in the data dictionary. Add Conditional Wait under Variables & Flow: X0 = Closed, 5000 ms, Y4, error code 128. Check wiring before choosing Y4; No Output is also available. True advances and false waits at this row; other tasks keep running until timeout.

Row	Operation	Example setting
1	Conditional Wait	X0 = Closed; timeout 5000 ms; alarm Y4
2	Task End	Reached only after X0 satisfies the comparison

If the condition remains false for five seconds, all running tasks pause, code 128 is recorded and Y4 stays active. Resolve the cause, then select Continue or use the configured physical Continue input. This timeout error/output clears and the row is checked again with a fresh five-second interval. Keep Paused is the default. Timeout zero waits indefinitely but still requires a user code. See Section 8.35.

18.3 Project-template handover

1. Load and check the source machine's complete MCU program, points, dictionary and settings; use Save as template > Whole machine.

2. Describe hardware, purpose, date and recalibration needs.

3. Open the whole-machine template offline, inspect Slot 0 to 10, functions and vision, then save it explicitly as a template.

4. Back up the target and match its hardware/mode; verify UID and use the template window's Save / Download target workflow.

5. Read back and verify home, directions, travel, vision and low-speed tasks. Download success does not replace mechanical acceptance.

18.4 Shift or machine handover

Item	Record	Check
Identity	Name, MCU UID, hardware, channel	□
Versions	Host, MCU, manual revision	□
Configuration	Mode, home, travel, speed, equivalence	□
Tasks	Downloaded revision, slots, functions, comments	□
Templates/backups	Names, Data path, backup time	□
State	Stopped/required state and no pending error	□
Permissions	Next operator account and authorization	□

Appendix Version checks and revision record

Version checks

When using this manual, record the installed Host version, MCU firmware version and hardware model. After changing computers or updating software, check the device name, MCU UID, connection channel and program content again.

At handover, retain this manual revision, task and parameter backups, template names and backup dates. Site records should identify the program and configuration actually used by each machine.

Applicability

This document covers the current software and workflows. Site personnel must verify wiring, travel, speed, load, home directions and vision calibration. Defaults have not been physically validated on every hardware combination. Apply the corresponding operation, programming and maintenance permissions.

Maintenance and revision

Recheck this manual and increment its document revision when controls, parameters, instruction semantics, template content or update workflows change. Identify a formal manual by version, date and verification evidence, not merely by latest.

Revision	Date	Content
R1.1	8 October 2026	Initial bilingual V4.22 review draft
R1.2	9 October 2026	Aligned device labels and floating task buttons; explained authorization suffixes, storage usage and 24Y/32Y update handling
R1.3	9 October 2026	Aligned current read buttons, Duration, draft status and MCU readback database confirmations
R1.4	9 October 2026	Formal edition with actual bilingual screenshots; clarified language restoration and Host versus MCU update paths
R1.5	10 October 2026	Manual approval once per MCU UID; immediate discovery of BLE modules advertising FFF0.
R1.6	10 October 2026	Release old BLE sessions and actively reconnect bound modules; preserve MCU UID approval.
R1.7	10 October 2026	Template type labels and names; direct MCU reads; save a whole-machine template when closing after edits.
R1.8	10 October 2026	Bounded Bootloader entry verification and saved BLE recovery routes; preserve manual approval and protection.
R1.9	11 October 2026	XYZR independent stage defaults, zero/default overrides and A-I grouped homing. Unified Reset speed and legacy single-stage execution are retired; old configurations retain their layout.
R1.10	11 October 2026	ZE: Multi-Axis Home adds Mode before Order, with Single-stage and Three-stage choices. Single-Axis Home uses Axis and Mode labels; matching MCU firmware is required.
R1.11	12 October 2026	Per-axis release pulses (0-65535); legacy default 0; matching MCU update required.
R1.12	12 October 2026	Rename startup homing checkbox; behavior unchanged.
R1.13	14 October 2026	Require database connection setup before first login; preserve source data.
R1.14	19 October 2026	Two-column task actions; Conditional Wait and timeout alarm recovery; manual Task Timer snapshots; compact status and operation selection; template overwrite and capacity notices.
R1.15	19 October 2026	Unified load/save/close workflows, 23-button controls, template selection, empty fallback, explicit consistency checking, complete progress and optional wait output.
R1.16	19 October 2026	Interpolation monitor protection and continuous motor tests; unified catalog and MT/ST IDs; explicit factory-default template loading.

Revision	Date	Content
R1.17	20 October 2026	Combined Configuration window and screen placement; timer lifecycle actions; user error codes; complete save/compare/close progress and unchanged-data transfers.
R1.18	21 October 2026	Program/template TreeView and detail editing, machine save-as templates, automation toolbar and status bar; one-click PostgreSQL materials and existing-account rules; pending Win7 acceptance with the same USB BLE Central adapter.
R1.19	21 October 2026	Firmware waiting and writing states, state-dependent connection actions, explicit BIN selection and original-image matching, BLE resumption and recovery checks. Uniform database and action toolbar buttons and alignment.
R1.20	23 October 2026	Emergency input checks before starts, reset state and execution-row refresh, live instruction-test status, configuration/debug menu names, compact toolbar and display-area fitting.
R1.21	24 October 2026	All-task homing guards; confirmed reset and initialization results; initialization in More only; retired physical initialization mappings; software-stop limits and paused-state Host handshake compatibility.