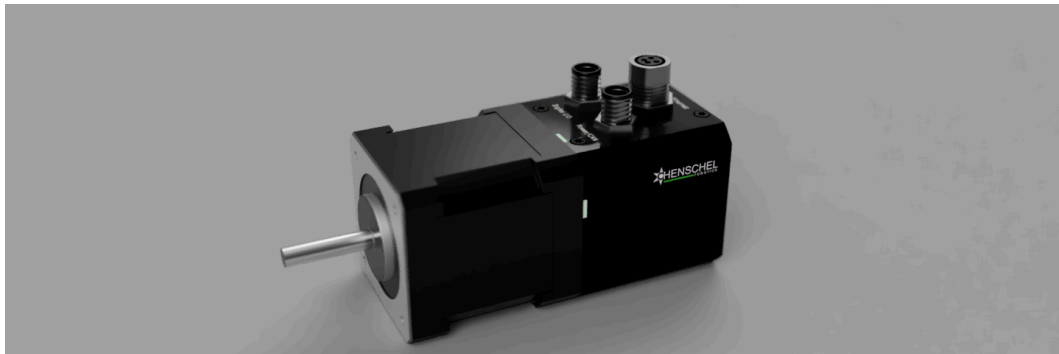




HDrive17/23-ETH-i

Servo Drive — User Manual

Compact servo drive with integrated web server, Ethernet, and CAN

Manufacturer



Henschel-Robotics GmbH
Mulchlingerstrasse 67
CH-8405 Winterthur

info@henschel-robotics.ch
www.henschel-robotics.ch

Product

The Henschel-Robotics HDrive17-ETH-i is a compact servo drive with an integrated web server. The drive can be commissioned, monitored, and controlled via Ethernet.

The direct drive is based on a high-pole-count bipolar stepper motor with field-oriented control (FOC).

- Closed loop
- High dynamics
- High efficiency
- Fast communication
- Internal CAN bus for up to 8 slave axes

Purpose of this document

This manual describes installation, commissioning, operation, communication, and service functions of the HDrive servo drive. It is intended for qualified technical personnel.

Read this manual before installing or commissioning the drive. The machine manufacturer and system integrator remain responsible for safe installation and operation of the complete machine or system, including compliance with the applicable local regulations.

Document history

Version	Date	Changes
0.0	01.09.2020	Document creation
0.1	20.04.2021	Updated drawings for Nema23-ETH-I servo drive
0.2	06.10.2021	Updated homing section regarding firmware 2.49 and above
1.0	28.10.2021	Final review
1.1	10.04.2022	Corrected reset pins for HDrive17-ETH and HDrive17-ETH-I
1.2	05.06.2022	Added new binary CAN ticket with slave torque and speed
1.3	07.06.2023	Corrected the advanced CAN ticket description
1.4	29.04.2026	Reformatted manual; updated CAN slave object access keys
1.5	12.05.2026	Updated OD ranges, CAN slave firmware update, control mode 135, and CAN limits
1.6	19.05.2026	Ethernet TCP/UDP routing, objRead responses, OD scope note; HDriveConfigFile sub-key tables for m2, m4, m5, m6; corrected timestamp units on outgoing tickets (μ s vs s)

Contents

1. Overview	5
1.1. Product overview	5
1.2. Intended use	5
1.3. Variants and order code	5
1.4. Certificates	5
2. Commissioning	7
2.1. Mechanical installation	7
2.2. Electrical installation	7
2.3. Initial configuration	7
2.4. Pin out	8
2.4.1. HDrive17-ETH-i	8
2.5. Host network configuration	9
2.6. Communication interfaces	10
2.6.1. Ethernet	10
2.6.2. Digital I/O	10
2.6.3. CAN	11
2.6.4. CAN bus topology	11
3. Control principles	12
3.1. Control concept	12
3.2. Motor calibration	13
4. Ethernet communication	14
4.1. Ethernet services and connection limits	14
4.2. Commands to the drive	14
4.2.1. Control ticket	14
4.3. CAN configuration ticket	17
4.3.1. CAN control ticket	19
4.3.2. CAN advanced configuration ticket	19
4.3.3. System ticket	20
4.3.4. Read and write objects	21
4.3.5. Read slave objects over CAN	21
4.3.6. Write slave objects over CAN	22
4.4. Messages from the drive to the host	22
4.4.1. HDrive ticket	22
4.4.2. Binary ticket – short	23
4.4.3. Binary ticket	24
4.4.4. Binary CAN ticket	24
4.4.5. Binary CAN ticket – full	25
4.4.6. CANTicket (XML)	27
4.5. Errors and fault handling	27
4.5.1. Reading the current error	27
4.5.2. Voltage faults and power-stage faults	28
4.5.3. Fault recovery	28
4.5.4. Error list	29
5. Firmware upgrade	30
5.1. Drive firmware	30
5.2. CAN slave firmware upgrade	30

5.3. Web GUI update	30
5.4. Factory reset	31
6. Object dictionary	32
6.1. Main keys and valid sub-key ranges	32
6.2. Persistent configuration files	32
6.3. HDriveConfigFile sub-keys	32
6.3.1. Main key 2 – control values (m2/s0 ... m2/s26)	33
6.3.2. Main key 5 – motor-system values (m5/s0 ... m5/s45)	34
6.3.3. Main key 4 – communication values (m4/s0 ... m4/s39)	36
6.3.4. Main key 6 – motor values (m6/s0 ... m6/s7)	37
6.4. Frequently used keys	37
7. CAN slave object access	38
7.1. Preconditions on the master	38
7.2. Request format	38
7.2.1. Read	38
7.2.2. Write	38
7.2.3. Slot mapping	38
7.3. Response (current firmware)	38
7.4. Poll helper slvobj=s	39
7.5. Read flow	39
7.5.1. Current firmware	39
7.5.2. Legacy firmware (QUEUED response)	39
7.6. Write flow	39
7.6.1. Current firmware	39
7.6.2. Legacy firmware	39
7.7. Common issues	39
8. Technical data	40
8.1. Factory default values	40
8.2. Electrical data	40
8.2.1. Supply	40
8.2.2. Electrical absolute maximum values	40
8.2.3. Motor characteristics	40
8.2.4. Timing characteristics	41
8.2.5. Environmental conditions	41
8.2.6. LED indications	41
8.3. Digital I/O	41
8.3.1. 24 V mode	41
8.3.2. 5 V mode	41
8.3.3. Digital input	41
8.4. Digital output	41
8.5. Torque curve, weight, and dimensions	42
8.5.1. HDrive17-ETH-i dimensions	42
8.5.2. HDrive23-ETH-i dimensions	42

1. Overview

1.1. Product overview

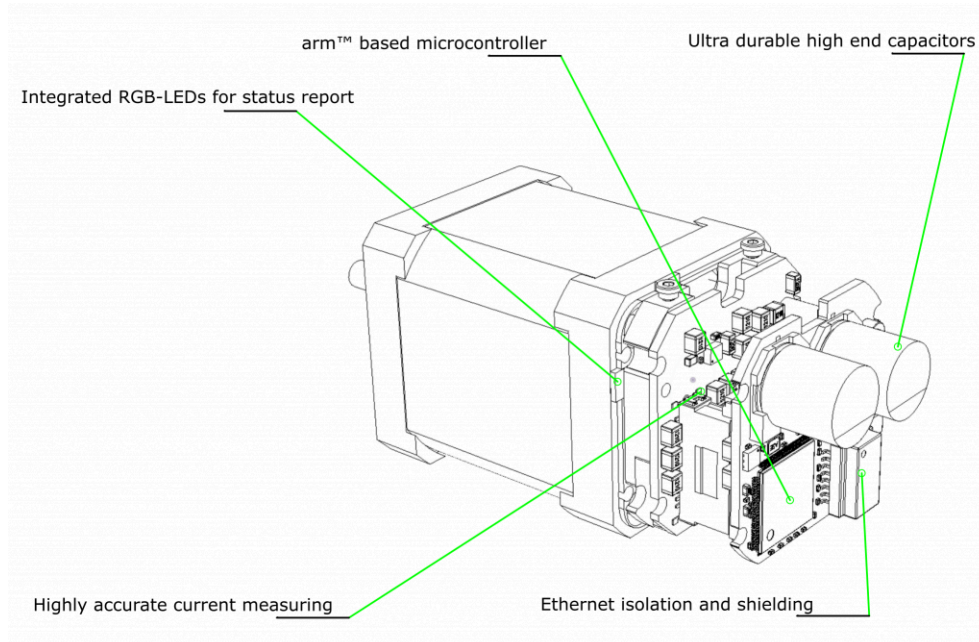


Figure 1: The HDrive17-ETH-i

The Henschel-Robotics HDrive is an integrated servo drive based on a multi-pole stepper motor. The unit integrates the power stage, communication interfaces, and position sensor electronics in a compact assembly.

The control electronics are based on an ARM™ microcontroller and provide position, speed, and torque control functions as well as trajectory generation for supported operating modes.

1.2. Intended use

The HDrive is intended for integration into machinery and automation equipment as an electrically actuated servo axis. Use the drive only within the electrical, thermal, mechanical, and environmental limits stated in this manual.

The drive is not a standalone safety device. If the application requires safe stopping, torque limiting, interlocking, or risk reduction functions, these functions must be implemented at system level.

1.3. Variants and order code

The HDrive is available with or without a gearbox. The product designation is:

HDriveAA-BBB-i-CCCC-P

Symbol	Description
AA	17 or 23 — flange size of the motor
BBB	ETH for Ethernet, ETC for EtherCAT
CCCC	GP15 or GP27 for planetary gearbox
P	P for precision gearbox, empty for standard gearbox

Table 1: Motor variants

1.4. Certificates

The HDrive successfully passed the following EMC standards on May 30, 2017:

Standard	Description	Result
EN 61000-6-3:2011 / IEC 61000-6-3:2011 (ed2.1)	EMC, Part 6-3: Generic standards — emission standard for residential, commercial and light-industrial environments	Pass
EN 61000-6-2:2016 / IEC 61000-6-2:2016	EMC, Part 6-2: Generic standards — immunity for industrial environments	Pass
EN 61326-1:2013 / IEC 61326-1:2012 (ed2.0)	Electrical equipment for measurement, control and laboratory use — EMC requirements, Part 1: General	Pass

Table 2: EMI certificates

2. Commissioning

This chapter describes the mechanical installation, electrical connection, and basic setup required to commission an HDrive servo drive.

2.1. Mechanical installation

Mount the drive only by means of the four front-side M3 threads (depth 4.5 mm) located near the shaft. Use the 2 mm deep centering ring with tolerance $22 \begin{smallmatrix} 0 \\ -0.05 \end{smallmatrix} mm$ to maintain concentricity between the application and the drive.

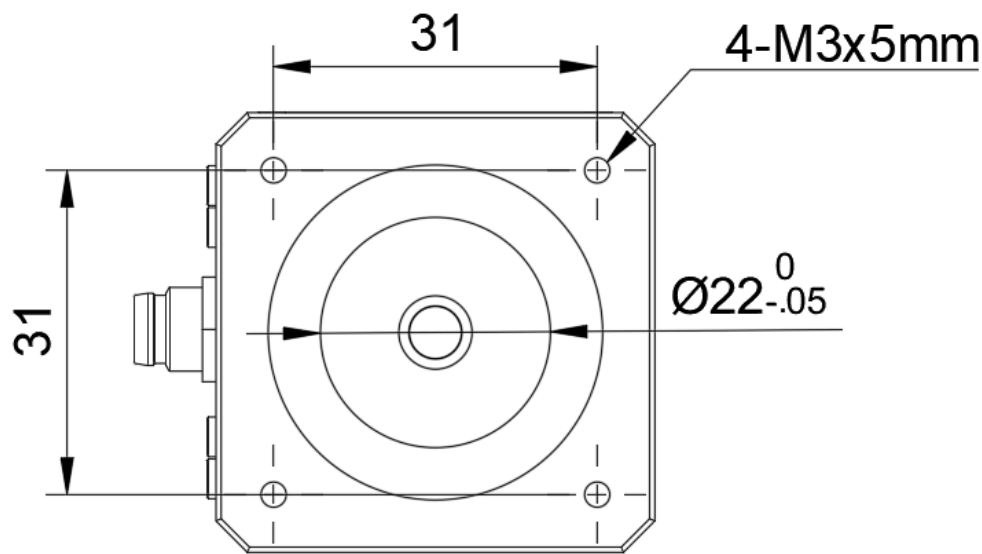


Figure 2: Motor flange

2.2. Electrical installation

The permitted supply voltage of the HDrive is 12 V to 24 V. Install a capacitor of at least 4.7 mF / 63 V close to the drive to absorb regenerative energy during braking.

Warning: Supply voltages above 28 V can damage the drive. The supply input includes reverse-polarity protection; if polarity is reversed, the drive does not start. Correct the wiring before applying power. Do not disconnect connections while energized.

Warning: Installation and commissioning shall be performed only by qualified personnel. Observe the applicable local electrical and machinery regulations.

2.3. Initial configuration

The drive includes an integrated web server. Operating data can be monitored in real time in a web browser, and parameters such as controller settings, current limits, and motion profile values can be configured. Google™ Chrome and Apple™ Safari are recommended for best presentation.

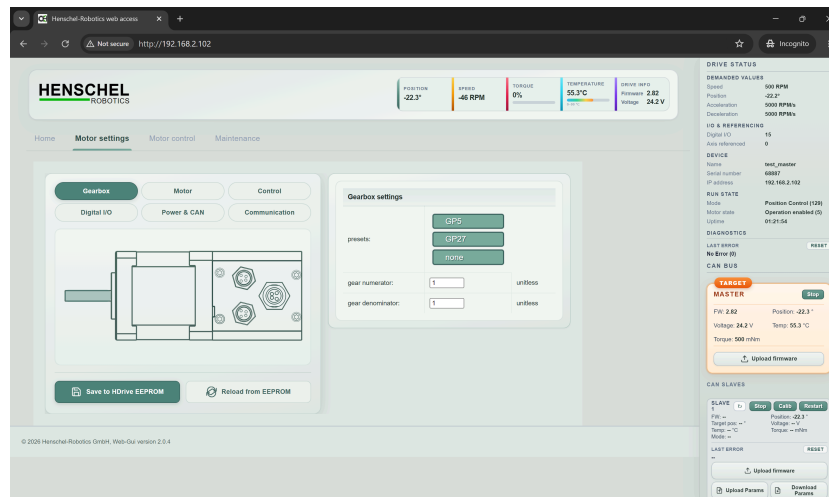


Figure 3: HDrive Web GUI

Warning: The Web GUI can command motion. Before using the control functions, make sure the axis can move safely and without obstruction.

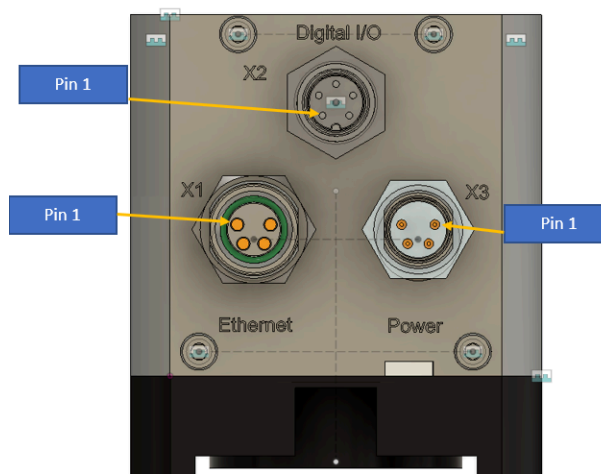
Warning: The integrated web server does not provide access protection. Network access protection must therefore be implemented at system level.

Warning: If the Web GUI is open in the “Motor Control” view, external motion commands may not take effect as expected. Confirm the active control source before commissioning the axis.

2.4. Pin out

This section lists the connector assignments for the HDrive17-ETH-i variants.

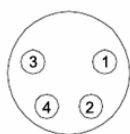
2.4.1. HDrive17-ETH-i



X3

Type: M8 / male
Coding: uncoded
Description: Power and CAN
Pin count: 4

Pin	Function
1	VCC
2	CAN_H
3	GND
4	CAN_L

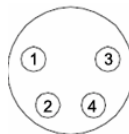


Front view, male plug (device side)

X1

Type: M8 / female
Coding: uncoded
Description: Ethernet
Pin count: 4

Pin	Function
1	ETH TX+
2	ETH RX+
3	ETH RX-
4	ETH TX-

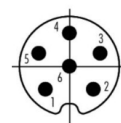


Front view, female plug (device side)

X2

Type: M8 / male
Coding: A-coded
Description: Digital I/O
Pin count: 6

Pin	Function
1	VCC (100 mA)
2	Digital I/O-A
3	GND
4	Digital I/O-B
5	Digital I/O-C
6	Digital I/O-D

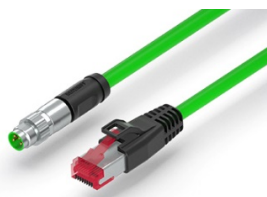


Front view, female plug (device side)

Cable, e.g. Beckhoff
ZK2020-3200-0xxx



Cable, e.g. Beckhoff
ZK1090-3191-0xxx



Cable, e.g. Phoenix Contact Art. Nr.
1522396



Table 3: HDrive17-ETH-i connector overview

2.5. Host network configuration

To configure the servo drive, connect the HDrive to an Ethernet switch or directly to a PC using an Ethernet cable. To establish communication, the host PC must be configured for the same IP subnet as the drive.

Note: A new or factory-reset drive uses IP address 192.168.1.102. In that case, configure the host PC with an address in the range 192.168.1.1 to 192.168.1.254, excluding 192.168.1.102.

An example of IPv4 configuration on Microsoft™ Windows is shown below:

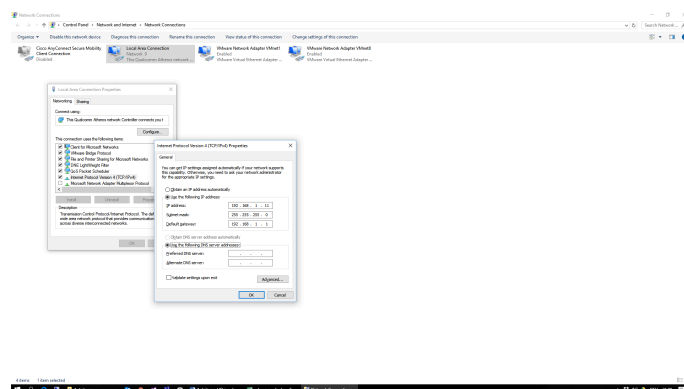


Figure 4: Changing the IP address of a network adapter on Microsoft™ Windows

On Windows 11, open Settings, select “Network & internet”, and edit the IPv4 configuration. Select “Manual” mode and enter a suitable IP address, for example 192.168.1.99.

After the host IP address has been configured, enter the IP address of the drive in a web browser. The drive serves its Web GUI on port 80 (HTTP).

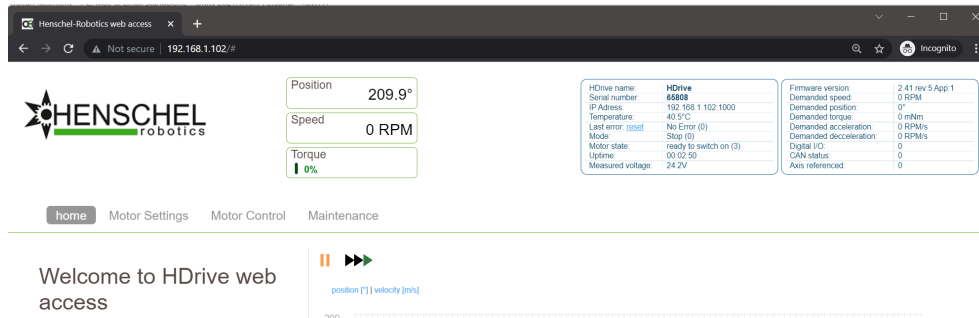


Figure 5: Insert the IP address into the address bar of your browser

After the first connection, the drive IP address can be adapted to the target network.

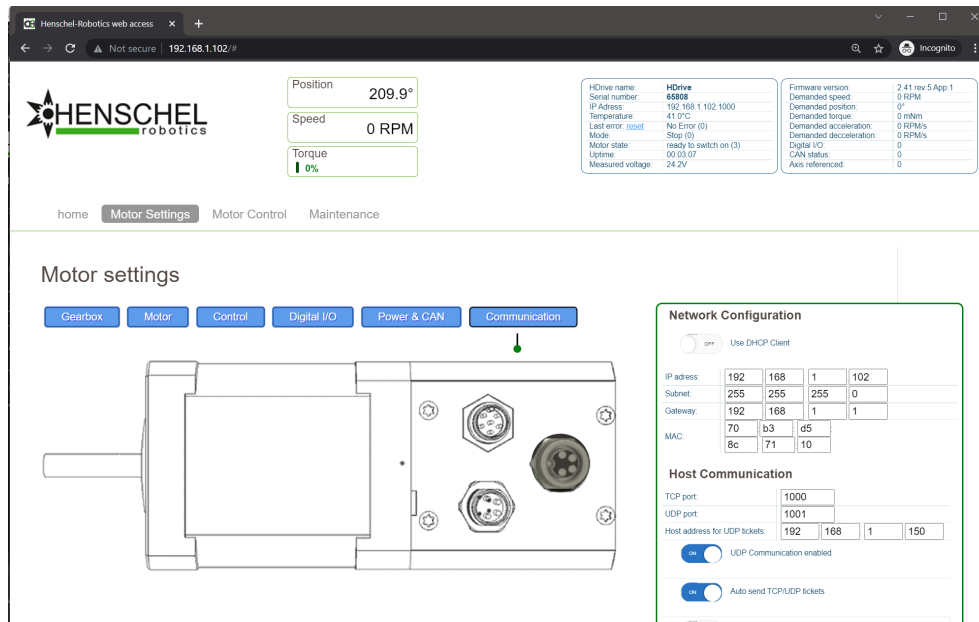


Figure 6: Changing the IP address of the drive

Note: After changing the drive IP address, restore the host network configuration as required for the plant network.

2.6. Communication interfaces

Each drive supports several communication interfaces. These interfaces are configured in the Web GUI.

2.6.1. Ethernet

Ethernet is the main interface to the drive. It is used for configuration and cannot be turned off.

2.6.2. Digital I/O

The digital I/O interface supports 5 V or 24 V logic. Without Ethernet, the drive can be controlled in Step/Dir, torque (PWM), or position (PWM) mode.

2.6.3. CAN

Every HDrive can be configured either as a CAN master or as a CAN slave. In CAN master mode, one drive can control up to 8 additional HDrive slave axes. This reduces host wiring and improves synchronization behavior in multi-axis applications.

Warning: The communication interfaces are not fail-safe. The higher-level controller must supervise communication and bring the machine to a safe state if communication is lost.

2.6.4. CAN bus topology

In the example below, two HDrive servo drives are controlled via Ethernet. One of these drives is configured as a CAN master and generates the CAN bus. Two additional drives configured as CAN slaves are connected to the 2-wire CAN bus. The higher-level controller can address all axes without being connected to the CAN bus directly. CAN slave axes are controlled through the CAN master by means of master-side slave commands. Each CAN slave requires a unique CAN ID from 1 to 8.

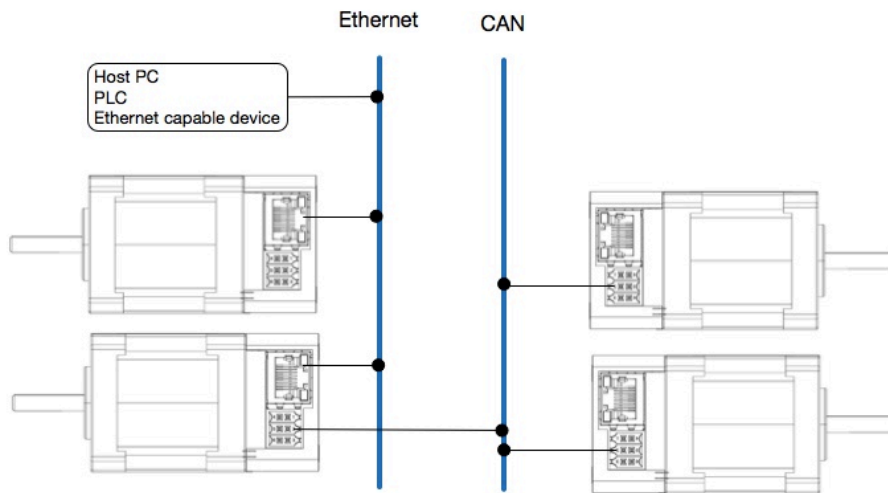


Figure 7: Operating several HDrive servomotors in a network

3. Control principles

The drive contains several discrete-time control loops. Torque, speed, and position control can be adjusted separately. The following section summarizes the control architecture relevant to commissioning and application tuning.

Warning: The control parameters must be matched to the application. Incorrect tuning can cause unstable motion, excessive heating, or damage to the drive and the connected mechanics.

3.1. Control concept

The stepper motor is controlled by field-oriented control. Torque-producing and reactive current components are regulated separately. The control structure is shown below:

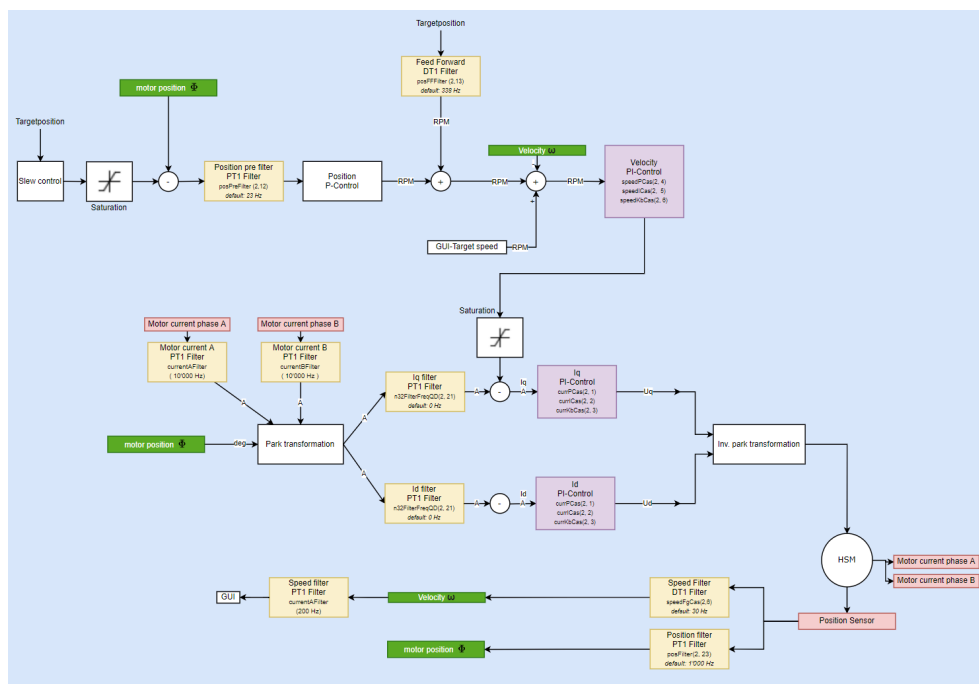


Figure 8: Control architecture

The drive must be tuned for each application. Dynamic behavior depends on the inertia and compliance of the driven load. The position controller bandwidth can be set in the Web GUI; the associated controller values are then calculated automatically. The current, speed, and position control loops can also be adjusted individually.

Use the lowest bandwidth and excitation frequencies consistent with the application in order to minimize noise and losses. Several predefined values are available as starting points during tuning.

The drive supports three principal closed-loop control categories:

Current control — The commanded current is maintained by the current controller. The current loop operates at approximately 20 kHz. In this mode, the speed and position controllers are inactive. Use this mode when constant shaft torque is required.

Speed control — The drive regulates shaft speed to the commanded value, including under changing load.

Position control — The drive regulates the shaft to a commanded target position while observing the configured torque, acceleration, and speed limits.

3.2. Motor calibration

The drive is calibrated at the factory before delivery. Recalibration may be required if operating behavior indicates a sensor or commutation mismatch, for example increased vibration.

Warning: Remove the external load from the shaft before starting calibration.

Calibration can be started from the Web GUI. Apply a 24 V supply and ensure that the shaft can rotate freely without external load.

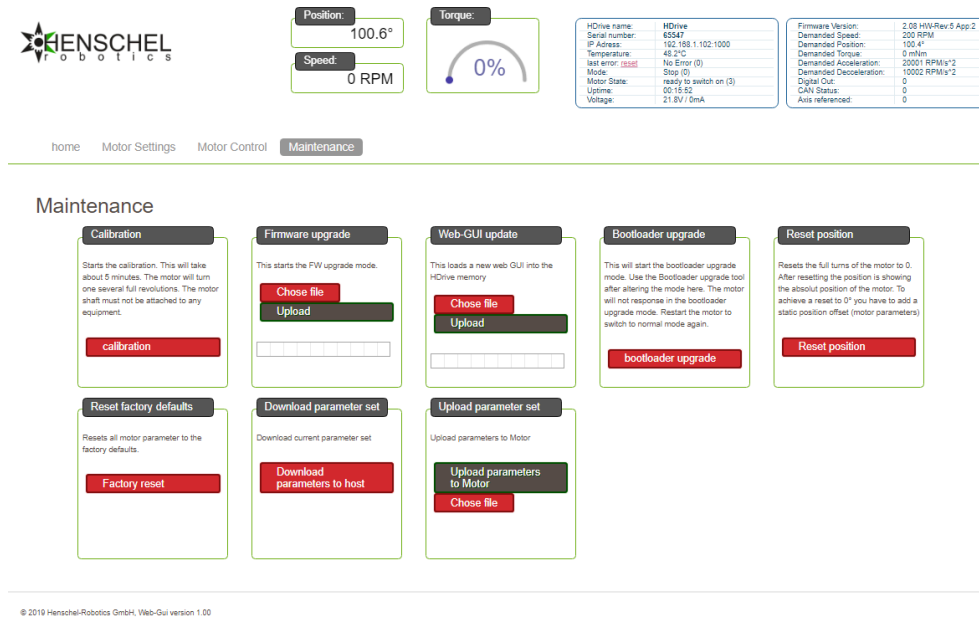


Figure 9: Maintenance mode

The “Calibration” button is located on the “Maintenance” tab. After the command is issued, the drive performs several reference revolutions and calculates the required calibration data. The procedure takes approximately 5 minutes. Do not operate the Web GUI during this time, as the calibration process may be interrupted. After successful completion, the drive restarts automatically.

4. Ethernet communication

In addition to the HTTP-based Web GUI, the HDrive provides TCP and UDP communication channels. The UDP channel is transmit-only. Motion and configuration commands are always sent via TCP. Outgoing status telegrams can be configured for TCP or UDP in the Web GUI.

For the binary interface, UDP is generally recommended for data reception because the payloads remain compact and are not fragmented in the intended use case.

4.1. Ethernet services and connection limits

The firmware network stack allows up to three simultaneous TCP connections to one drive (`Ip.socketMaxConnections` = 3). Typical parallel use is:

1. HTTP on port 80 (Web GUI and `getData.cgi` / `writeTicket.cgi`),
2. the XML command channel on the configured TCP port (`m4/s16`, factory default 1000), and
3. the optional binary motor-data stream on TCP port + 1 (factory default 1001, same 33×int32 payload as `getData.cgi?bin`).

The command port handles one active client session at a time. A second TCP client to the same command port is rejected (connection closed) so the network stack stays within `Ip.socketMaxConnections`; use a single session and multiplex tickets on it. The motor stream port likewise serves one connected client at a time.

Outgoing cyclic status telegrams are controlled in the Web GUI on the **Communication** tab (Object Dictionary main key 4):

1. **Auto send TCP/UDP tickets** (`m4/s34`) — enables periodic transmission at the rate set by the prescaler (`m4/s18`).
2. **UDP Communication enabled** (`m4/s19`, `sendUDP`) — selects UDP instead of TCP for those periodic telegrams.
3. **Communication Host to HDrive Ticket** (`m4/s22`, `TXTicket`) — selects the telegram format (XML, binary, CAN variants, and so on).

The UDP destination is configured with **Host address for UDP tickets** (`m4/s4–m4/s7`). If all four octets are 0 (0.0.0.0, factory default after reset), the drive sends UDP telegrams to the IPv4 address of the host that currently has an open TCP command connection. The target address is updated when a client connects and on each UDP transmission while that session is active. With 0.0.0.0 and no TCP command client connected, no UDP destination is available until a client connects.

4.2. Commands to the drive

A target position, speed, or current can be specified with the control telegram. For supported operating modes, the internal path planner calculates the trajectory based on the configured acceleration and speed values.

4.2.1. Control ticket

Example: move to position 100.0° at 200 rpm with maximum torque 200 mNm and acceleration/deceleration of 1000 rpm/s:

```
<control pos="1000" speed="200" torque="200" mode="129" acc="1000" decc="1000" />
```

Warning: Observe the expected attribute order. The parser is sensitive to spaces within or immediately before attribute values.

4.2.1.1. Parameters

XML tag	Range	Unit	Function
pos	$\pm 2^{31}$	1/10 degree	Target position
speed	± 2000	RPM	Maximum speed of the internal path planner
torque	± 600	mNm	Maximum torque
mode	see table below	—	Operation mode. The current GUI exposes the supported user-facing subset of implemented mode IDs.
acc	1 to 50'000	RPM/s	Acceleration of the internal path planner
decc	1 to 50'000	RPM/s	Deceleration of the internal path planner

Table 4: Control ticket parameters

4.2.1.2. Operational modes

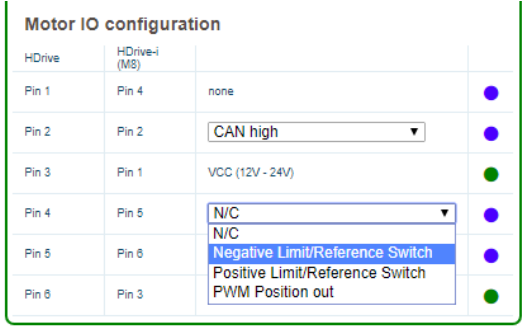
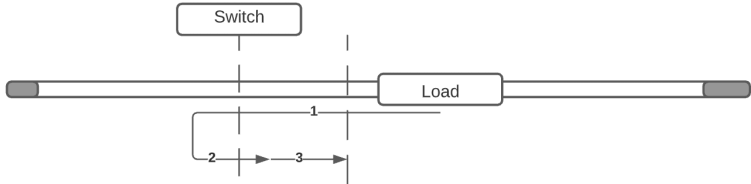
Mode ID	Function	Description
-1	Error	The drive has a fault. The error status can be read in the Web GUI or via object 3.4.
0	Stop / motor off	De-energizes the motor; the shaft can be rotated freely.
8	Stepper mode	Stepper-motor mode for extremely slow and even movements.
9	Motor calibration	The shaft must rotate without resistance. The drive performs several revolutions and stores the calibration data in EEPROM.
15	Limit switch left, advanced	The limit switch can be configured as low-active or high-active in the Web GUI.  The speed and maximum torque are taken from the control ticket (speed and torque). Limit switch sequence:  The motor moves left until the switch is active for more than 1 ms (internal debounce). It then reverses at 20% of the commanded speed until the switch is released, and continues until the next index pulse of its absolute encoder. Operating mode then changes to mode 20 ("Calibration done"). The position accuracy is around $\pm 1.5^\circ$ after successful homing; switch to position control to reach any target.
16	Limit switch right, advanced	Same as 15, but reversed direction.
17	Limit switch left, simple	Same as 15, but without moving to the encoder index pulse.
18	Limit switch right, simple	Same as 17, but reversed direction.
20	Limit switch finished	Activated by states 15-18 once the limit-switch sequence is done.

Table 5: Control ticket operational modes – basic and homing modes

Mode ID	Function	Description
101	Set zero	Resets the position to 0.
128	Motor current control	The active motor current is regulated to a fixed value, producing constant torque.
129	Motor position control	Position controller with active path planner.
130	Motor speed control	Speed controller with internal path planner.
132	Motor speed control NPP ¹	Speed controller with deactivated path planner.
133	Motor position control NPP	Position controller with deactivated path planner.
134	Motor position control, stepper mode	The drive behaves like a closed-loop stepper. Useful for low-dynamic, very smooth applications (e.g. tracking a telescope). Once the position is reached, holding current is reduced to 20% of the nominal current.
135	Motor position control, oscillating	Sinusoidal position control around an offset. In this mode, pos = amplitude [1/10°], speed = frequency [0.01 Hz], torque = torque limit [mNm], acc = offset [1/10°], and decc = phase [1/10°].
153	Referencing mode with torque limit	The drive runs in speed mode with the configured acceleration and speed. When the torque limit is reached, the system sets the reference bit to 1 (mechanical limit). The drive keeps pushing against the limit until another command arrives. Useful when the limit acts against gravity.
154	Referencing mode with torque limit and auto-off	Same as 153, but the motor stops as soon as the limit is reached.

Table 6: Control ticket operational modes — closed-loop and reference modes

4.3. CAN configuration ticket

Note: To use this ticket, first assign the digital I/O channels to “CAN-High” and “CAN-Low” in the Web GUI. Then select “Can Ticket” on the “Communication” tab under “Communication Host to HDrive Ticket”.

The CAN master axis and all configured CAN slave axes can be configured directly with this ticket. The operating mode and maximum torque of each drive can be commanded, for example to switch multiple slave axes to a homing mode.

¹NPP = no path planner

Description	Range	Unit	Function	XML tag
Master torque	±600	mNm	Target torque of the CAN master axis	m
Master mode			Operating mode of the CAN master axis	mm
Slave 1 torque	±600	mNm	Target torque of slave axis 1	Sl1t
Slave 2 torque	±600	mNm	Target torque of slave axis 2	Sl2t
Slave 3 torque	±600	mNm	Target torque of slave axis 3	Sl3t
Slave 4 torque	±600	mNm	Target torque of slave axis 4	Sl4t
Slave 5 torque	±600	mNm	Target torque of slave axis 5	Sl5t
Slave 6 torque	±600	mNm	Target torque of slave axis 6	Sl6t
Slave 7 torque	±600	mNm	Target torque of slave axis 7	Sl7t
Slave 8 torque	±600	mNm	Target torque of slave axis 8	Sl8t
Slave 1 mode			Operating mode of slave axis 1	Sl1m
Slave 2 mode			Operating mode of slave axis 2	Sl2m
Slave 3 mode			Operating mode of slave axis 3	Sl3m
Slave 4 mode			Operating mode of slave axis 4	Sl4m
Slave 5 mode			Operating mode of slave axis 5	Sl5m
Slave 6 mode			Operating mode of slave axis 6	Sl6m
Slave 7 mode			Operating mode of slave axis 7	Sl7m
Slave 8 mode			Operating mode of slave axis 8	Sl8m
Slave 1 special function			Special function of slave axis 1	Sl1f
Slave 2 special function			Special function of slave axis 2	Sl2f
Slave 3 special function			Special function of slave axis 3	Sl3f
Slave 4 special function			Special function of slave axis 4	Sl4f
Slave 5 special function			Special function of slave axis 5	Sl5f
Slave 6 special function			Special function of slave axis 6	Sl6f
Slave 7 special function			Special function of slave axis 7	Sl7f
Slave 8 special function			Special function of slave axis 8	Sl8f

Table 7: CAN configuration ticket fields

CAN special functions:

XML tag	Range	Unit	Function
Sf1-8	0, 4 or 5	—	0 = no function, 4 = save data to EEPROM, 5 = reset last error

Table 8: CAN special functions

The following example sets the master and the first three slave axes to mode 133 (position control without path planner). The maximum torque is set to 200 mNm for the master and 100 mNm for the slaves. The special-function fields are set to 0 (no function):

```
<canConf m="200" mm="133"
  sl1t="100" sl2t="100" sl3t="100" sl4t="0" sl5t="0" sl6t="0" sl7t="0" sl8t="0"
  sl1m="133" sl2m="133" sl3m="133" sl4m="0" sl5m="0" sl6m="0" sl7m="0" sl8m="0"
  sl1f="0" sl2f="0" sl3f="0" sl4f="0" sl5f="0" sl6f="0" sl7f="0" sl8f="0" />
```

Warning: Do not change the expected attribute order. All fields defined by the ticket shall be transmitted in the expected sequence, even if a given slot is unused.

4.3.1. CAN control ticket

With the control ticket, target positions for all configured axes are sent. These target positions are executed according to the operating mode previously configured with a canConf command.

Description	Range	Unit	Function	XML tag
Position	$\pm 2^{31}$	1/10 degree	Target position of master	pos
Slave pos 1	$\pm 2^{31}$	1/10 degree	Target position of slave axis 1	sl1
Slave pos 2	$\pm 2^{31}$	1/10 degree	Target position of slave axis 2	sl2
Slave pos 3	$\pm 2^{31}$	1/10 degree	Target position of slave axis 3	sl3
Slave pos 4	$\pm 2^{31}$	1/10 degree	Target position of slave axis 4	sl4
Slave pos 5	$\pm 2^{31}$	1/10 degree	Target position of slave axis 5	sl5
Slave pos 6	$\pm 2^{31}$	1/10 degree	Target position of slave axis 6	sl6
Slave pos 7	$\pm 2^{31}$	1/10 degree	Target position of slave axis 7	sl7
Slave pos 8	$\pm 2^{31}$	1/10 degree	Target position of slave axis 8	sl8

Table 9: CAN control ticket fields

```
<canPos pos="1000" sl1="200" sl2="20" sl3="500" sl4="100" sl5="100" sl6="100" sl7="100" sl8="100" />
```

Typical start sequence:

1. Set the operating mode to position control without path planner (133), set the master torque to 200 mNm, and set the slave torque to 100 mNm:

```
<canConf m="200" mm="133"
  sl1t="100" sl2t="100" sl3t="100" sl4t="100" sl5t="100" sl6t="0" sl7t="0" sl8t="0"
  sl1m="133" sl2m="133" sl3m="133" sl4m="133" sl5m="133" sl6m="0" sl7m="0" sl8m="0" />
```

2. Command the target positions:

```
<canPos pos="1000" sl1="200" sl2="20" sl3="500" sl4="1000" sl5="1000" sl6="1000" sl7="1000"
sl8="1000" />
```

Warning: Do not change the expected attribute order. Every configured axis position in the telegram must be present, even if a given slot is unused.

4.3.2. CAN advanced configuration ticket

This ticket sets the profile speed, acceleration, and deceleration values. It is required for mode 129 (position control), mode 130 (speed control), and for homing modes that use configured travel speed, for example mode 15 ("limit switch left, advanced").

Description	Range	Unit	Function	XML tag
Master speed	uint16	RPM	Profile speed of the master axis	ms
Master acceleration	uint16	RPM/s	Profile acceleration of the master axis	ma
Master deceleration	uint16	RPM/s	Profile deceleration of the master axis	md
Slave 1 speed	uint16	RPM	Profile speed of slave axis 1	s1s
Slave 1 acceleration	uint16	RPM/s	Profile acceleration of slave axis 1	s1a
Slave 1 deceleration	uint16	RPM/s	Profile deceleration of slave axis 1	s1d
Slave 2 speed	uint16	RPM	Profile speed of slave axis 2	s2s
Slave 2 acceleration	uint16	RPM/s	Profile acceleration of slave axis 2	s2a
Slave 2 deceleration	uint16	RPM/s	Profile deceleration of slave axis 2	s2d
Slave 3 speed	uint16	RPM	Profile speed of slave axis 3	s3s
Slave 3 acceleration	uint16	RPM/s	Profile acceleration of slave axis 3	s3a
Slave 3 deceleration	uint16	RPM/s	Profile deceleration of slave axis 3	s3d
Slave 4 speed	uint16	RPM	Profile speed of slave axis 4	s4s
Slave 4 acceleration	uint16	RPM/s	Profile acceleration of slave axis 4	s4a
Slave 4 deceleration	uint16	RPM/s	Profile deceleration of slave axis 4	s4d
Slave 5 speed	uint16	RPM	Profile speed of slave axis 5	s5s
Slave 5 acceleration	uint16	RPM/s	Profile acceleration of slave axis 5	s5a
Slave 5 deceleration	uint16	RPM/s	Profile deceleration of slave axis 5	s5d
Slave 6 speed	uint16	RPM	Profile speed of slave axis 6	s6s
Slave 6 acceleration	uint16	RPM/s	Profile acceleration of slave axis 6	s6a
Slave 6 deceleration	uint16	RPM/s	Profile deceleration of slave axis 6	s6d
Slave 7 speed	uint16	RPM	Profile speed of slave axis 7	s7s
Slave 7 acceleration	uint16	RPM/s	Profile acceleration of slave axis 7	s7a
Slave 7 deceleration	uint16	RPM/s	Profile deceleration of slave axis 7	s7d
Slave 8 speed	uint16	RPM	Profile speed of slave axis 8	s8s
Slave 8 acceleration	uint16	RPM/s	Profile acceleration of slave axis 8	s8a
Slave 8 deceleration	uint16	RPM/s	Profile deceleration of slave axis 8	s8d

Table 10: CAN advanced configuration ticket fields

Example: the command sets the master speed to 500 rpm, acceleration to 200 rpm/s, and deceleration to 2000 rpm/s. It also sets the speed to 77 rpm and both acceleration and deceleration to 1000 rpm/s for the first two slave axes:

```
<canC2 ms="500" ma="200" md="2000"
  s1s="77" s1a="1000" s1d="1000"
  s2s="77" s2a="1000" s2d="1000"
  s3s="0" s3a="0" s3d="0"
  s4s="0" s4a="0" s4d="0"
  s5s="0" s5a="0" s5d="0"
  s6s="0" s6a="0" s6d="0"
  s7s="0" s7a="0" s7d="0"
  s8s="0" s8a="0" s8d="0" />
```

Warning: Do not change the expected attribute order. Every configured axis profile set in the telegram must be present, even if a given slot is unused.

4.3.3. System ticket

The system telegram is used to trigger predefined service functions and state transitions.

XML tag	Range	Unit	Function
mode	0 - 7	—	0 = firmware upgrade 1 = bootloader upgrade 2 = position reset (not permanent) 3 = factory reset 4 = save data to EEPROM 5 = reset last error 6 = restart motor 7 = save MAC

Table 11: System-ticket fields

The following example resets the current position value to 0:

```
<system mode="2" a="1" b="2" c="3" />
```

After a restart, the rotor position again corresponds to the absolute encoder position plus the configured offset.

The XML attributes a, b, and c must be 1, 2, and 3 respectively. They currently serve only as fixed confirmation values.

4.3.4. Read and write objects

XML tag	Range	Unit	Function
m	0 - 9	—	Main key
s	0 - 74	—	Sub-key. The valid upper limit depends on m.
v	$\pm 2^{31}$	—	Value (write only)

Table 12: Object read/write fields

Example: read object 3.4 (lastError):

```
<objRead m="3" s="4" />
```

Example: write object 3.4 (lastError):

```
<objWrite m="3" s="4" v="0" />
```

Successful read response:

```
<r a="3" b="4" v="0" />
```

Error response (invalid main/sub key or malformed ticket):

```
<r e="3.4" />
```

A successful objWrite produces no TCP reply. If the ticket is malformed or the key is out of range, the drive sets m3/s4 (lastError) to the wrong-format error code and sends no response on the command channel.

4.3.5. Read slave objects over CAN

The CAN master can read one Object Dictionary entry from a configured CAN slave via the TCP command channel:

XML tag	Range	Unit	Function
sl	0 - 7	—	Slave slot (0 = slaveID1, 7 = slaveID8)
m	valid OD key	—	Main key on the addressed slave
s	valid OD sub-key	—	Sub-key within m

Table 13: CAN slave object read fields

Example: read object 3.15 from slave slot 0:

```
<objReadCAN sl="0" m="3" s="15" />
```

Successful response:

```
<r sl="0" a="3" b="15" v="281" />
```

Error response:

```
<r e="0.3.15" />
```

This ticket is available only when CAN is enabled, the drive operates in CAN master mode, and the selected slot has a configured slave ID. The mailbox is single-shot, so only one CAN slave object transaction can be active at any time. If the read fails (invalid slot or OD key, missing slave, bus-off, timeout, or slave-side error), the ticket returns the error form above; the detailed cause is stored in object 3.4 (lastError) of the master.

4.3.6. Write slave objects over CAN

The CAN master can write one Object Dictionary entry on a configured CAN slave via the TCP command channel. Fields are the same as for read, plus v:

```
<objWriteCAN sl="0" m="3" s="15" v="281" />
```

Successful response:

```
<w sl="0" a="3" b="15" v="281" />
```

Error response:

```
<w e="0.3.15" />
```

The same mailbox, master-mode, and single-shot rules as objReadCAN apply.

4.4. Messages from the drive to the host

As soon as a TCP connection is established, the drive can transmit status data either as XML or in binary form, depending on the selected telegram type in the Web GUI.

Note: UDP mode can reduce transport overhead for cyclic status data.

Note: **System time** fields use the CPU timestamp in **microseconds** since boot (getTime_us()): binary word 0, Time in <HDrive ...>, and u32Time in <CANTicket ...>. The full 33-word **binary ticket** also includes **uptime** in **seconds** at word 21 (actualSystemTime, Object Dictionary 0/33).

4.4.1. HDrive ticket

```
<HDrive Position="300" Speed="0" Torque="0" Time="000216" />
```

XML tag	Range	Unit	Function
Position	10 digits	1/10 degree	Current position of the drive
Speed	8 digits	RPM	Current speed of the drive
Torque	8 digits	mA	Active current, directly related to torque (torque constant)
Time	10 digits	µs	System time in microseconds since boot

Table 14: HDrive ticket fields

This ticket always has a length of 82 bytes.

4.4.2. Binary ticket — short

This telegram is transmitted in binary form in order to reduce parsing effort on the host.

Word No.	Description	Type
0	Time [µs]	int32
1	Position	int32
2	Speed	int32
3	Torque	int32

Table 15: Binary ticket — short

This telegram has a length of 16 bytes.

Note: In binary mode, both network overhead and host-side parsing effort are reduced.

4.4.3. Binary ticket

Word No.	Description	Type
0	Time [μ s]	int32
1	Position	int32
2	Speed	int32
3	Current A [mA]	int32
4	Current B [mA]	int32
5	Calibration value [inc]	int32
6	Fid [mA]	int32
7	Fiq [mA]	int32
8	Reserved (0)	int32
9	Temperature [1/10 °C]	int32
10	Motor mode	int32
11	Motor voltage [mV]	int32
12	Demanded speed	int32
13	Demanded position	int32
14	Demanded torque	int32
15	Demanded acceleration	int32
16	Demanded deceleration	int32
17	GPIO	int32
18	Actual motor state	int32
19	Software version	int32
20	Current (RMS) [mA]	int32
21	Uptime [s]	int32
22	Homing completed	int32
23	Slave 1 position	int32
24	Slave 2 position	int32
25	Slave 3 position	int32
26	Slave 4 position	int32
27	Slave 5 position	int32
28	Slave 6 position	int32
29	Slave 7 position	int32
30	Slave 8 position	int32

Table 16: Binary ticket

This telegram has a total length of 124 bytes.

4.4.4. Binary CAN ticket

This telegram can be generated only by a CAN master and contains the position data of the connected CAN slave axes.

Word No.	Description	Type
0	Time [μ s]	int32
1	Master position	int32
2	Slave 1 position	int32
3	Slave 2 position	int32
4	Slave 3 position	int32
5	Slave 4 position	int32
6	Slave 5 position	int32
7	Slave 6 position	int32
8	Slave 7 position	int32
9	Slave 8 position	int32
10	Master mode	int32
11	Slave 1 mode	int32
12	Slave 2 mode	int32
13	Slave 3 mode	int32
14	Slave 4 mode	int32
15	Slave 5 mode	int32
16	Slave 6 mode	int32
17	Slave 7 mode	int32
18	Slave 8 mode	int32
19	Master state	int32
20	Slave 1 state	int32
21	Slave 2 state	int32
22	Slave 3 state	int32
23	Slave 4 state	int32
24	Slave 5 state	int32
25	Slave 6 state	int32
26	Slave 7 state	int32
27	Slave 8 state	int32
28	Path planner program index	int32

Table 17: Binary CAN ticket

This telegram has a length of 116 bytes.

4.4.5. Binary CAN ticket – full

This telegram can be generated only by a CAN master and contains position, velocity, torque, mode, and state data of the connected CAN slave axes.

Word No.	Description	Type
0	System time [μ s]	int32
1	Master position	int32
2	Slave 1 position	int32
3	Slave 2 position	int32
4	Slave 3 position	int32
5	Slave 4 position	int32
6	Slave 5 position	int32
7	Slave 6 position	int32
8	Slave 7 position	int32
9	Slave 8 position	int32
10	Master velocity	int32
11	Slave 1 velocity	int32
12	Slave 2 velocity	int32
13	Slave 3 velocity	int32
14	Slave 4 velocity	int32
15	Slave 5 velocity	int32
16	Slave 6 velocity	int32
17	Slave 7 velocity	int32
18	Slave 8 velocity	int32
19	Master torque	int32
20	Slave 1 torque	int32
21	Slave 2 torque	int32
22	Slave 3 torque	int32
23	Slave 4 torque	int32
24	Slave 5 torque	int32
25	Slave 6 torque	int32
26	Slave 7 torque	int32
27	Slave 8 torque	int32

Table 18: Binary CAN ticket — full, words 0 to 27

Word No.	Description	Type
28	Master mode	int32
29	Slave 1 mode	int32
30	Slave 2 mode	int32
31	Slave 3 mode	int32
32	Slave 4 mode	int32
33	Slave 5 mode	int32
34	Slave 6 mode	int32
35	Slave 7 mode	int32
36	Slave 8 mode	int32
37	Master state	int32
38	Slave 1 state	int32
39	Slave 2 state	int32
40	Slave 3 state	int32
41	Slave 4 state	int32
42	Slave 5 state	int32
43	Slave 6 state	int32
44	Slave 7 state	int32
45	Slave 8 state	int32
46	Path planner program index	int32
47	Reserved (0)	int32
48	Reserved (0)	int32

Table 19: Binary CAN ticket – full, words 28 to 48

This telegram has a length of 196 bytes.

4.4.6. CANTicket (XML)

When **Communication Host to HDrive Ticket** is set to the XML CAN ticket, the drive sends three consecutive XML fragments per cycle (position, mode, state). The state fragment includes a timestamp:

```
<CANTicket State="0" sS1="0" sS2="0" sS3="0" sS4="0" sS5="0" sS6="0" sS7="0" sS8="0" prgIdx="0"
u32Time="000216" />
```

XML attribute	Range	Unit	Function
prgIdx	8 digits		Path planner program index
u32Time	10 digits	µs	System time in microseconds since boot

Table 20: CANTicket state fragment – timestamp fields

4.5. Errors and fault handling

If the drive enters a fault state, first remove the cause before attempting to resume operation. The most recent latched fault code is available in the Web GUI and in Object Dictionary entry 3.4 (lastError).

4.5.1. Reading the current error

Use one of the following methods to identify the current fault:

- Check the fault or status indication in the Web GUI.
- Read Object Dictionary entry 3.4 (lastError), for example with `<objRead m="3" s="4" />`.

The stored code remains available until it is cleared. Clearing lastError does not remove the underlying cause. If the fault condition is still present, the drive will set the error again.

4.5.2. Voltage faults and power-stage faults

The current firmware distinguishes between supply-voltage limit violations and power-stage fault feedback:

- Under-voltage and Over-voltage are generated from the measured drive supply voltage. These faults indicate that the DC supply is outside the permitted operating range.
- Power-stage error is a separate fault reported by the drive stage itself. In current firmware it is not the same check as the measured supply-voltage limits above. Treat it as a drive-stage protection event and inspect the supply wiring, the local DC-link capacitor, cooling, and load conditions.

In practice, start with the power supply and installation:

- Verify the supply voltage at the drive terminals under load.
- Check the power wiring, polarity, connectors, and grounding.
- Add an external capacitor near the drive to absorb braking energy and provide energy during dynamic acceleration.
- For recurring Power-stage error or Over-temperature, check cooling, ambient temperature, and whether the application is demanding excessive current or repeated hard braking.

4.5.3. Fault recovery

After the cause has been removed, clear and restart the fault in a controlled way:

1. Bring the machine to a safe state and make sure unintended motion cannot occur.
2. Remove the cause of the fault, for example by correcting the supply, reducing braking energy, or eliminating the mechanical overload.
3. Clear the latched fault either in the Web GUI, with system command `mode = 5` (reset last error), or by writing 0 to object 3.4 (lastError).
4. If the power stage does not resume normal operation, restart the drive with system command `mode = 6` (restart motor) or cycle power after the cause has been removed.

Factory reset is not required for normal fault recovery.

4.5.4. Error list

Error No.	Message	Description
16	Over-temperature	Switches the motor off if the board temperature exceeds 85 °C.
17	Under-voltage	The measured supply voltage falls below the permitted operating range. The drive switches the motor off and saves all states.
18	Over-voltage	The measured supply voltage exceeds the permitted operating range. The drive switches the motor off. Generator operation can cause the voltage to rise sharply, especially during braking. Add an external capacitor or braking resistor if needed.
19	Over-speed	If motor speed exceeds 2000 RPM, the resulting induced voltage can be excessive. The motor brakes automatically.
20	Positive software limit	The positive software position limit was hit; the motor was stopped.
21	Negative software limit	The negative software position limit was hit; the motor was stopped.
22	Negative limit switch	The negative limit switch was triggered on the referenced axis. Do not configure the digital I/O as a reference switch if this error is undesired.
23	Positive limit switch	The positive limit switch was triggered on the referenced axis. Do not configure the digital I/O as a reference switch if this error is undesired.
25	Limit-switch timeout	No limit switch found within 300 s.
26	Position-sensor error	Calibration error. May indicate a mechanical defect.
27	Power-stage error	The drive stage reports its own fault input. In current firmware this is distinct from the measured under-voltage and over-voltage checks above. Inspect the supply wiring, the local capacitor near the axis, cooling, and load conditions.
28	Watchdog timeout	If the watchdog is enabled and not reset by the host in time, the motor is switched off immediately.
50	Limit-switch min. distance	The limit switch is too close to the index pulse of the encoder. Move the motor mechanically so that there are at least 5° between the encoder index and the motor zero position.

Table 21: Drive error codes

5. Firmware upgrade

Application firmware can be updated from the Web GUI in the “Maintenance” menu. New firmware files are announced at www.henschel-robotics.ch/Firmware.

5.1. Drive firmware

The “Firmware upgrade” card on the “Maintenance” tab uploads new application firmware to the selected drive. The current GUI file picker accepts .bin, .cgi, and .txt. In normal operation, use the HDrive firmware package provided for the respective drive.

5.2. CAN slave firmware upgrade

When the drive is configured as a CAN master, each online CAN slave card in the “CAN Slaves” sidebar provides its own “Upload firmware” button.

Current GUI and firmware behavior:

- CAN must be enabled on the master, and the master must be in CAN master mode.
- The target slave must be registered in slot 0 to 7 (slaveID1 to slaveID8) and currently online.
- Only one CAN slave firmware transfer can be active at a time.
- The GUI first reads the slave’s active application (m3/s15), then selects the inactive half of the combined HDrive firmware image and flashes only that bank.
- The transfer runs in 0x250-byte CAN chunks, batched as two chunks per HTTP request between browser and master. After the last chunk, the master requests CRC verification, application switch, and restart on the slave.
- During a slave firmware transfer, master firmware upload, Web GUI upload, and slave parameter upload/download are blocked in the GUI.
- The slave motor is forced off during the upgrade. After the final switch command, the slave reboots immediately; a missing final reply can still mean the new image was accepted successfully.

5.3. Web GUI update

The Web GUI can be updated separately via the “Web GUI update” card in the “Maintenance” tab. The uploaded file format is .htmlmin. If no main GUI is installed on the drive, entering the IP address may show only a blank page.

In any case, the address <http://192.168.1.102/fallback.html> can be used to load a valid GUI from Henschel-Robotics or a custom GUI onto the drive.

Fallback

This is a fallback page. It means that your HDrive is running fine but seems not to have a valid web-GUI stored. Here you can upload a new web GUI or Firmware into your HDrive.

The screenshot shows a web interface with two main sections: 'Web GUI Update' and 'Firmware Update'. Both sections have a green header bar with the section name. Below each header, there is a brief instruction. In the 'Web GUI Update' section, the instruction is 'This loads a new web GUI into the HDrive memory'. Below this is a button labeled 'Upload Web GUI' and a progress bar. At the bottom of the section, it says 'upload status: upload not started yet.' The 'Firmware Update' section has the instruction 'Please chose a valid firmware file below to start the firmware upload process. The process will take several seconds.' Below this is a button labeled 'Upload Firmware' and a progress bar. At the bottom of the section, it also says 'upload status: upload not started yet.'

Figure 10: Firmware upgrade in the Web GUI

5.4. Factory reset

If the IP address is unknown or the drive has been misconfigured, the factory settings can be restored. To do this, toggle the following digital I/O pin three times within 1 second to V_{CC} (24 V) during startup:

HDrive17	HDrive17-ETH-i	HDrive17-ETH-i Rev. 7+
Pin 1	Pin 4 (GPIO-B)	Pin 6 (GPIO-B)

Table 22: Factory-reset pins

If the reset is successful, the LED flashes white rapidly to indicate bootloader mode. New firmware can then be loaded through a web browser.

6. Object dictionary

The drive's Object Dictionary (OD) is the internal int32 table used for live values, setpoints, configuration data, and CAN-master state. All OD access uses a main key *m* and a sub-key *s*.

For raw `getData.cgi?obj=` access, the packed key is `objKey = (subKey << 8) | mainKey`.

The firmware allocates 75 cells per main-key row internally, but each row has its own valid sub-key limit. Requests outside the valid range are rejected.

Sub-key tables for the persistent `HDriveConfigFile` subset are listed in Section 6.3. All other main keys are summarized by valid range only; see `Dictionary/ObjectDictionary.h` for the full firmware enumeration.

6.1. Main keys and valid sub-key ranges

Main key	Valid sub-keys	Purpose
0	0 - 52	Actual motor data: measured live values and diagnostic counters
1	0 - 11	Demanded values: target position, speed, torque, mode, and related setpoints
2	0 - 26	Control values: controller gains, filters, bandwidth hints, and tuning mirrors
3	0 - 16	Stored values: firmware version, hardware version, last error, MAC, active application, bootloader version
4	0 - 39	Communication values: Ethernet settings, ticket selection, CAN role, slave registration, watchdog, DHCP, CAN enable
5	0 - 45	Motor-system values: gearbox, digital I/O configuration, limits, offsets, step/dir settings, ripple-related settings
6	0 - 7	Motor values: electrical and mechanical motor constants
7	0 - 74	CAN-master slave workspace: per-slave feedback, target values, and command flags
8	0 - 50	CAN-master advanced slave workspace: <code>slvobj</code> mailbox, CAN diagnostics, and CAN slave firmware-update status
9	0 - 14	Path planner values

Table 23: Object dictionary main keys and valid sub-key ranges

6.2. Persistent configuration files

The Web GUI parameter-file format (`HDriveConfigFile`) intentionally covers only the persistent configuration subset below. CAN slave parameter upload/download uses the same subset and validates it before writing.

Main key	Export/import range	Meaning
2	0 - 26	Control tuning and controller helper values
4	0 - 39	Communication, Ethernet, and CAN configuration
5	0 - 45	Mechanical, digital I/O, limit, and calibration-related configuration
6	0 - 7	Motor constants

Table 24: Current `HDriveConfigFile` OD subset

6.3. `HDriveConfigFile` sub-keys

Parameter export (`sendHDriveConfig.cgi`) and import (`sav upload`) use CSV lines `main,sub,value`. The drive writes every sub-key in the ranges below (same order as the download file: `m2`, then `m5`, then `m4`, then `m6`). Values are signed int32; GUI scaling notes are given where applicable.

6.3.1. Main key 2 — control values (m2/s0 ... m2/s26)

s	Name	Meaning
0	positionReached	Position-in-window flag (runtime; still stored in export)
1	currPCas	Current-controller P gain (GUI: mV/A)
2	currICas	Current-controller I gain (GUI: V/(A·s))
3	currKbCas	Current back-EMF / feedforward gain
4	speedPCas	Speed-controller P gain
5	speedICas	Speed-controller I gain
6	speedFgCas	Speed friction / torque feedforward
7	speedKbCas	Speed KB term
8	speedImaxCas	Speed integrator clamp
9	posPCas	Position-controller P gain
10	posMaxSpeedCas	Position-loop speed limit (RPM)
11	posMaxSlewRateCas	Position slew-rate limit
12	posPreFilter	Position pre-filter coefficient
13	posFFFiter	Position feed-forward filter
14	currentCtrlBW	Current-loop bandwidth hint for auto-tuning
15	speedCtrlBW	Speed-loop bandwidth hint
16	positionCtrlBW	Position-loop bandwidth hint
17	currentCtrlDF	Current-loop damping / derivative factor
18	speedCtrlDF	Speed-loop damping factor
19	positionCtrlDF	Position-loop damping factor
20	filterFreqAB	A/B encoder filter corner frequency
21	filterFreqQD	Q/D (dq) current filter corner frequency
22	calcParameter	Trigger: recalculate CAS parameters from bandwidth fields (1 = calculate)
23	filterFreqPos	Position measurement filter corner frequency
24	CTRL_DATA_targetSpeed	Mirrored speed target for tuning UI
25	CTRL_DATA_targetPosition	Mirrored position target for tuning UI
26	CTRL_DATA_targetCurrent	Mirrored current target for tuning UI

Table 25: HDriveConfigFile sub-keys — main key 2 (control values)

6.3.2. Main key 5 — motor-system values (m5/s0 ... m5/s45)

s	Name	Meaning
0	gearDen	Virtual gearbox denominator
1	gearNum	Virtual gearbox numerator
2	PIN2	Multifunction pin 2 configuration
3	PIN4	Multifunction pin 4 configuration (also step/dir variant select)
4	PIN5	Multifunction pin 5 configuration
5	PIN6	Multifunction pin 6 configuration
6	PIN7	Multifunction pin 7 configuration
7	PIN8	Multifunction pin 8 configuration
8	stepWidth1	Step/Dir resolution when PIN4 is low
9	stepWidth2	Step/Dir resolution when PIN4 is high
10	phaseOffset	Electrical angle trim (high speed)
11	stepTorque	Step/Dir or PWM torque scale (mNm)
12	stepSpeed	Step/Dir speed / feed-forward scale (RPM)
13	limitMinimum	Negative software position limit (1/10°)
14	limitMaximum	Positive software position limit (1/10°)
15	limitEnable	Enable software limit processing
16	pathPlannerEnabled	Enable coordinated path planner (m9)
17	positionOffset_deg10	Position zero offset (1/10°)
18	switchKind	Limit-switch polarity (1 = high-active, 0 = low-active)
19	blinkWhenWebGUIConnected	Blink status LED while Web GUI session is active
20	researchMode	Measurement / experiment mode selector
21	chirpType	Chirp waveform (1 = sine, 0 = square)
22	chirpStartFreq	Chirp start frequency
23	chirpStopFreq	Chirp stop frequency
24	chirpAmpl	Chirp amplitude
25	chirpOffset	Chirp DC offset
26	chirpDelay	Delay before chirp starts
27	chirpLength	Chirp duration
28	speedOffset	Constant speed trim (RPM)
29	stepSlewRate	Step/Dir slew-rate limit
30	phaseOffInvPark	Inverse-Park phase trim
31	calibrationValid	Ripple / torque calibration blob valid flag
32	PIN_OUT_1	Digital output 1 mapping
33	PIN_OUT_2	Digital output 2 mapping
34	PIN_OUT_3	Digital output 3 mapping
35	PIN_OUT_4	Digital output 4 mapping
36	fiveVoltGPIO	Enable 5 V tolerant digital-input comparator reference
37	stepAcc	Step/Dir profile acceleration
38	stepDec	Step/Dir profile deceleration
39	measuredPWMFrequency	Measured inverter PWM frequency (Hz)
40	torqueRippleGain	Ripple torque compensation gain
41	torqueRipplePhaseOffset_deg10	Ripple compensation phase (1/10°)
42	measureRippleCurrent	Enable ripple-current logging
43	rippleCurrentCalibrationValid	Ripple current LUT loaded flag
44	currentScaleA	Current sensor scale channel A
45	currentScaleB	Current sensor scale channel B

Table 26: HDriveConfigFile sub-keys — main key 5 (motor-system values)

6.3.3. Main key 4 — communication values (m4/s0 ... m4/s39)

s	Name	Meaning
0	ipA	Device IPv4 address octet A
1	ipB	Device IPv4 address octet B
2	ipC	Device IPv4 address octet C
3	ipD	Device IPv4 address octet D
4	ipuA	UDP destination octet A (0 with B–D → use TCP client IP)
5	ipuB	UDP destination octet B
6	ipuC	UDP destination octet C
7	ipuD	UDP destination octet D
8	subA	Subnet mask octet A
9	subB	Subnet mask octet B
10	subC	Subnet mask octet C
11	subD	Subnet mask octet D
12	gwA	Default gateway octet A
13	gwB	Default gateway octet B
14	gwC	Default gateway octet C
15	gwD	Default gateway octet D
16	tcpPort	TCP XML command port (factory default 1000)
17	udpPort	UDP telemetry port (factory default 1001)
18	preescaler	Telemetry rate prescaler (4 kHz base tick)
19	sendUDP	1 = periodic telegrams on UDP, 0 = on active TCP command link
20	canID	CAN node ID when this drive is a CAN slave (1–8)
21	RXTicket	Host-to-drive ticket decoder selection
22	TXTicket	Drive-to-host ticket format selection
23	canMaster	CAN role (0 = slave, non-zero = master in GUI)
24	slaveID1	Registered CAN slave ID for slot 1 (0 = unused)
25	slaveID2	Registered CAN slave ID for slot 2
26	slaveID3	Registered CAN slave ID for slot 3
27	slaveID4	Registered CAN slave ID for slot 4
28	slaveID5	Registered CAN slave ID for slot 5
29	slaveID6	Registered CAN slave ID for slot 6
30	slaveID7	Registered CAN slave ID for slot 7
31	slaveID8	Registered CAN slave ID for slot 8
32	CAN_Baudrate	CAN bitrate encoding / register value
33	webGUIActiveCounter	Web-GUI activity heartbeat (runtime counter)
34	autosend	Enable periodic TCP/UDP telemetry (1 = on)
35	watchdogEnabled	Communication watchdog enable
36	watchdogTimeout	Watchdog timeout (control-loop ticks)
37	DHCP_used	Use DHCP for IPv4 configuration
38	CAN_used	Enable CAN peripheral
39	GetTicketOnce	One-shot full telemetry ticket request

Table 27: HDriveConfigFile sub-keys — main key 4 (communication values)

6.3.4. Main key 6 — motor values (m6/s0 ... m6/s7)

s	Name	Meaning
0	motorResistance	Phase resistance (mΩ)
1	motorInductance	Phase inductance (μH)
2	motorMaxI	Maximum allowable winding current
3	motorKm	Torque constant Kt (mNm/A)
4	motorInertia	Rotor inertia (g·cm ²)
5	motorDamping	Viscous damping coefficient
6	motorResonance	Mechanical resonance hint / filter
7	motorIsNema23	1 = NEMA23 thermal model branch

Table 28: HDriveConfigFile sub-keys — main key 6 (motor values)

6.4. Frequently used keys

OD key	Name	Meaning
m3/s0	Software version	Application firmware version
m3/s1	Hardware version	Hardware revision code
m3/s4	Last error	Most recent error code
m3/s15	Active application	1 = APP1, 2 = APP2
m3/s16	Bootloader version	Bootloader version reported by the drive
m4/s4 - m4/s7	UDP host address	Destination for outgoing UDP telegrams. All octets 0 (0.0.0.0) uses the IPv4 address of the active TCP command client.
m4/s16	TCP port	XML command channel (factory default 1000)
m4/s17	UDP port	Outgoing UDP telegrams (factory default 1001)
m4/s18	Autosend prescaler	Divider for periodic TCP/UDP telemetry rate
m4/s19	sendUDP	1 = periodic telegrams on UDP, 0 = on the active TCP command connection
m4/s22	TXTicket	Selected outgoing telegram format
m4/s34	autosend	Enables periodic outgoing TCP/UDP telegrams
m4/s20	CAN node ID	CAN ID used when this drive runs as a CAN slave. Current firmware supports IDs 1 to 8.
m4/s23	CAN role	0 = master, 1 = slave in the current firmware and GUI
m4/s24 - m4/s31	Slave registration	Configured CAN slave IDs for slots 1 to 8
m4/s38	CAN enable	Enables or disables the CAN interface
m5/s0	Gear denominator	Virtual gearbox denominator
m5/s1	Gear numerator	Virtual gearbox numerator
m5/s18	Limit-switch polarity	1 = high-active, 0 = low-active
m8/s24 - m8/s30	slvobj mailbox	CAN slave-object operation, target slot/key, result, and mailbox status
m8/s33 - m8/s41	Slave FW update status	Current slot, operation, progress, last acknowledged chunk, error code, and timeout/liveness data for CAN slave firmware update

Table 29: Frequently used OD keys

7. CAN slave object access

This section describes how to read and write a CAN slave Object Dictionary (OD) entry through the CAN master by means of `getData.cgi`.

7.1. Preconditions on the master

- CAN is enabled
- Master mode is active
- Slave IDs are configured
- The slave is online on the bus

7.2. Request format

7.2.1. Read

```
getData.cgi?slvobj=r_<slot>_<mainKey>_<subKey>
```

Example: read the actual voltage of slave 2:

```
getData.cgi?slvobj=r_1_0_6
```

- slot 1 → slaveID2
- mainKey = 0 (actualMotordata)
- subKey = 6 (actualVoltage)

7.2.2. Write

```
getData.cgi?slvobj=w_<slot>_<mainKey>_<subKey>_<value>
```

Example: write `lastError` of slave 2 to 777:

```
getData.cgi?slvobj=w_1_3_4_777
```

- slot 1
- mainKey = 3 (storedValues)
- subKey = 4 (lastError)
- value 777

7.2.3. Slot mapping

- slot 0 → slaveID1
- slot 1 → slaveID2
- ...
- slot 7 → slaveID8

7.3. Response (current firmware)

Read and write URLs (`slvobj=r_...` / `slvobj=w_...`) are handled **synchronously** inside the firmware: the HTTP handler waits until the CAN transaction finishes (or fails), then returns **plain text** in the response body.

1. Success: OK <integer> — the slave OD value (int32).
2. Failure: ERR <code> — see Table 30 below.

The handler waits by yielding with RTOS sleep calls, so motion control and CAN ISR paths continue to run. Only the HTTP worker serving the request is blocked for the duration of the transaction.

Code	Meaning
ERR 1	Not usable — CAN disabled or device not in CAN master mode
ERR 2	Bad URL / invalid slot or OD indices
ERR 3	Timeout waiting for the slave reply
ERR 4	CAN controller reports bus-off
ERR 5	Slave-side error — invalid OD indices or rejected reply

Table 30: HTTP body errors after slvobj=r_... or slvobj=w_...

7.4. Poll helper slvobj=s

getData.cgi?slvobj=s (optional trailing & ignored) returns one line: <CAN_objectStatus> <CAN_objectResultValue> — useful for debugging or tooling.

This does **not** start a transaction; it only mirrors the master's OD mailbox.

7.5. Read flow

7.5.1. Current firmware

1. Send: getData.cgi?slvobj=r_<slot>_<main>_<sub>.
2. Read the HTTP body: OK <value> on success, or ERR <code> on failure.

7.5.2. Legacy firmware (QUEUED response)

1. Send the same read URL.
2. Confirm the response body starts with QUEUED.
3. Poll obj=7688 (status) and obj=7432 (result) every 100–300 ms.
4. Stop when status is not 2.
5. If status is 3, the result key holds the value.

7.6. Write flow

7.6.1. Current firmware

1. Send: getData.cgi?slvobj=w_<slot>_<main>_<sub>_<value>.
2. Read the HTTP body: OK <value> (echo of written cell) or ERR <code>.

7.6.2. Legacy firmware

1. Send the write URL; if the body is QUEUED, poll status as in the legacy read flow until status 3.

7.7. Common issues

- ERR 2 immediately after the request — request format is wrong, or the firmware is too old for the route.
- ERR 3 / ERR 4 / ERR 5 — timeout, CAN bus-off, or slave OD/reply error respectively (OK/ERR body).
- Legacy only: status stays at 2 and then becomes 5 — the slave did not reply (check CAN ID mapping, bus, and slave firmware support).
- Legacy only: result looks stale — only trust the result when the current request reaches status 3.
- The slave stops reporting after the first object request — slave firmware needs a dedicated TX mailbox for object replies.

8. Technical data

8.1. Factory default values

Setting	Value
DHCP	Off
IP address	192.168.1.102
Subnet mask	255.255.255.0
TCP port	1000
UDP port	1001
UDP host	0.0.0.0 (use TCP command client IP)

Table 31: Factory-reset defaults

8.2. Electrical data

8.2.1. Supply

Name	Function	Unit	Value
VCC	Supply voltage	V	12 - 24
Max. current (2 s)		A	3
Continuous current		A	2

Table 32: Supply

Note: The HDrive includes reverse-polarity protection. If supply polarity is reversed, the drive does not start. Correct the wiring before power-up. Do not rely on the protection as part of normal operation.

8.2.2. Electrical absolute maximum values

Name	Function	Unit	Value
VCC	Supply voltage	V	48 (HW rev 6a)
Max current	Supply current	A	3
Digital In	Digital input	V	0 - 24

Table 33: Absolute maximum values

8.2.3. Motor characteristics

Name	Min	Typ	Max	Unit
No-load speed	-800		800	RPM
Holding torque	-0.6	—	0.6	Nm
Single-turn absolute encoder		14		bit
Absolute accuracy		±0.1 (1σ)		°
Torque constant		200		mNm/A

Table 34: Motor characteristics

8.2.4. Timing characteristics

Name	Min	Typ	Max	Unit
Current control frequency		19		kHz
Position/speed control frequency		6		kHz
Power-up start time		2		s
Position data from drive (TCP) max.		—	0.5	kHz
Position data from drive (UDP) max.			2	kHz

Table 35: Timing characteristics

8.2.5. Environmental conditions

Item	Value
Ambient temperature	−10 °C to +40 °C, non-condensing

Table 36: Environmental conditions

8.2.6. LED indications

LED	Function
Green flashing	Drive ready
Green steady	Power stage enabled
Red flashing	Fault present
Red/green fast flashing	Firmware reset
Blue flashing	Active browser access
Blue steady	Power stage enabled and browser access active
Yellow flashing	Drive ready, no Ethernet link

Table 37: LED indications

8.3. Digital I/O

The HDrive inputs and outputs can be configured individually. Each GPIO can be defined as an input or an output. A 5 V logic mode can be enabled in the Web GUI so that the inputs can interface directly with 5 V logic.

8.3.1. 24 V mode

Digital-In voltage 0 - 24 VDC
 Logic 0 < 4.8 V
 Logic 1 > 14.4 V

8.3.2. 5 V mode

Digital-In voltage 0 - 5 VDC
 Logic 0 < 0.8 V
 Logic 1 > 2.0 V

8.3.3. Digital input

The inputs can be configured as low-active or high-active. An input can be switched to 0 V or +24 V to trigger the configured function.

8.4. Digital output

Each digital I/O can be configured as a digital output. The output stage is open-drain and can switch a load up to 100 mA.

To connect a load, feed the load from 24 V and connect the return side of the load to the drive I/O pin. When the output is activated, the open-drain stage switches the load to supply GND.

Note: All logic inputs on the drive are 24 V compatible.

8.5. Torque curve, weight, and dimensions

This section provides the principal dimensional data for the HDrive variants.

8.5.1. HDrive17-ETH-i dimensions

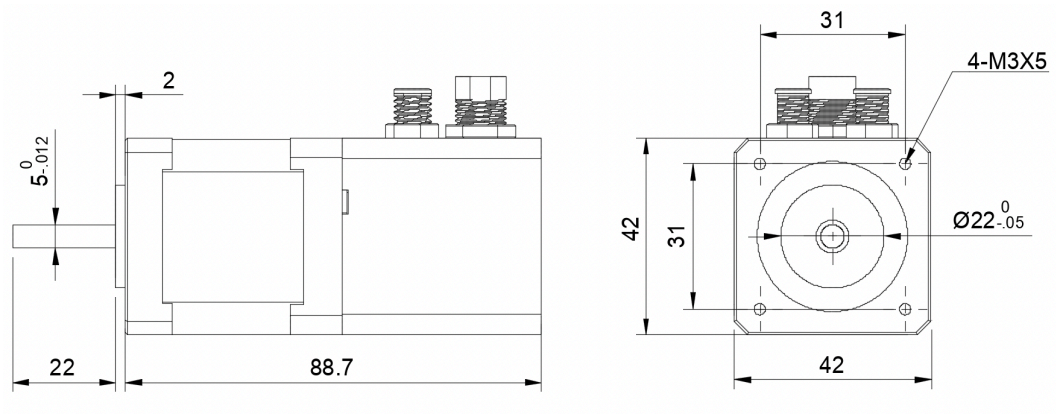


Figure 11: HDrive17-ETH-i dimensions in mm; weight 470 g

8.5.2. HDrive23-ETH-i dimensions

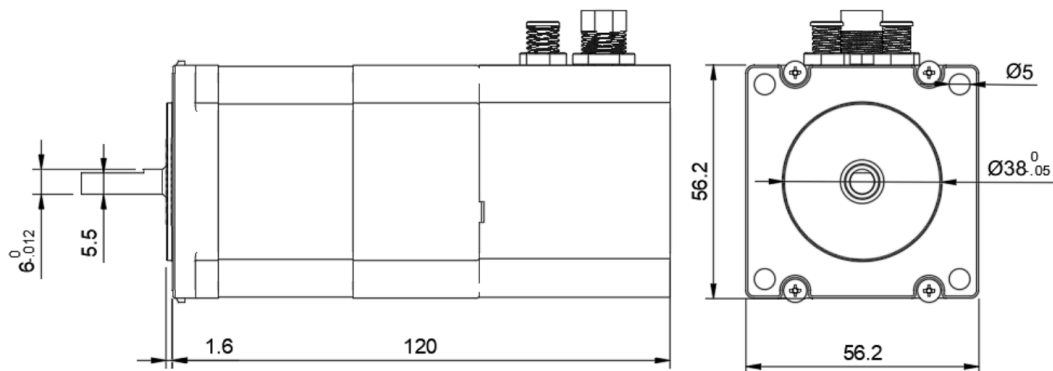


Figure 12: HDrive23-ETH-i dimensions in mm; weight 1250 g