Zubax EPM Datasheet

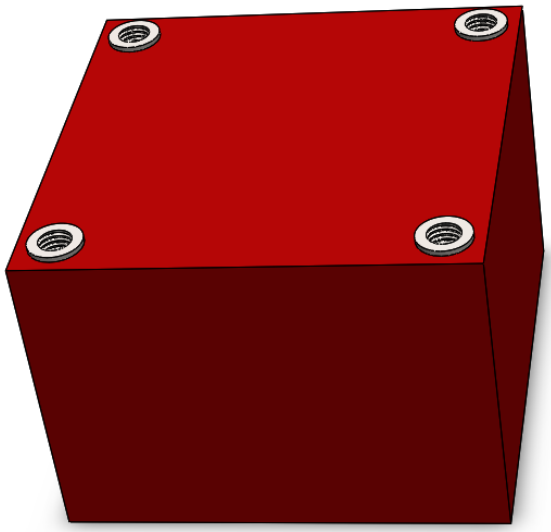
Zubax Robotics
Akadeemia rd. 21/1, Tallinn 12618, Estonia
info@zubax.com
Q&A: forum.zubax.com

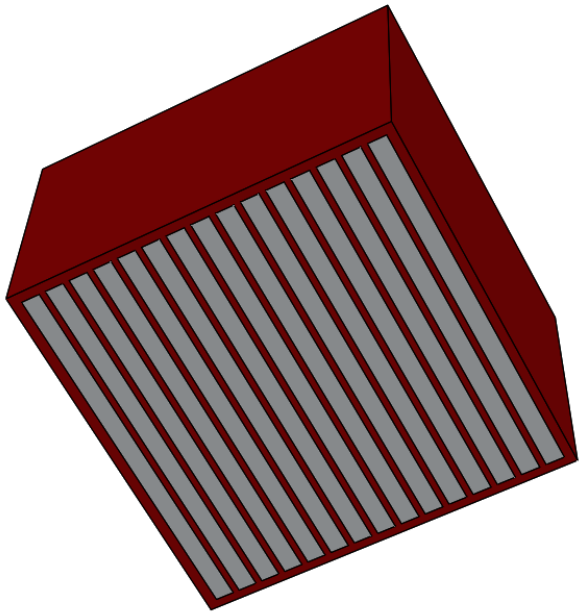
Revision 2023-02-22

Overview

Zubax EPM is an *electro-permanent magnet*. It combines the advantages of electro- and permanent magnets by being able to switch between the on-state and off-state on demand (like an electromagnet); while consuming zero power in either state (like a permanent magnet). The device supports a variety of communication interfaces that enable easy integration into any end system.

Zubax EPM follows high safety and reliability standards that ensure dependable operation in hostile environments.





Applications

- Payload attachment in unmanned aerial vehicles.
- Workholding in CNC machines, manipulators, and robots.
- Magnetic locks, clamps, and holders.

Features

- Strong magnet in a very compact and lightweight form-factor.
- Zero power consumption in either on or off state.
- Fast on↔off state transition in approx. 1 second.
- Low (de-)magnetization energy consumption.
- Entirely solid-state construction, no moving parts inside.
- IP68 rated — dust-tight, submersible.
- AEC-Q grade 2 electronics.
- Resistant to vibration, mechanical stress, high and low temperatures, electromagnetic interference, humidity, dust, and some aggressive chemicals.
- Wide power supply voltage range ensures trivial integration into the end system.
- Compatible with industry-standard interfaces:
  - Cyphal/CAN FD control with diagnostic telemetry;
  - RCPWM control input;
  - voltage level control input.

Table of contents

List of tables

List of figures

1 Operating principles

DANGER: The device contains high voltage electronics and poses a severe electric shock and fire hazard if disassembled, even if the external power supply is disconnected. **Do not attempt to disassemble or modify the construction of the device in any fashion.**

Certain features are available only in specific models; for details, see section 2.1.

1.1 Fundamentals

Zubax EPM is an electropermanent magnet that contains a remagnetizable magnetic assembly and the control electronics for it in a single unit. The magnetic assembly can be magnetized, which is referred to as the on-state, and demagnetized, which is referred to as the off-state. Both states are stable in the sense that the device can reside in either state for an unlimited time while powered down; external power is required only for transitioning between the two states. A full state transition is usually performed in approximately 1 second, depending on the voltage and impedance of the power supply; however, most of the magnetization is induced instantaneously.

When the device is powered on but does not perform on-off state transitions — or, in other words, resides in the *idle state* — some insignificant power is consumed by the built-in electronics and its external interfaces.

The device accepts the on/off (magnetization) commands via the external interfaces in the form of the desired state. By virtue of the built-in magnetic flux sensors, the device is aware of the current magnetization state and can avoid unnecessary energy expenditure and disturbance of the magnetic circuit if its current state matches the commanded state. Regardless, as will be shown later, sometimes it is desirable to force a magnetization cycle even if the magnet is already magnetized. Therefore, the device accepts three commands: *OFF*, *ON*, and *FORCE*, whose handling is described in the following state transition table.

Command	State	Action
OFF	Off	
OFF	On	Demagnetize
ON	Off	Magnetize
ON	On	
FORCE	(any)	Magnetize once

Table 1.1: Magnet state transition

Having received a *FORCE* command, the device will execute one magnetization cycle regardless of the current state and ignore further *FORCE* commands until any other command is received.

1.2 Interface and power supply

The on-off commands are delivered to the device via any of its communication interfaces, which are:

- Cyphal[1] (with redundancy as an option for high-integrity systems) — described in section 3.
- RCPWM (only in some models) — described in section 4.2.
- Logic level (only in some models) — described in section 4.3.

High-level interfaces such as Cyphal publish telemetry data that provides insights into the state and performance of the device, which is an important feature for high-integrity low-maintenance systems. Additionally, minimal status reporting is implemented via the pull resistor connected to the analog input port for applications that are unable to leverage the Cyphal interface.

The device is powered either from the Cyphal network interface or via the analog port connector. The built-in voltage regulator accepts a wide range of input supply voltages which allow easy integration of the magnet into the end application without the need for additional power converters.

[1] https://opencyphal.org

1.3 Magnetic circuit

The *target* (payload) is attached to the bottom surface of the magnet where the metallic magnetic guides are exposed. The target forms a closed magnetic circuit with the exposed magnetic guides, which creates the force pulling the target towards the magnet. The force that is required to pull the target away from the magnet is referred to as the *holding force*. The holding force is an essential characteristic and it depends not only on the performance of the magnet itself, but also on the qualities of the target. To maximize the holding force, the following conditions have to be considered:

- The target should be made of a highly magnetically permeable material with $\mu_r > 1000$, e.g., electrical steel. In case of sheet metal, the thickness of the sheet should not be less than 1 mm.
- The target should be at least the same size as the outer dimensions of the working surface of the magnet. In other words, the target should cover the magnet entirely.
- The target should be in direct contact with the exposed magnetic guides at the working surface of the magnet in their entirety. Any obstruction such as dirt, dust, metallic shavings, and other foreign objects may impair the performance of the magnet and cause mechanical damage.
- The target should be in close proximity of the exposed magnetic guides during magnetization (i.e., during the off→on state transition) for best performance. If the magnet is magnetized without the target attached, another — forced — magnetization cycle performed after the attachment of the target may increase the holding force.

Strong magnetic fields in the vicinity of the working surface of the magnet may cause spurious magnetization or demagnetization.

2 Characteristics

2.1 Models

Modifications that are not explicitly listed may be available upon request. Please contact Zubax Robotics for details.

Model name	Width/length	Height	Cyphal/CAN iface	Analog iface
EPM401M	40	30	Single UCANPHY Micro	No
EPM401MA	40	30	Single UCANPHY Micro	Yes
EPM402M	40	30	Double UCANPHY Micro	No

Table 2.1: Model comparison

All dimensions are in millimeters. In the following text, references to model families are made by specifying only the leading symbols of the model name.

2.2 Absolute maximum ratings

Stresses that exceed the limits specified in this section may cause permanent damage to the device. Proper operation of the device within the specified limits is not guaranteed.

Parameter	Condition	Min	Max	Unit
Supply voltage		-0.3	+45	V
Operating temperature (device)		-40	+105	°C
Analog port input voltage		-0.3	+10	V
CAN FD H/L input voltage		-42	+42	V
Force, magnitude of	EPM4		250	N

Table 2.2: Absolute maximum ratings

2.3 Power supply characteristics

Parameter	Condition	Min	Typ	Max	Unit
Supply voltage		4		30	V
Current consumption	idle		10	30	mA
Current consumption	(de-)magnetization			1000	mA
(De-)magnetization energy consumption	EPM4	3	4	5	J

Table 2.3: Power supply characteristics

2.4 CAN FD interface characteristics

2.4.1 General

Zubax EPM may be equipped with a single- or doubly-redundant CAN FD interface, depending on the model. The CAN FD interfaces may be equipped with different connector types, which are specified in dedicated sections. Models that are equipped with doubly-redundant interfaces can be used with non-redundant networks if and only if interfaced via the first (i.e., main) interface.

Additional connector options are available upon request.

Parameter	Condition	Min	Typ	Max	Unit
Differential output voltage, dominant	60Ω load	+1.5		+5.1	V
Differential output voltage, recessive	60Ω load	-0.1	0	+0.1	V
Differential receiver threshold voltage		0.4		1.15	V
Differential receiver hysteresis voltage		0.05		0.3	V
Arbitration phase bit rate		10		1000	kbps
Data phase bit rate		10		4000	kbps

Table 2.4: CAN FD interface characteristics

2.4.2 UCANPHY Micro

The UCANPHY Micro connector is documented extensively in the UCANPHY Specification. This connector type is based on JST GH and is implemented on a cable; external T-connectors may be necessary to attach the device to the CAN bus. The key parameters are summarized in this section.

Pin no.	Type	Function
1	Power	Power supply input, see 2.3
2	Input/output	CAN high
3	Input/output	CAN low
4	Ground	Ground

Table 2.5: UCANPHY Micro connector pinout

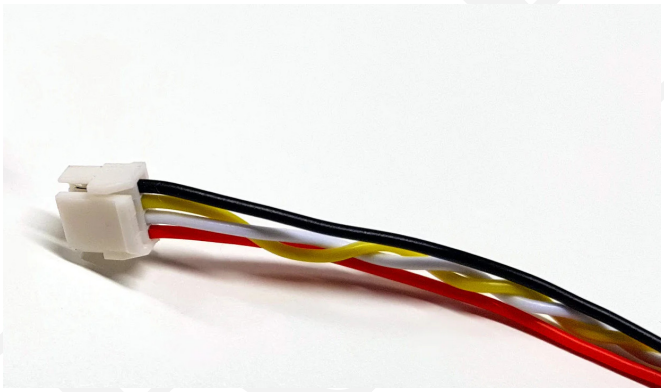


Figure 2.1: CAN bus interface cable terminated with UCANPHY Micro

2.5 Analog interface characteristics

The analog interface is implemented on a wire terminated with the industry-standard 2.54 mm three-pin RCPWM connector. The current state of the magnet is communicated by connecting the pull resistor either to the ground or to some positive voltage level; more on this in section 4. The switchable pull resistor not only provides the state feedback to the controlling system, but also ensures that the magnet resides in the current state if the input is left floating.

Parameter	Min	Typ	Max	Unit
Logic input low			1.0	V
Logic input high	3.8			V
Pull resistor value	4800	5100	5500	Ω

Table 2.6: Analog interface characteristics

Pin no.	Type	Function
1	Input	RCPWM or voltage level input with pull resistor feedback
2	Power	Power supply input, see 2.3
3	Ground	Ground

Table 2.7: RCPWM connector pinout

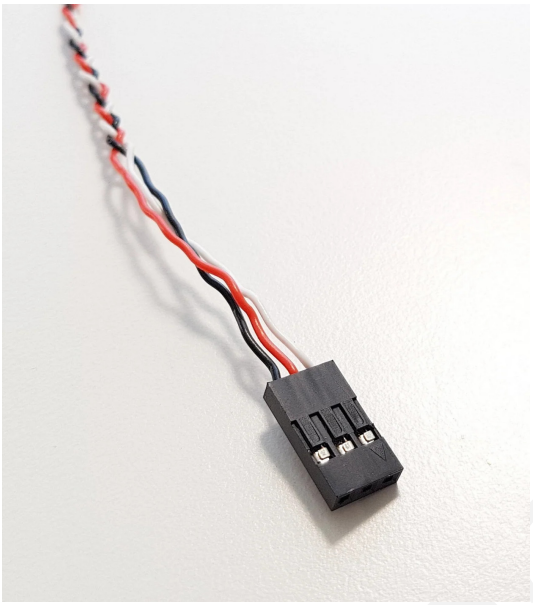


Figure 2.2: Analog interface cable terminated with the industry-standard RCPWM connector

2.6 Mechanical characteristics

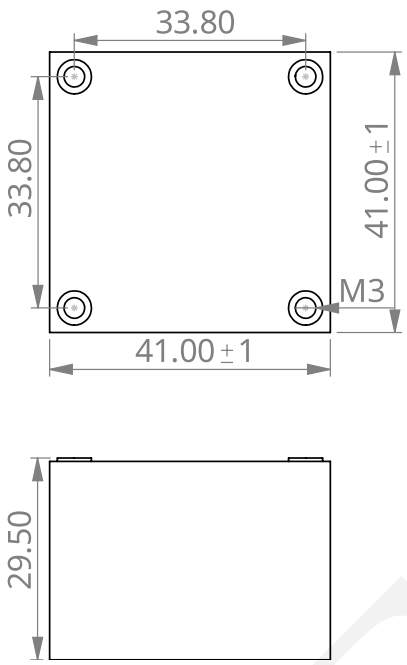
The entire device — the magnetic assembly and the control electronics — is potted in a low-density, high strength epoxy compound that provides IP68 protection rating (dust tight, submersible). The product is highly resilient to vibration and adverse chemicals. The electronics are built with AEC-Q grade 2 components.

2.6.1 EPM4

The power supply and interface connectors are implemented on cables exposed via the top surface of the device. The cables are terminated with connectors documented in the respective sections.

Parameter	Condition	Value	Unit
Mass		70	g
Holding force	Ideal target, section 1.3	200	N
Ingress protection		IP68	
Interface cable length		0.12	m

Table 2.8: EPM4 mechanical characteristics



All dimensions are in millimeters. Interface cables not shown. Not to scale.

Figure 2.3: EPM4 drawing

2.7 Reliability and safety

Please contact Zubax Robotics for additional information about reliability and safety.

Parameter	Value	Unit
Replacement life	8	year
Operational service life	100 000	hour
Remagnetization cycle limit	TBD	cycles
Mean time to failure (MTTF)	TBD	hour

Table 2.9: Reliability

2.7.1 Quality assurance

Every manufactured device undergoes an automated hardware verification process, the test logs of which can be viewed at https://device.zubax.com/device_info. To facilitate traceability and reduce the risk of counterfeits, every manufactured device stores a strong digital signature in its non-volatile memory to identify its origin.

Additional information is available upon request from Zubax Robotics.

3 Cyphal interface

3.1 General

Cyphal was known as UAVCAN until March 2022. For historical reasons, registers related to the Cyphal stack are named with the "uavcan." prefix.

Zubax EPM utilizes data types from the vendor-specific DSDL namespace zubax[2]. Human-friendly documentation for this and other DSDL namespaces can be viewed at http://nunaweb.opencyphal.org.

3.1.1 Plug-and-play node-ID allocation

By default, the local Cyphal node does not have an assigned node-ID; that is, the uavcan.node.id (page 15) register is set to 65535 ($FFFF_{16}$). When this is the case, EPM will commence the standard plug-and-play node-ID allocation sequence defined in the Cyphal Specification[3].

Once the node-ID allocation is completed, the uavcan.node.id (page 15) register will be updated with the allocated node-ID value.

3.2 Publications

For the standard configuration registers pertaining to the non-fixed-port-ID topics, refer to section 5.2. Priority of the non-fixed-port-ID topics is configurable.

Topic name	Type	Priority	Period	Timeout	Description
	uavcan.node.Heartbeat.1	4 (nominal)	1 s	period	Sect.3.2.1
	uavcan.node.port.List	7 (optional)	10 s	period	
	uavcan.pnp.NodeIDAllocationData.1	6 (slow)	random	1 s	Only during PnP allocation; sect. 3.1.1
feedback	zubax.epm.Feedback.1	4 (nominal)	1 s and on change	1 s	Sect. 3.2.2
flux_density	uavcan.si.sample.magnetic_flux_density.Scalar.1	5 (low)	0.1 s	period	Sect. 3.2.3
input_voltage	uavcan.si.sample.voltage.Scalar.1	5 (low)	0.1 s	period	Sect. 3.2.4
capacitor_voltage	uavcan.si.sample.voltage.Scalar.1	5 (low)	0.1 s	period	Sect. 3.2.5
temperature	uavcan.si.sample.temperature.Scalar.1	6 (slow)	1 s	period	Sect. 3.2.6

Table 3.1: Cyphal publications

3.2.1 Heartbeat

The mode field is set to OPERATIONAL unless the device is running in the bootloader mode (section 6), in which case the mode is SOFTWARE_UPDATE and the rest of this section does not apply.

The health field is set to NOMINAL unless the self-diagnostic indicates one of the problems listed below, in which case the health is reported as CAUTION:

- The magnetic assembly is suspected to have sustained mechanical or electrical damage.
- The internal sensors are suspected to be damaged or yield inconsistent data.
- The internal high-voltage circuitry is not operating correctly.

The health is reported as ADVISORY in the following cases:

- The temperature is outside the design limits.

[2] https://github.com/Zubax/zubax_dsdl
[3] https://opencyphal.org/specification

Definition of `uavcan.node.Heartbeat.1.0`; extent 12 bytes:

```
# Abstract node status information.
# This is the only high-level function that shall be implemented by all nodes.
#
# All Cyphal nodes that have a node-ID are required to publish this message to its fixed subject periodically.
# Nodes that do not have a node-ID (also known as "anonymous nodes") shall not publish to this subject.
#
# The default subject-ID 7509 is 1110101010101 in binary. The alternating bit pattern at the end helps transceiver
# synchronization (e.g., on CAN-based networks) and on some transports permits automatic bit rate detection.
#
# Network-wide health monitoring can be implemented by subscribing to the fixed subject.

uint16 MAX_PUBLICATION_PERIOD = 1   # [second]
# The publication period shall not exceed this limit.
# The period should not change while the node is running.

uint16 OFFLINE_TIMEOUT = 3          # [second]
# If the last message from the node was received more than this amount of time ago, it should be considered offline.

uint32 uptime                       # [second]
# The uptime seconds counter should never overflow. The counter will reach the upper limit in ~136 years,
# upon which time it should stay at 0xFFFFFFFF until the node is restarted.
# Other nodes may detect that a remote node has restarted when this value leaps backwards.

Health.1.0 health
# The abstract health status of this node.

Mode.1.0 mode
# The abstract operating mode of the publishing node.
# This field indicates the general level of readiness that can be further elaborated on a per-activity basis
# using various specialized interfaces.

uint8 vendor_specific_status_code
# Optional, vendor-specific node status code, e.g. a fault code or a status bitmask.

@assert _offset_ % 8 == {0}
@assert _offset_ == {56}  # Fits into a single-frame Classic CAN transfer (least capable transport, smallest MTU).
@extent 12 * 8
```

3.2.2 Feedback

This topic is published to at a low fixed frequency and on change. Applications that do not require the extended status information can subscribe to this topic using `uavcan.primitive.scalar.Bit.1`.

Definition of `zubax.epm.Feedback.0.1`; extent 63 bytes:

```
# Zubax EPM status feedback message.
# This type is a structural subtype of uavcan.primitive.scalar.Bit.

bool magnetized
# True if the magnet is currently turned on.
# The value is a best guess if remagnetization is currently in progress.

int3 remagnetization_state
# -1 -- the magnet is being demagnetized (turning off).
#  0 -- the magnet is idle.
# +1 -- the magnet is being magnetized (turning on).

void4

@assert _offset_ == {8}

uint24[2] cycles_on_off
# The number of magnetization and demagnetization cycles commenced, respectively, since the device was powered on.
# The values are incremented immediately at the commencement of the cycle.

@extent 63 * 8
```

3.2.3 Flux density

Not for production use.

Definition of `uavcan.si.sample.magnetic_flux_density.Scalar.1.0`; extent 11 bytes:

```
uavcan.time.SynchronizedTimestamp.1.0 timestamp
float32 tesla
@sealed
```

3.2.4 Input voltage

The voltage at the supply rail that powers the device; see section 2.3.

Definition of `uavcan.si.sample.voltage.Scalar.1.0`; extent 11 bytes:

```
uavcan.time.SynchronizedTimestamp.1.0 timestamp
float32 volt
@sealed
```

3.2.5 Capacitor voltage

Not for production use.

Definition of `uavcan.si.sample.voltage.Scalar.1.0`; extent 11 bytes:

```
uavcan.time.SynchronizedTimestamp.1.0 timestamp
float32 volt
@sealed
```

3.2.6 Temperature

The temperature inside the device.

Definition of `uavcan.si.sample.temperature.Scalar.1.0`; extent 11 bytes:

```
uavcan.time.SynchronizedTimestamp.1.0 timestamp
float32 kelvin
@sealed
```

3.3 Subscriptions

Topic name	Type	TID timeout	Description
	`uavcan.pnp.NodeIDAllocationData.1`		Only during PnP allocation; sect. 3.1.1
`command`	`uavcan.primitive.scalar.Integer8.1`	0.5 s	Sect. 3.3.1

Table 3.2: Cyphal subscriptions

3.3.1 Command

The published values are interpreted as specified below; values that are not explicitly listed shall not be used. See table 1.1 for the state transition diagram.

Value	Interpretation
0	OFF
1	ON
2	FORCE

Table 3.3: Command subscription handling

The command can also accept `uavcan.primitive.scalar.Bit.1`, since falsity is interpreted as zero, and truth as +1. This enables simplified control of the magnet through a simple boolean flag commanding the desired state.

Definition of `uavcan.primitive.scalar.Integer8.1.0`; extent 1 bytes:

```
int8 value
@sealed
```

3.4 RPC servers

Type	Description
`uavcan.node.ExecuteCommand.1`	Sect. 3.4.2
`uavcan.node.GetInfo.1`	Sect. 3.4.1
`uavcan.register.Access.1`	Sect. 3.4.3
`uavcan.register.List.1`	Sect. 3.4.3

Table 3.4: Cyphal RPC servers

3.4.1 Node info

The node name is reported as `com.zubax.epm`.

The major hardware version number indicates the first digit of the model number per table 2.1; e.g., 4 for EPM4. The minor hardware version number indicates the sub-model; contact Zubax Robotics for the specifics.

The certificate of authenticity (CoA) field contains the digital signature installed at the factory; see section 2.7.1. Contact Zubax Robotics for the details on its verification.

Definition of `uavcan.node.GetInfo.1.0`; extent 0 / 448 bytes:

```
# Full node info request.
# All of the returned information shall be static (unchanged) while the node is running.
# It is highly recommended to support this service on all nodes.

@sealed

---

Version.1.0 protocol_version
# The Cyphal protocol version implemented on this node, both major and minor.
# Not to be changed while the node is running.

Version.1.0 hardware_version
Version.1.0 software_version
# The version information shall not be changed while the node is running.
# The correct hardware version shall be reported at all times, excepting software-only nodes, in which
# case it should be set to zeros.
# If the node is equipped with a Cyphal-capable bootloader, the bootloader should report the software
# version of the installed application, if there is any; if no application is found, zeros should be reported.

uint64 software_vcs_revision_id
# A version control system (VCS) revision number or hash. Not to be changed while the node is running.
# For example, this field can be used for reporting the short git commit hash of the current
# software revision.
# Set to zero if not used.

uint8[16] unique_id
# The unique-ID (UID) is a 128-bit long sequence that is likely to be globally unique per node.
# The vendor shall ensure that the probability of a collision with any other node UID globally is negligibly low.
# UID is defined once per hardware unit and should never be changed.
# All zeros is not a valid UID.
# If the node is equipped with a Cyphal-capable bootloader, the bootloader shall use the same UID.

@assert _offset_ == {30 * 8}
# Manual serialization note: only fixed-size fields up to this point. The following fields are dynamically sized.

uint8[<=50] name
# Human-readable non-empty ASCII node name. An empty name is not permitted.
# The name shall not be changed while the node is running.
# Allowed characters are: a-z (lowercase ASCII letters) 0-9 (decimal digits) . (dot) - (dash) _ (underscore).
# Node name is a reversed Internet domain name (like Java packages), e.g. "com.manufacturer.project.product".

uint64[<=1] software_image_crc
# The value of an arbitrary hash function applied to the software image. Not to be changed while the node is running.
# This field can be used to detect whether the software or firmware running on the node is an exact
# same version as a certain specific revision. This field provides a very strong identity guarantee,
# unlike the version fields above, which can be the same for different builds of the software.
# As can be seen from its definition, this field is optional.
#
# The exact hash function and the methods of its application are implementation-defined.
# However, implementations are recommended to adhere to the following guidelines, fully or partially:
#   - The hash function should be CRC-64-WE.
#   - The hash function should be applied to the entire application image padded to 8 bytes.
#   - If the computed image CRC is stored within the software image itself, the value of
#     the hash function becomes ill-defined, because it becomes recursively dependent on itself.
#     In order to circumvent this issue, while computing or checking the CRC, its value stored
#     within the image should be zeroed out.

uint8[<=222] certificate_of_authenticity
# The certificate of authenticity (COA) of the node, 222 bytes max, optional. This field can be used for
# reporting digital signatures (e.g., RSA-1776, or ECDSA if a higher degree of cryptographic strength is desired).
# Leave empty if not used. Not to be changed while the node is running.

@assert _offset_ % 8 == {0}
@assert _offset_.max == (313 * 8)     # At most five CAN FD frames
@extent 448 * 8
```

3.4.2 Execute command

The standard command execution service is implemented. The following commands are supported; commands that are not explicitly listed return `STATUS_BAD_COMMAND`; commands whose command code is less than 32768 which are not listed here shall not be invoked, otherwise, permanent hardware damage may occur.

Command	Parameter	Description
STORE_PERSISTENT_STATES		Alias for RESTART.
RESTART		Restarts the device (see 5)
BEGIN_SOFTWARE_UPDATE	Firmware image file name	Restart into the bootloader (see 6)
FACTORY_RESET		Reset all configuration registers to defaults

Table 3.5: Cyphal standard command execution service

Definition of `uavcan.node.ExecuteCommand.1.1`; extent 300 / 48 bytes:

```
# Instructs the server node to execute or commence execution of a simple predefined command.
# All standard commands are optional; i.e., not guaranteed to be supported by all nodes.

uint16 command
# Standard pre-defined commands are at the top of the range (defined below).
# Vendors can define arbitrary, vendor-specific commands in the bottom part of the range (starting from zero).
# Vendor-specific commands shall not use identifiers above 32767.

uint16 COMMAND_RESTART = 65535
# Reboot the node.
# Note that some standard commands may or may not require a restart in order to take effect; e.g., factory reset.

uint16 COMMAND_POWER_OFF = 65534
# Shut down the node; further access will not be possible until the power is turned back on.

uint16 COMMAND_BEGIN_SOFTWARE_UPDATE = 65533
# Begin the software update process using uavcan.file.Read. This command makes use of the "parameter" field below.
# The parameter contains the path to the new software image file to be downloaded by the server from the client
# using the standard service uavcan.file.Read. Observe that this operation swaps the roles of the client and
# the server.
#
# Upon reception of this command, the server (updatee) will evaluate whether it is possible to begin the
# software update process. If that is deemed impossible, the command will be rejected with one of the
# error codes defined in the response section of this definition (e.g., BAD_STATE if the node is currently
# on-duty and a sudden interruption of its activities is considered unsafe, and so on).
# If an update process is already underway, the updatee should abort the process and restart with the new file,
# unless the updatee can determine that the specified file is the same file that is already being downloaded,
# in which case it is allowed to respond SUCCESS and continue the old update process.
# If there are no other conditions precluding the requested update, the updatee will return a SUCCESS and
# initiate the file transfer process by invoking the standard service uavcan.file.Read repeatedly until the file
# is transferred fully (please refer to the documentation for that data type for more information about its usage).
#
# While the software is being updated, the updatee should set its mode (the field "mode" in uavcan.node.Heartbeat)
# to MODE_SOFTWARE_UPDATE. Please refer to the documentation for uavcan.node.Heartbeat for more information.
#
# It is recognized that most systems will have to interrupt their normal services to perform the software update
# (unless some form of software hot swapping is implemented, as is the case in some high-availability systems).
#
# Microcontrollers that are requested to update their firmware may need to stop execution of their current firmware
# and start the embedded bootloader (although other approaches are possible as well). In that case,
# while the embedded bootloader is running, the mode reported via the message uavcan.node.Heartbeat should be
# MODE_SOFTWARE_UPDATE as long as the bootloader is runing, even if no update-related activities
# are currently underway. For example, if the update process failed and the bootloader cannot load the software,
# the same mode MODE_SOFTWARE_UPDATE will be reported.
# It is also recognized that in a microcontroller setting, the application that served the update request will have
# to pass the update-related metadata (such as the node-ID of the server and the firmware image file path) to
# the embedded bootloader. The tactics of that transaction lie outside of the scope of this specification.

uint16 COMMAND_FACTORY_RESET = 65532
# Return the node's configuration back to the factory default settings (may require restart).
# Due to the uncertainty whether a restart is required, generic interfaces should always force a restart.

uint16 COMMAND_EMERGENCY_STOP = 65531
# Cease activities immediately, enter a safe state until restarted.
# Further operation may no longer be possible until a restart command is executed.

uint16 COMMAND_STORE_PERSISTENT_STATES = 65530
# This command instructs the node to store the current configuration parameter values and other persistent states
# to the non-volatile storage. Nodes are allowed to manage persistent states automatically, obviating the need for
# this command by committing all such data to the non-volatile memory automatically as necessary. However, some
# nodes may lack this functionality, in which case this parameter should be used. Generic interfaces should always
# invoke this command in order to ensure that the data is stored even if the node doesn't implement automatic
# persistence management.

uint8[<=uavcan.file.Path.2.0.MAX_LENGTH] parameter
# A string parameter supplied to the command. The format and interpretation is command-specific.
# The standard commands do not use this field (ignore it), excepting the following:
#   - COMMAND_BEGIN_SOFTWARE_UPDATE

@extent 300 * 8

---

uint8 STATUS_SUCCESS         = 0     # Started or executed successfully
uint8 STATUS_FAILURE         = 1     # Could not start or the desired outcome could not be reached
uint8 STATUS_NOT_AUTHORIZED = 2     # Denied due to lack of authorization
uint8 STATUS_BAD_COMMAND     = 3     # The requested command is not known or not supported
uint8 STATUS_BAD_PARAMETER   = 4     # The supplied parameter cannot be used with the selected command
uint8 STATUS_BAD_STATE       = 5     # The current state of the node does not permit execution of this command
uint8 STATUS_INTERNAL_ERROR = 6     # The operation should have succeeded but an unexpected failure occurred
uint8 status
# The result of the request.

@extent 48 * 8
```

3.4.3 Register

The node implements the standard register service `uavcan.register`. The data type definitions are not shown here for reasons of brevity; please consult with the official Cyphal documentation instead.

3.5 Transports

3.5.1 Cyphal/CAN

The product supports the CAN FD bus interface, which may be doubly-redundant depending on the hardware configuration. The primary interface is labeled CAN1 and the redundant, if available, is labeled CAN2.

The value of the register `uavcan.can.count` (page 15) reflects the number of CAN interfaces that are actually used in the application. If a redundant CAN interface is available but only one is used, the used interface shall be strictly CAN1, and the count register shall be set to 1, thereby disabling CAN2. If set to zero, the device will no longer be possible to contact via Cyphal/CAN.

The CAN bitrates are configured via register `uavcan.can.bitrate` (page 15), where the first value sets the arbitration phase bitrate and the second value sets the data phase bitrate. If the configured bitrate is not supported, EPM will automatically revert to the default value.

The CAN MTU is set via `uavcan.can.mtu` (page 15); the only valid values are 8 and 64 bytes. Setting MTU to 8 bytes and both arbitration and data phase bitrates to the same value will disable CAN FD and select Classic CAN instead, which enables compatibility with legacy CAN networks. The Classic CAN compatibility is enabled by default to simplify deployment; the CAN FD support needs to be enabled explicitly if necessary.

Zubax EPM does not support automatic CAN bitrate detection.

4 Analog interface

4.1 Construction

The analog interface allows the device to be controlled via a single pin using either simple voltage levels or the industry-standard RCPWM interface[4] instead of relying on the more capable and complex Cyphal interface.

The analog interface also provides the magnet state feedback via the same pin by driving the pull resistor to the $V_{\text{on}} = 3.0V$ level when the device is magnetized, and to the ground when the magnet is demagnetized. The pull resistor feedback is always enabled regardless of the control mode chosen. The pull resistor feedback can also be used to drive an external LED or other indicators even if the magnet is controlled via Cyphal.

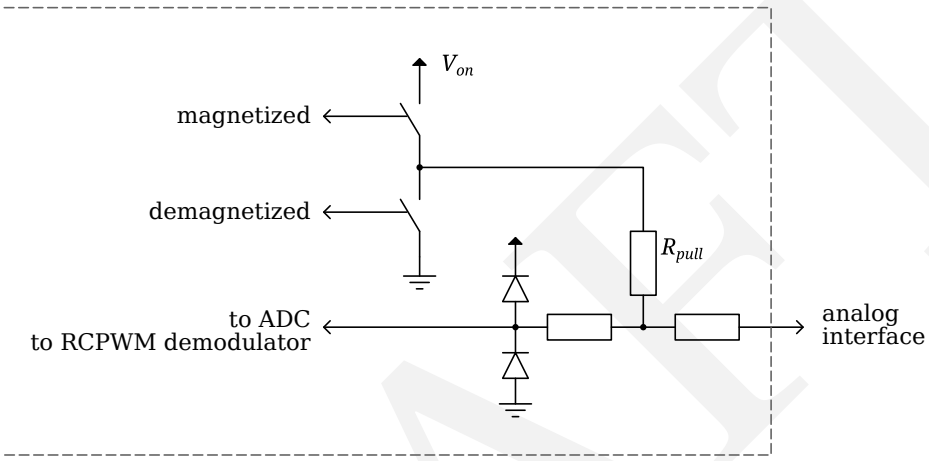


Figure 4.1: Analog interface internal schematic

For the physical characteristics of the analog interface, see section 2.5.

The choice between the RCPWM and analog control mode is made by writing the appropriate value into the `analog.mode` (page 15) configuration register as specified below. Values other than the ones specified below shall not be used.

Value of `analog.mode`	Mode
`rcpwm`	RCPWM control, section 4.2
`level`	Voltage level control, section 4.3
(empty)	Analog control disabled.

Table 4.1: Analog interface mode

4.2 RCPWM control

To control the magnet in the RCPWM control mode, an external source of RCPWM signal needs to be connected to the analog interface. The impedance of the source should be low enough to override the pull resistor feedback.

The RCPWM signal is sampled by the device and the magnet is commanded to the corresponding state as specified in table 4.2. Hysteresis is implemented through the RCPWM pulse durations unaccounted for in the table. See table 1.1 for the state transition diagram.

Bottom	Top	Interpretation
0.1 ms	1.2 ms	OFF
1.4 ms	1.9 ms	ON
2.1 ms	3.2 ms	FORCE

Table 4.2: RCPWM pulse duration thresholds

[4] https://en.wikipedia.org/wiki/Servo_control

4.3 Voltage level control

To control the magnet in the voltage level control mode, a low-impedance external voltage source needs to be connected to the analog interface such that it overrides the pull resistor feedback.

The voltage at the analog interface V_{analog} is measured by the device and the magnet is commanded to the corresponding state as specified in table 4.3. Hysteresis is implemented through the voltage ranges unaccounted for in the table. See table 1.1 for the state transition diagram.

Bottom	Top	Interpretation
	$1.0V$	OFF
$2.0V$	$3.5V$	ON
$4.5V$		FORCE

Table 4.3: Voltage level thresholds

The pull resistor feedback ensures that the magnet will reside in the current state if the input is left floating.

5 Registers

5.1 Implementation

Zubax EPM provides access to its configuration parameters and diagnostic information via Cyphal registers. Per the Cyphal specification, there are four kinds of registers: mutable/immutable × persistent/volatile. Configuration parameters are mapped to mutable persistent registers.

Mutable persistent registers that affect the operation of the device may take effect after restart or immediately. By default, one should assume the latter unless documented otherwise.

Configuration parameters (i.e., mutable persistent registers) are not committed to the non-volatile memory until the device is commanded to restart.

This document purposefully omits the description of some registers. These are not intended for production use and should not be relied upon; their name, type, contents, and semantics may change arbitrarily between minor releases with no regard for compatibility.

5.2 Configuration parameter index

Name	Type	Unit	Default	Pages	Description
`uavcan.node.id`	`natural16`		65535	7	Node ID of the local Cyphal node; 65535 enables PnP
`uavcan.node.description`	`string`		“” (empty)		Arbitrary user-provided description of the node
`uavcan.can.mtu`	`natural8`	byte	8	12	Cyphal/CAN maximum transmission unit, either 8 or 64
`uavcan.can.bitrate`	`natural16[2]`	bps	1000000, 1000000	12	Arbitration and data phase bitrates
`uavcan.can.count`	`natural8`			12	Number of Cyphal/CAN interfaces to use
`uavcan.pub.feedback.id`	`natural16`		65535		`feedback` publication topic-ID
`uavcan.pub.feedback.prio`	`natural8`		4		`feedback` publication priority
`uavcan.pub.flux_density.id`	`natural16`		65535		`flux_density` publication topic-ID
`uavcan.pub.flux_density.prio`	`natural8`		5		`flux_density` publication priority
`uavcan.pub.input_voltage.id`	`natural16`		65535		`input_voltage` publication topic-ID
`uavcan.pub.input_voltage.prio`	`natural8`		5		`input_voltage` publication priority
`uavcan.pub.capacitor_voltage.id`	`natural16`		65535		`capacitor_voltage` publication topic-ID
`uavcan.pub.capacitor_voltage.prio`	`natural8`		5		`capacitor_voltage` publication priority
`uavcan.pub.temperature.id`	`natural16`		65535		`temperature` publication topic-ID
`uavcan.pub.temperature.prio`	`natural8`		6		`temperature` publication priority
`uavcan.sub.command.id`	`natural16`		65535		`command` subscription topic-ID
`analog.mode`	`string`		“” (empty)	13	Analog interface mode

Table 5.1: Configuration parameter index

6 Embedded bootloader

TBD

7 Commissioning

TBD