

PROFINET SET UP GUIDE

Riccardo Piccinini
Mini Motor S.P.A.

FBS-EPN-EN
Firmware Revision: 5.006

Version 6, 10-2026

Table of Contents

1. DS402 Getting Started 1
- 1.1. What is DS402 1
- 1.2. The state machine 3
- 1.3. Enabling the motor in code 5
- 1.4. Profile Velocity 6
- 1.5. Profile Position 7
- 1.6. Homing 8
- 1.7. From here on 9
- 1.8. PROFINET specifics: fixed configuration and acyclic messaging 9

2. PROFINET 10
- 2.1. Device identity 10
- 2.2. Process Data 10
- 2.3. Saving parameters 17

3. GSDML 18

4. Application Example 20
- 4.1. Tools 20
- 4.2. Overview 20
- 4.3. Basic Application 20
- 4.4. Example Explanation 22

5. PROFIdrive 28
- 5.1. Drive settings 28
- 5.2. Device configuration 29
- 5.3. Telegrams 29
- 5.4. Using the Siemens blocks 32

1. DS402 Getting Started

This chapter is a hands-on introduction to controlling a Mini Motor servo drive over a fieldbus using the **CANopen DS402** (CiA 402) device profile. It explains the protocol, walks through the drive state machine and shows, with ready-to-read code, how to enable the motor and command it in *Profile Velocity* and *Profile Position*. Once these concepts are clear, the vendor-specific Application Example that follows will be much easier to read: it is the same logic applied to a concrete PLC.

1.1. What is DS402

CiA 402 (also known as **DS402** / **DSP402**, standardised as *IEC 61800-7-201/301*) is the device profile for drives and motion control. It defines, independently from the underlying fieldbus:

- an **object dictionary**: every quantity exchanged with the drive (commands, references, state, feedback) is an object addressed by a 16-bit hexadecimal index, e.g. `6040h`;
- a **state machine** that governs when power is applied to the motor;
- a set of **modes of operation** (Profile Position, Profile Velocity, Profile Torque, Homing, …);
- the meaning of the two central objects, **Control Word** (`6040h`) and **Status Word** (`6041h`).

The value of the profile is interoperability: the same objects and the same control logic apply on **CANopen**, **EtherCAT**, **EtherNet/IP**, **PROFINET** and **POWERLINK**. Only the transport layer changes; the code you write to talk to the motor stays the same. Modbus is the only exception — it does not use DS402 but a set of dedicated registers.

1.1.1. The master/slave dialogue

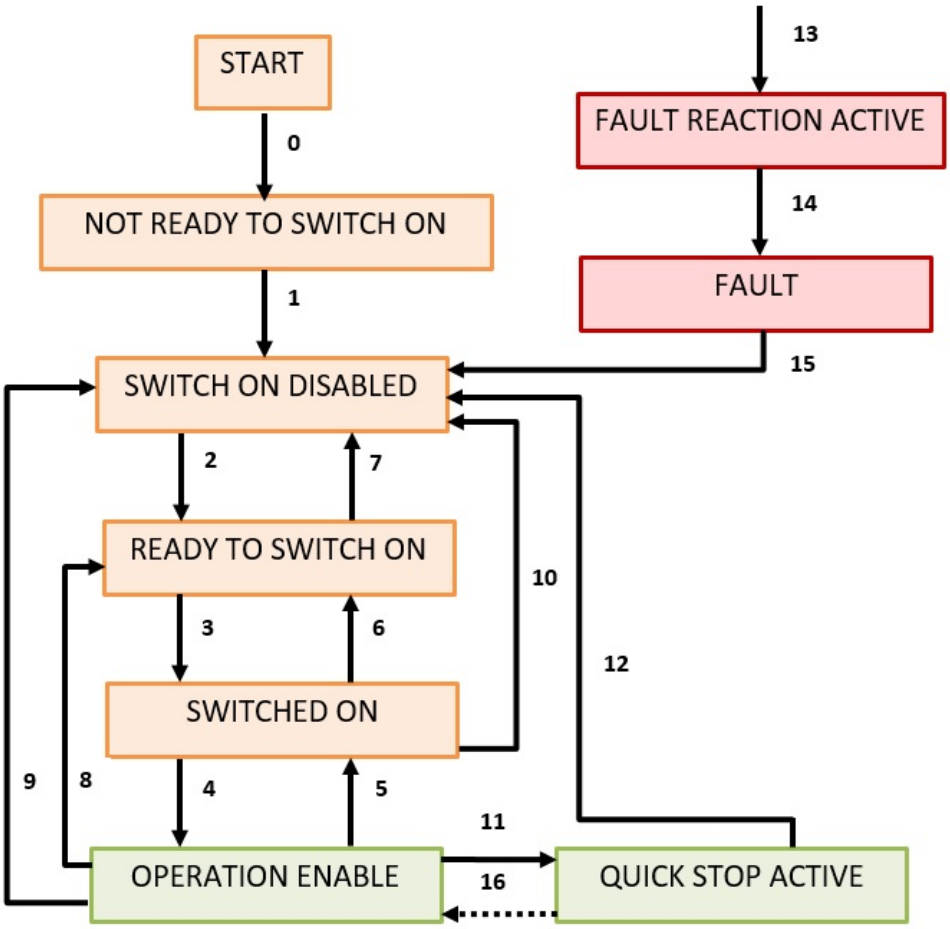
Whatever the mode, the interaction between the master (the controller) and the slave (the servo drive) always follows the same pattern:

1. The master selects the **mode of operation** by writing object *Modes of Operation* (`6060h`); the drive confirms it in *Modes of Operation Display* (`6061h`).
2. The master brings the drive to the *Operation Enabled* state by writing the **Control Word** (`6040h`), following the state machine described below.
3. The master writes the **reference** for the active mode (target velocity `60FFh`, target position `607Ah`, target torque `6071h`, …) and, where required, triggers the motion.
4. The master monitors the drive through the **Status Word** (`6041h`) and the actual-value objects (position `6064h`, velocity `606Ch`, …).

On cyclic fieldbuses these objects are mapped into process data (PDO) and exchanged every cycle; on CANopen they are also reachable via SDO. From the application point of view, you are always reading and writing the same objects.

1.2. The state machine

The motor produces torque only in the **Operation Enabled** state. To get there, the master walks the DS402 finite state machine by writing the Control Word and observing the Status Word.



States in **orange** have power disabled on the motor, states in **green** have power enabled, states in **red** are alarm states.

State	Meaning
Switch On Disabled	Drive ready to be prepared; parameters can be transferred, power can be enabled.
Ready To Switch On	Parameters can be transferred, power can be enabled, the motor cannot run yet.
Switched On	Same as above, one transition away from running.
Operation Enabled	No fault present, motor functions and power enabled: the motor obeys the active mode.
Quick Stop Active	The drive stopped on the emergency ramp; power still enabled, functions disabled.
Fault	A fault is active, the drive is stopped and disabled.

1.2.1. Control Word and Status Word

The transitions are driven by a few bit patterns of the **Control Word** (`6040h`). In practice you only need these commands:

Command	Control Word	Resulting transition
Shutdown	`16#0006`	to *Ready To Switch On*
Switch On	`16#0007`	to *Switched On*
Enable Operation	`16#000F`	to *Operation Enabled* (motor runs)
Disable Voltage	`16#0000`	to *Switch On Disabled*
Quick Stop	`16#0002`	to *Quick Stop Active*
Fault Reset	`16#0080`	clears a fault (acts on the rising edge of bit 7)

The current state is read back from the **Status Word** (`6041h`). By masking the low bits you can tell exactly where the drive is:

Status Word (binary)	State
`xxxx xxxx x1xx 0000`	Switch On Disabled
`xxxx xxxx x01x 0001`	Ready To Switch On
`xxxx xxxx x01x 0011`	Switched On
`xxxx xxxx x01x 0111`	Operation Enabled
`xxxx xxxx x00x 0111`	Quick Stop Active
`xxxx xxxx x0xx 1000`	Fault

Two more Status Word bits are used constantly in the examples below:

- **bit 10 — Target Reached**: set when the drive has reached the commanded position/velocity.
- **bit 12 — Set Point Acknowledge** (Profile Position): the drive acknowledges that it has latched a new position setpoint.

1.3. Enabling the motor in code

The following Structured Text (IEC 61131-3) snippets are vendor-neutral: they use `M1_Cw` for the Control Word written to the motor, `M1_Sw` for the Status Word read from it, `M1_ModeOfOp` / `M1_ModeOfOpDisplay` for objects `6060h` /`6061h`, and the target/actual objects by name. Map these to the tags of your own PLC.

The recommended pattern is a small state machine that reproduces the DS402 transitions. Every cycle it looks at the Status Word and issues the next Control Word until *Operation Enabled* is reached:

```
(* --- Bring the drive to Operation Enabled --- *)
IF (M1_Sw AND 16#004F) = 16#0008 THEN
    (* Fault: request a fault reset on the rising edge of bit 7 *)
    M1_Cw := 16#0080;
ELSIF (M1_Sw AND 16#004F) = 16#0040 THEN
    (* Switch On Disabled -> Ready To Switch On *)
    M1_Cw := 16#0006;
ELSIF (M1_Sw AND 16#006F) = 16#0021 THEN
    (* Ready To Switch On -> Switched On *)
    M1_Cw := 16#0007;
ELSIF (M1_Sw AND 16#006F) = 16#0023 THEN
    (* Switched On -> Operation Enabled *)
    M1_Cw := 16#000F;
END_IF;
```

The masks (`16#004F`, `16#006F`) isolate the state bits of the Status Word so each pattern matches exactly one state; the constant on the right is the state you are leaving. Once the Status Word settles on `···x01x 0111` (Operation Enabled) the motor obeys the active mode.



The `Fault Reset` command (`16#0080`) acts on the **rising edge** of bit 7. Make sure the Control Word returns to a value with bit 7 low before requesting another reset, otherwise the edge is lost.

1.4. Profile Velocity

In **Profile Velocity** (mode `3`) the drive keeps a target speed, reaching it through the configured acceleration/deceleration ramps. The reference is *Target Velocity* (`60FFh`), in rpm.

The logic is:

1. select the mode (`M1_ModeOfOp := 3`) and wait for the drive to confirm it (`M1_ModeOfOpDisplay = 3`);
2. write the target velocity;
3. drive the state machine to Operation Enabled (`16#000F`) to make the motor run.

```
(* --- Profile Velocity --- *)
M1_ModeOfOp := 3;                       (* 6060h: Profile Velocity *)
M1_TargetVelocity := GUI_TargetRpm;     (* 60FFh: rpm, signed = direction *)

IF (M1_ModeOfOpDisplay = 3) THEN
    M1_Cw := 16#000F;                   (* Enable Operation: the motor runs *)
ELSE
    M1_Cw := 16#0006;                   (* mode not confirmed yet: stay ready *)
END_IF;
```

The sign of `M1_TargetVelocity` sets the direction. Writing a new value changes the speed on the fly; there is no start trigger to send, unlike Profile Position. To stop, either write `0` or drop the Control Word back to `16#0006` (Shutdown).

1.5. Profile Position

In **Profile Position** (mode `1`) the drive builds a trapezoidal trajectory from the current position to *Target Position* (`607Ah`), using *Profile Velocity* (`6081h`) and the acceleration/deceleration objects. Unlike Profile Velocity, a target only becomes active when the master toggles the **New Setpoint** bit (bit 4) of the Control Word; the drive confirms it with **Set Point Acknowledge** (bit 12 of the Status Word). This handshake lets you load a new target at any time without the drive picking it up prematurely.

The sequence for a single move is:

1. select the mode (`M1_ModeOfOp := 1`) and wait for confirmation (`M1_ModeOfOpDisplay = 1`);
2. write target position and profile velocity;
3. hold the drive in Operation Enabled (`16#000F`);
4. pulse bit 4 (`16#001F`) to latch the target — the drive raises Status Word bit 12;
5. clear bit 4 (back to `16#000F`) so the handshake can be repeated for the next move.

```
(* --- Profile Position (single move with New Setpoint handshake) --- *)
M1_ModeOfOp := 1;                          (* 6060h: Profile Position *)
M1_TargetPos := GUI_TargetPos;             (* 607Ah *)
M1_ProfileVel := GUI_ProfileRpm;           (* 6081h, rpm *)

IF (M1_ModeOfOpDisplay = 1) THEN
    IF NewSetReq THEN
        (* raise New Setpoint (bit 4) to latch the target *)
        M1_Cw := 16#001F;
        (* the drive acknowledges with Status Word bit 12 (Set Point Ack) *)
        IF (M1_Sw AND 16#1000) = 16#1000 THEN
            M1_Cw := 16#000F;              (* drop bit 4: handshake done *)
            NewSetReq := FALSE;
        END_IF;
    ELSE
        M1_Cw := 16#000F;                  (* Operation Enabled, no new target *)
    END_IF;
ELSE
    M1_Cw := 16#0006;
END_IF;

(* the move is complete when Target Reached (bit 10) is set *)
Arrived := (M1_Sw AND 16#0400) = 16#0400;
```

A few notes on the handshake:

- Set `NewSetReq` (from your logic or the HMI) whenever you want a new positioning to start; the code above turns it into the required bit-4 pulse.
- **Set Point Acknowledge** (bit 12) tells you the drive latched the target. Only then is it safe to clear bit 4.
- **Target Reached** (bit 10) tells you the motion is finished.
- Bit 6 of the Control Word selects **absolute** (0) or **relative** (1) positioning; the examples use absolute targets.

For continuous or alternating moves (e.g. shuttling between two positions), keep bit 4 low between moves and pulse it again for each new target, waiting for bit 12 each time.

1.6. Homing

Homing (mode `6`) establishes the absolute position reference of the axis. The drive runs the procedure selected by *Homing Method* (`6098h`); the search speeds and acceleration are set by `6099h` and `609Ah`. The Mini Motor implementation supports several methods (limit/home switches, index, and the simple "set current position as zero"); see the Homing chapter of the product manual for the full list.

Like Profile Position, homing is triggered by the **New Setpoint / Homing Start** bit (bit 4) of the Control Word, but here the completion is reported by **Homing Done** — Status Word **bit 10** — while a failed homing is flagged by **Homing Error** — Status Word **bit 13**.

The sequence is:

1. select the mode (`M1_ModeOfOp := 6`) and the homing method, then wait for confirmation (`M1_ModeOfOpDisplay = 6`);
2. hold the drive in Operation Enabled (`16#000F`);
3. pulse bit 4 (`16#001F`) to start homing;
4. wait for Homing Done (bit 10), then clear bit 4.

```
(* --- Homing --- *)
M1_ModeOfOp := 6;                       (* 6060h: Homing *)
M1_HomingMethod := 37;                  (* 6098h, e.g. 37 = set current position as zero *)

IF (M1_ModeOfOpDisplay = 6) THEN
    IF HomeReq THEN
        M1_Cw := 16#001F;               (* Homing Start (bit 4) *)
    ELSE
        M1_Cw := 16#000F;               (* Operation Enabled, idle *)
    END_IF;
ELSE
    M1_Cw := 16#0006;
END_IF;

HomingDone  := (M1_Sw AND 16#0400) = 16#0400;   (* bit 10 *)
HomingError := (M1_Sw AND 16#2000) = 16#2000;   (* bit 13 *)

IF HomingDone THEN
    HomeReq := FALSE;                   (* drop bit 4: homing complete *)
END_IF;
```

Method `37` (set the current position as zero) needs no motion and completes immediately; the switch/index-based methods make the motor move to search for the reference and only then raise Homing Done. Always check **Homing Error** (bit 13) so your logic can react to a homing that could not complete.

1.7. From here on

The rest of this guide applies exactly this logic to a specific controller and development environment: installing the device description, mapping the process data, and the ready-made example project. When you read the vendor code, look for the same three ingredients — the state-machine walk to *Operation Enabled*, the mode selection, and the reference plus (for Profile Position) the New Setpoint handshake.

1.8. PROFINET specifics: fixed configuration and acyclic messaging

On PROFINET the DS402 objects are exchanged through the cyclic IO data defined by the GSD. Two points make it different from CANopen/EtherCAT and are worth keeping in mind while reading the code above:

- **The cyclic process data is not freely composable.** You cannot assemble a custom PDO object by object; you must select one of the five **fixed configurations** — **Default (DFL)**, **A**, **B**, **C** and **D** — described in the *Process Data* section further down. *Default* carries the standard Profile Velocity/Torque/Position/Homing data; *A* adds digital/analog inputs and fieldbus-commanded digital outputs; *B* adds the CiA 402 touch probe; *C* the manufacturer touch probe; *D* the three-axis RMS acceleration values.
- **There is no SDO channel.** Objects that are not part of the cyclic data cannot be reached with an SDO as on CANopen. To read or write them you use **acyclic messaging**, described later in the application example.

2. PROFINET



Note: available only for drivers equipped with PROFINET optional board (---/---/EPN).

This document outlines the details of the PROFINET implementation for Minimotor DBS55 integrated servomotor. By enabling PROFINET/DSP402 “reference mode”, the servodrive is commanded by means of CANopen DSP402 state machine and profiles.

For further information, please refer to:

- DBS55 user manual: information about parametrization through USB and PC interface
- Dsp402 v3: information about device state machine, control word/status word encoding and object dictionary.

2.1. Device identity

- Vendor_ID: 0x0427
- Device_ID: 0x0001

2.2. Process Data

There are five possible configurations of the PROFINET process data.

- **Default(DFL)**: the standard process data implementing the profile velocity/torque/position/homing
- **A**: Standard process data with the possibility to read Digital and Analog inputs and command via fieldbus the Digital Out. Parameter P015 must be on 10-Fieldbus.
- **B**: A process data with the CiA 402 Touch probe functionality
- **C**: A process data with our manufacturer custom Touch Probe implementation
- **D**: A process data with rms acceleration values of the three axis

You must select the corresponding device description and set the P176 parameter to the desired value.

2.2.1. Layout Default (DFL)

The configuration for process data is the following:

PLC Outputs (42 Bytes)

Offset	Type	Attribute	CIA402 Object	Default	Meaning
0x00	U16	ControlWord	0x6040	0000h	see DSP402
0x02	S32	TargetPosition	0x607A	+0	position units
0x06	U16	TorqueLimit	0x6072	1000	× 0.1% Trq
0x08	S32	TargetVelocity	0x60FF	+0	rpm
0x0C	S16	TargetTorque	0x6071	+0	× 0.1% Trq
0x0E	U8	ModeOfOperation	0x6060	1	Profile [1]
0x0F	U8	Padding Byte			
0x10	U32	ProfileVelocity	0x6081	+3000	rpm
0x14	U32	ProfileAccel	0x6083	+3000	rpm/s
0x18	U32	ProfileDecel	0x6084	+3000	rpm/s
0x1C	S32	PositionLimit.Neg	0x607D.1	-2147483648	position units
0x20	S32	PositionLimit.Pos	0x607D.2	+2147483647	position units
0x24	S32	HomingOffset	0x607C	+0	position units
0x28	S8	HomingMethod	0x6098	37	SetQuota [1]
0x29	U8	Padding Byte			

PLC Inputs (18 Bytes)

Offset	Type	Attribute	CIA402 Object	Default	Meaning
0x00	U16	StatusWord	0x6041	0250h	Motor Status [1]
0x02	S32	PositionActualValue	0x6064	+0	position units
0x06	S16	TorqueActualValue	0x6077	+0	× 0.1% Trq
0x08	S32	VelocityActualValue	0x606C	+0	rpm
0x0C	U16	VDclink	0x3010	+0	× 0.1 V
0x0E	U8	ModeOfOperation	0x6060	1	Profile [1]
0x0F	U8	Padding Byte			
0x10	U16	ActualAlarm	0x3008	0	see DBS Manual

2.2.2. Layout A

The configuration for process data is the following:

PLC outputs (44 Bytes)

Offset	Type	Attribute	CIA402 Object	Default	Meaning
0x00	U16	ControlWord	0x6040	0000h	see DSP402
0x02	S32	TargetPosition	0x607A	+0	position units
0x06	U16	TorqueLimit	0x6072	1000	× 0.1% Trq
0x08	S32	TargetVelocity	0x60FF	+0	rpm
0x0C	S16	TargetTorque	0x6071	+0	× 0.1% Trq
0x0E	U8	ModeOfOperation	0x6060	1	Profile [1]
0x0F	U8	Padding Byte			
0x10	U32	ProfileVelocity	0x6081	+3000	rpm
0x14	U32	ProfileAccel	0x6083	+3000	rpm/s
0x18	U32	ProfileDecel	0x6084	+3000	rpm/s
0x1C	S32	PositionLimit.Neg	0x607D.1	-2147483648	position units
0x20	S32	PositionLimit.Pos	0x607D.2	+2147483647	position units
0x24	S32	HomingOffset	0x607C	+0	position units
0x28	S8	HomingMethod	0x6098	37	SetQuota [1]
0x29	U8	Padding Byte			
0x2A	U16	Digital output	0x3002	0	bit0: DOUT1 bit1..15: not used [2]

PLC inputs (22 bytes)

Offset	Type	Attribute	CIA402 Object	Default	Meaning
0x00	U16	StatusWord	0x6041	0250h	Motor Status [1]
0x02	S32	PositionActualValue	0x6064	+0	position units
0x06	S16	TorqueActualValue	0x6077	+0	× 0.1% Trq
0x08	S32	VelocityActualValue	0x606C	+0	rpm
0x0C	U16	VDclink	0x3010	+0	× 0.1 V
0x0E	U8	ModeOfOperation	0x6060	1	Profile [1]
0x0F	U8	Padding Byte			

Offset	Type	Attribute	CIA402 Object	Default	Meaning
0x10	U16	ActualAlarm	0x3008	0	see DBS Manual
0x12	S16	Analog Input	0x3000	0	-32768 … +32767
0x14	U16	Digital Input	0x3001	0	Bit0: DIN1 Bit1: DIN2 Bit2: DIN3 Bit3: DIN4 Bit4: DIN5 Bit5..15: not used

2.2.3. Layout B

The configuration for process data is the following:

PLC outputs (46 bytes)

Offset	Type	Attribute	CIA402 Object	Default	Meaning
0x00	U16	ControlWord	0x6040	0000h	see DSP402
0x02	S32	TargetPosition	0x607A	+0	position units
0x06	U16	TorqueLimit	0x6072	1000	× 0.1% Trq
0x08	S32	TargetVelocity	0x60FF	+0	rpm
0x0C	S16	TargetTorque	0x6071	+0	× 0.1% Trq
0x0E	U8	ModeOfOperation	0x6060	1	Profile [1]
0x0F	U8	Padding Byte			
0x10	U32	ProfileVelocity	0x6081	+3000	rpm
0x14	U32	ProfileAccel	0x6083	+3000	rpm/s
0x18	U32	ProfileDecel	0x6084	+3000	rpm/s
0x1C	S32	PositionLimit.Neg	0x607D.1	-2147483648	position units
0x20	S32	PositionLimit.Pos	0x607D.2	+2147483647	position units
0x24	S32	HomingOffset	0x607C	+0	position units
0x28	S8	HomingMethod	0x6098	37	SetQuota [1]
0x29	U8	Padding Byte			
0x2A	U16	Digital output	0x3002	0	bit0: DOUT1 bit1..15: not used [2]
0x2C	U16	Touch Probe Function	0x60B8	0	[1]

PLC inputs (32 bytes)

Offset	Type	Attribute	CIA402 Object	Default	Meaning
0x00	U16	StatusWord	0x6041	0250h	Motor Status [1]
0x02	S32	PositionActualValue	0x6064	+0	position units
0x06	S16	TorqueActualValue	0x6077	+0	× 0.1% Trq
0x08	S32	VelocityActualValue	0x606C	+0	rpm
0x0C	U16	VDclink	0x3010	+0	× 0.1 V
0x0E	U8	ModeOfOperation	0x6060	1	Profile [1]
0x0F	U8	Padding Byte			
0x10	U16	ActualAlarm	0x3008	0	see DBS Manual
0x12	S16	Analog Input	0x3000	0	-32768 … +32767
0x14	U16	Digital Input	0x3001	0	Bit0: DIN1 Bit1: DIN2 Bit2: DIN3 Bit3: DIN4 Bit4: DIN5 Bit5..15: not used
0x16	U16	Touch Probe Status	0x60B9	0	[1]
0x18	S32	Touch Probe Rising Edge position	0x60BA	0	position units
0x1C	S32	Touch Probe Falling Edge position	0x60BB	0	position units

2.2.4. Layout C

The configuration for process data is the following:

PLC outputs (52 Bytes)

Offset	Type	Attribute	CIA402 Object	Default	Meaning
0x00	U16	ControlWord	0x6040	0000h	see DSP402
0x02	S32	TargetPosition	0x607A	+0	position units
0x06	U16	TorqueLimit	0x6072	1000	0.1% Trq
0x08	S32	TargetVelocity	0x60FF	+0	rpm
0x0C	S16	TargetTorque	0x6071	+0	0.1% Trq
0x0E	U8	ModeOfOperation	0x6060	1	Profile [1]
0x0F	U8	Padding Byte			
0x10	U32	ProfileVelocity	0x6081	+3000	rpm
0x14	U32	ProfileAccel	0x6083	+3000	rpm/s

Offset	Type	Attribute	CIA402 Object	Default	Meaning
0x18	U32	ProfileDecel	0x6084	+3000	rpm/s
0x1C	S32	PositionLimit.Neg	0x607D.1	-2147483648	position units
0x20	S32	PositionLimit.Pos	0x607D.2	+2147483647	position units
0x24	S32	HomingOffset	0x607C	+0	position units
0x28	S8	HomingMethod	0x6098	37	SetQuota [1]
0x29	U8	Padding Byte			
0x2A	U16	Digital output	0x3002	0	bit0: DOUT1 bit1..15: not used [2]
0x2C	S32	Touch Probe positioning offset	0x3020	0	Position Units
0x2C	S32	Touch Probe positioning modulo	0x3021	0	Position Units

PLC inputs (22 bytes)

Offset	Type	Attribute	CIA402 Object	Default	Meaning
0x00	U16	StatusWord	0x6041	0250h	Motor Status [1]
0x02	S32	PositionActualValue	0x6064	+0	position units
0x06	S16	TorqueActualValue	0x6077	+0	× 0.1% Trq
0x08	S32	VelocityActualValue	0x606C	+0	rpm
0x0C	U16	VDclink	0x3010	+0	× 0.1 V
0x0E	U8	ModeOfOperation	0x6060	1	Profile [1]
0x0F	U8	Padding Byte			
0x10	U16	ActualAlarm	0x3008	0	see DBS Manual
0x12	S16	Analog Input	0x3000	0	-32768 … +32767
0x14	U16	Digital Input	0x3001	0	Bit0: DIN1 Bit1: DIN2 Bit2: DIN3 Bit3: DIN4 Bit4: DIN5 Bit5..15: not used

2.2.5. Layout D

The configuration for process data is the following:

PLC outputs (44 Bytes)

Offset	Type	Attribute	CIA402 Object	Default	Meaning
0x00	U16	ControlWord	0x6040	0000h	see DSP402
0x02	S32	TargetPosition	0x607A	+0	position units
0x06	U16	TorqueLimit	0x6072	1000	× 0.1% Trq
0x08	S32	TargetVelocity	0x60FF	+0	rpm
0x0C	S16	TargetTorque	0x6071	+0	× 0.1% Trq
0x0E	U8	ModeOfOperation	0x6060	1	Profile [1]
0x0F	U8	Padding Byte			
0x10	U32	ProfileVelocity	0x6081	+3000	rpm
0x14	U32	ProfileAccel	0x6083	+3000	rpm/s
0x18	U32	ProfileDecel	0x6084	+3000	rpm/s
0x1C	S32	PositionLimit.Neg	0x607D.1	-2147483648	position units
0x20	S32	PositionLimit.Pos	0x607D.2	+2147483647	position units
0x24	S32	HomingOffset	0x607C	+0	position units
0x28	S8	HomingMethod	0x6098	37	SetQuota [1]
0x29	U8	Padding Byte			
0x2A	U16	Digital output	0x3002	0	bit0: DOUT1 bit1..15: not used [2]

PLC inputs (28 Bytes)

Offset	Type	Attribute	CIA402 Object	Default	Meaning
0x00	U16	StatusWord	0x6041	0250h	Motor Status [1]
0x02	S32	PositionActualValue	0x6064	+0	position units
0x06	S16	TorqueActualValue	0x6077	+0	× 0.1% Trq
0x08	S32	VelocityActualValue	0x606C	+0	rpm
0x0C	U16	VDclink	0x3010	+0	× 0.1 V
0x0E	U8	ModeOfOperation	0x6060	1	Profile [1]
0x0F	U8	Padding Byte			
0x10	U16	ActualAlarm	0x3008	0	see DBS Manual
0x12	S16	Analog Input	0x3000	0	-32768 … +32767

Offset	Type	Attribute	CIA402 Object	Default	Meaning
0x14	U16	Digital Input	0x3001	0	Bit0: DIN1 Bit1: DIN2 Bit2: DIN3 Bit3: DIN4 Bit4: DIN5 Bit5..15: not used
0x16	U16	Acceleration rms x	0x3031		
0x19	U16	Acceleration rms y	0x3032		
0x1A	U16	Acceleration rms z	0x3033		

2.3. Saving parameters



Saving parameters to non-volatile memory

An acyclic write to a parameter only alters its RAM image: at the next power-up the drive reloads the values stored previously.

To make a change permanent, perform a 32 bit write of the value `65766173h` (the ASCII string "SAVE") to the *Store parameters* index `1010h`, whose subindex is not addressable.

[1] see DSP402

[2] P015=10 required for fieldbus operation

3. GSDML

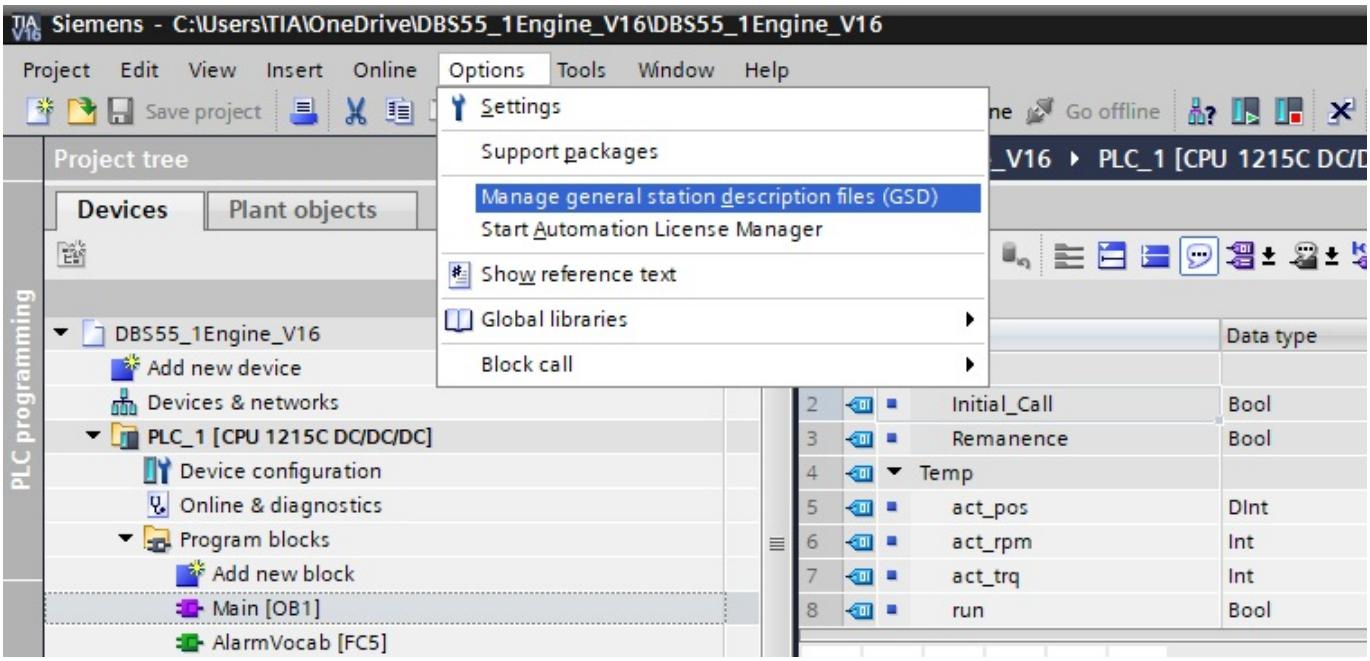
From firmware 5.006 a single GSDML describes the drive for both the DS402 layouts and PROFIdrive. In the BSI package it is in the folder:

```
MM-DBS_vX.XXX\DeviceDesc\Profinet\GSDML-V2.42-MiniMotor-DBS55-<date>.xml
```

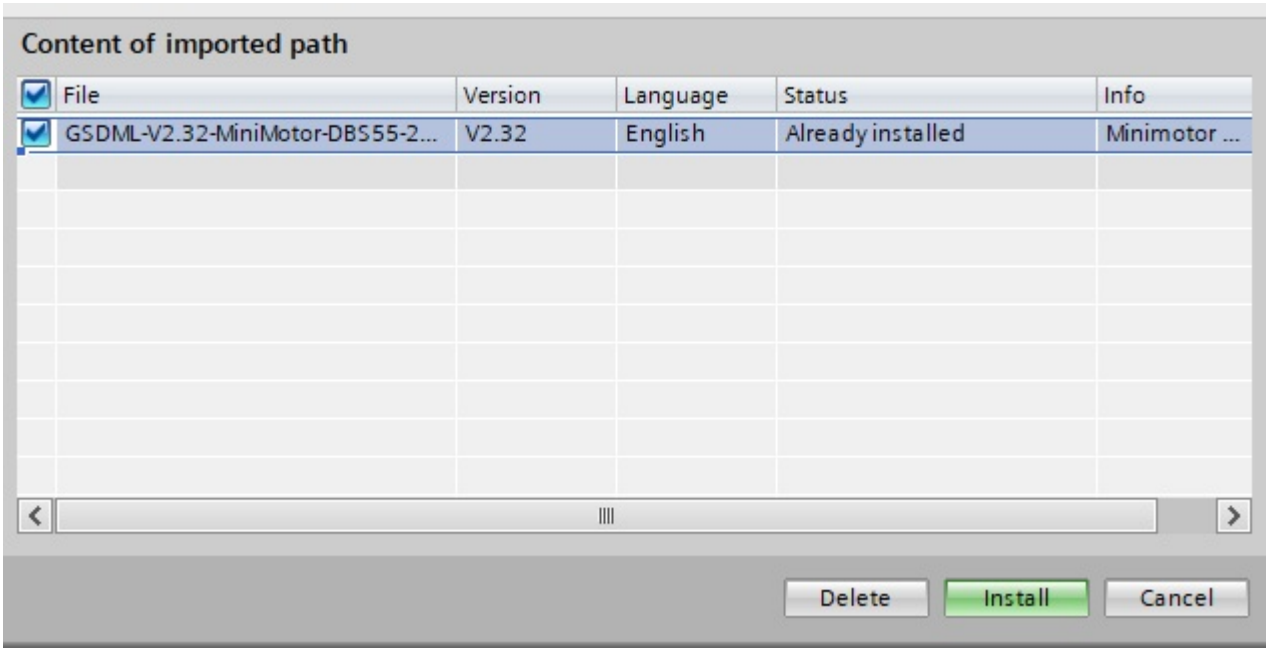
The separate PROFIdrive GSDML of firmware 5.004, in the folder DeviceDesc\Profidrive, is no longer needed.

To install it:

- Go to Options→Manage general station description files (GSD)



- Find the folder MM-DBS_vX.XXX\DeviceDesc\Profinet from the BSI Interface download
- Select the shown GSD and click on Install



In the hardware catalog the device is under the **Drives** family, **DBS55**. The GSDML holds six device access points: choose the one that matches the drive settings.

Table 1. Device access points of the Mini Motor GSDML

Catalog name	P001	P176	Content
DBS55-XXX Default	9	0	DS402 Default layout
DBS55-XXX Extend-A	9	1	Default layout plus analog and digital I/O
DBS55-XXX Extend-B	9	2	Extend-A plus CiA 402 touch probe
DBS55-XXX Extend-C	9	3	Extend-A plus Mini Motor touch probe
DBS55-XXX Extend-D	9	4	Extend-A plus RMS acceleration of the three axes
DBS55-XXX PROFIdrive	18	-	PROFIdrive telegrams 1, 2, 9, 102 and 105, chosen with P217

4. Application Example

4.1. Tools

Tia V20 Example

BSI Interface Software

4.2. Overview

This document shows the creation of a Mini Motor Integrated Drive application using PROFINET communication interface; a Siemens 1215C CPU controller is configured and an application is written into the Tia Portal V20 development environment.

References:

- Siemens 1215C DC/DC/DC CPU
- DBS/FC firmware version 5.004
- Tia Portal V20

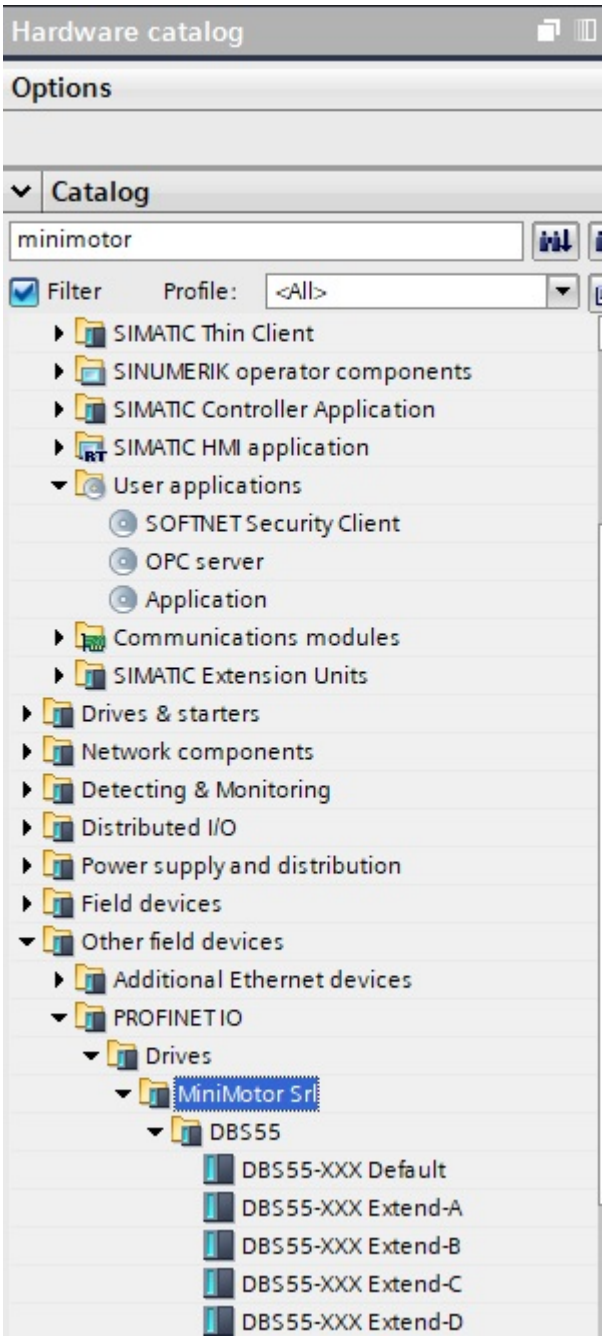
4.3. Basic Application

4.3.1. Drive settings

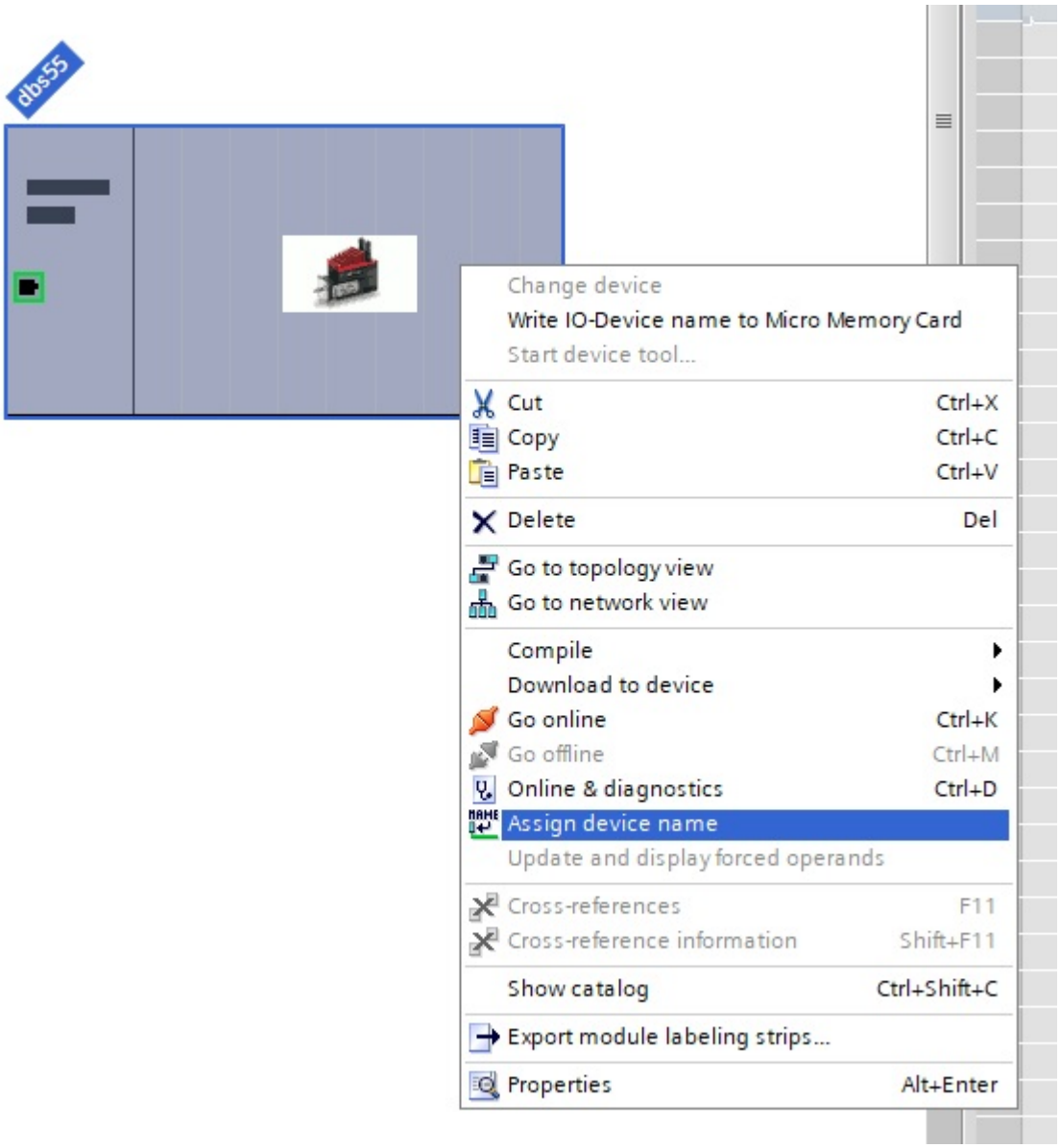
Set P001 MODE to **9** → **Profinet IO** and P176 ProcDataLayout to the layout you want to use (0 for the Default layout of the example). A reboot is needed after the change of mode.

4.3.2. Add Mini Motor system to the project

- After having installed the GSDML (see GSDML) find in the Hardware catalog the motor



- Import in your project the DS402 device access point with the Process data you want to use, from **DBS55-XXX Default** to **DBS55-XXX Extend-D**. It must match P176. Refer to the first section of this guide for how these process data are formed. For the example, the Default Process Data is used.
- Once imported, remember to assign the device a name.



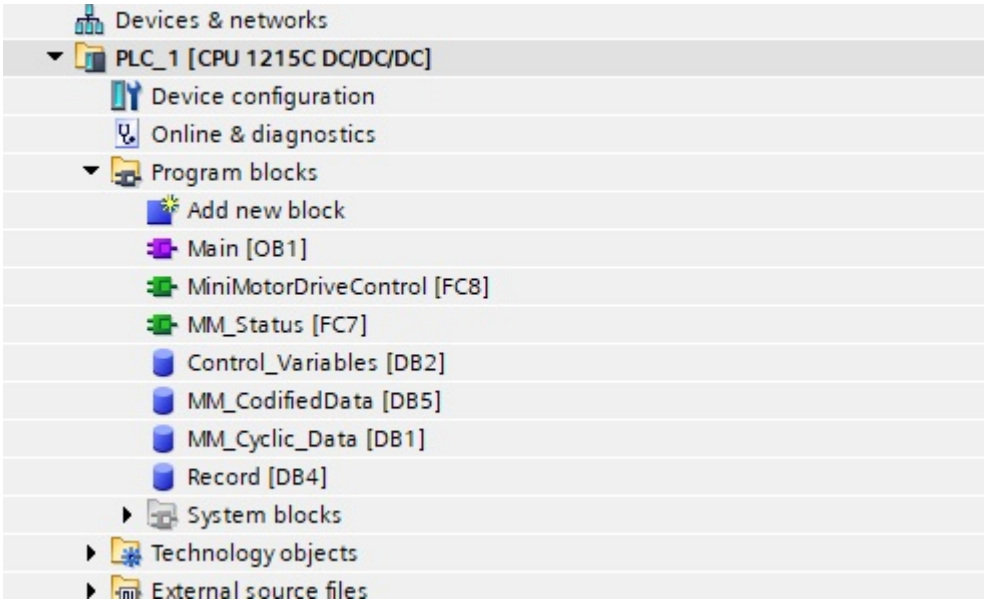
4.4. Example Explanation

Download and open the Tia V20 Example

We will take a look at how it works. It can be commanded via the variables in the MAIN code.

4.4.1. Mini Motor Elements

The Example is composed of 2 Mini Motor made FB and 2 Mini Motor made DB and 2 data types.



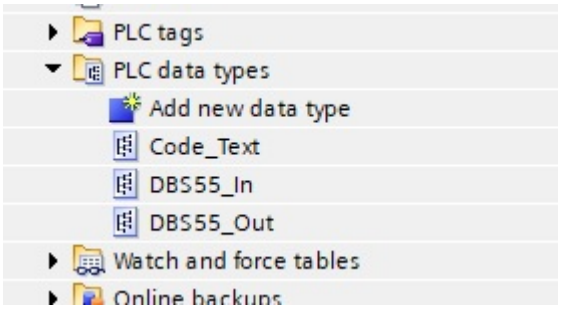
The MiniMotorDriveControl FC is responsible for all motor control. The MM_Status is a block to show readily available information at all times from the specific unit.

The MM_CodifiedData DB has string information and codes used in the Status block to give a text feedback on the motor status based on the input data.

The MM_Syclic_Data has a copy of the cyclic data coming from the motor area of memory, for local manipulation.

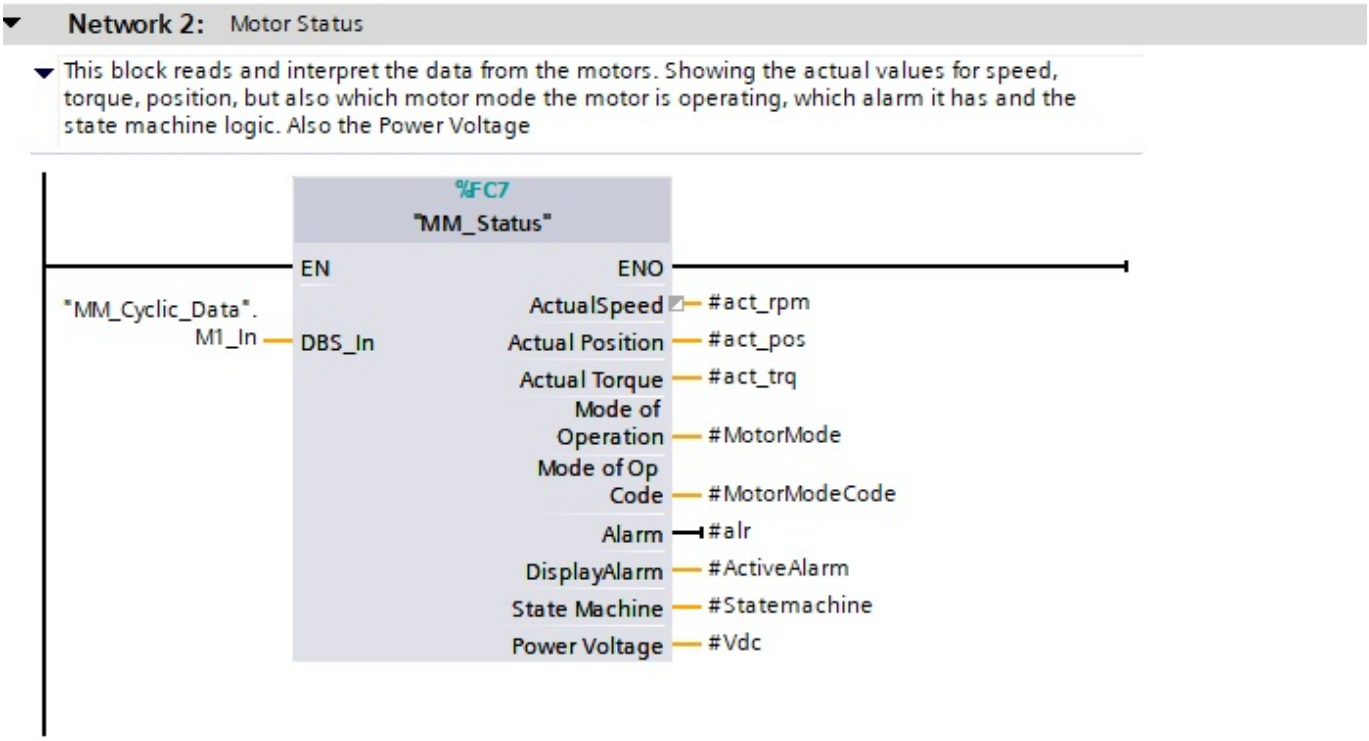
The other two DBS, Control_Variables and Record, are used and/or created automatically to support Siemens blocks or to support the example commands to be changed in real time.

The Data types are just the Motor Cyclic data packaged in a neat format.



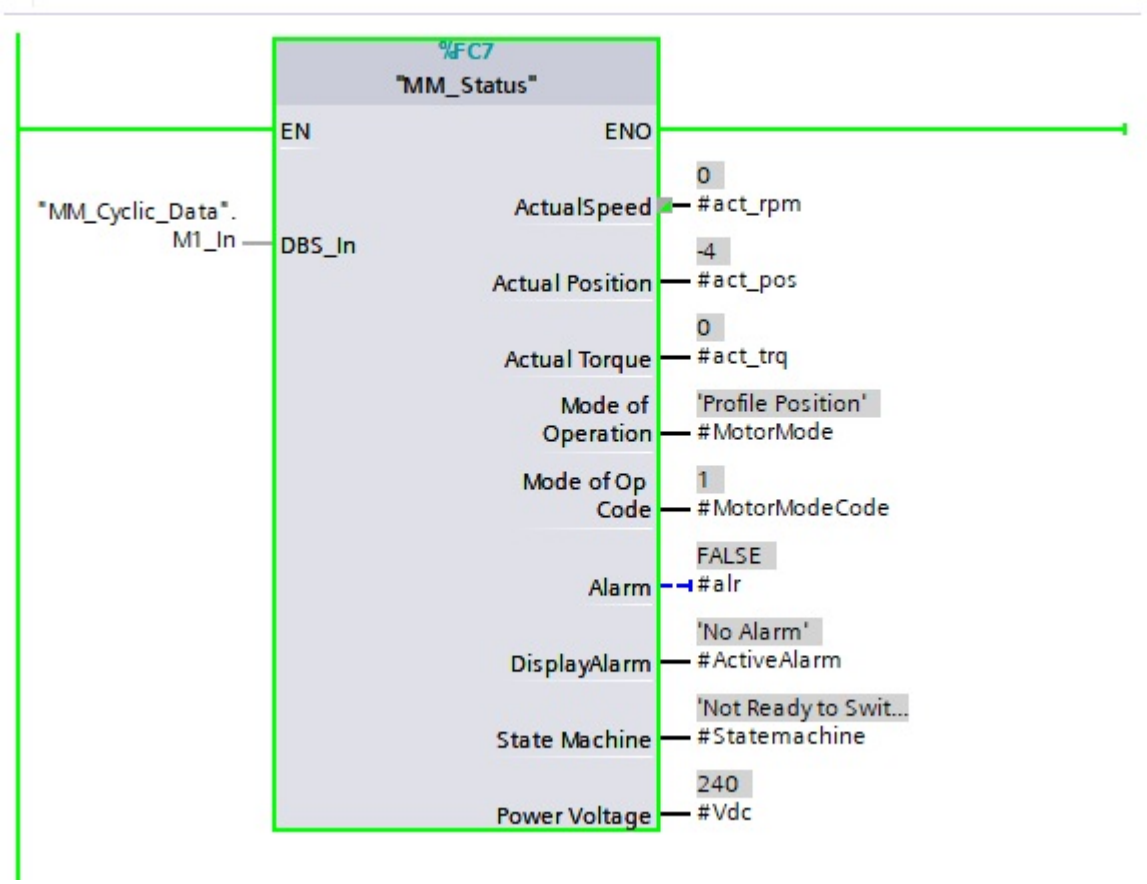
4.4.2. FC Explanations

MM_Status



The MM_Status FC takes as an input the MM_Cyclic_Data.M1_in, which is a local copy of the Motor image sent to the PLC, copied in the first network of the Main program.

The Code inside looks at the Status Word, voltage, mode of operation and other data useful for a feedback from the motor.

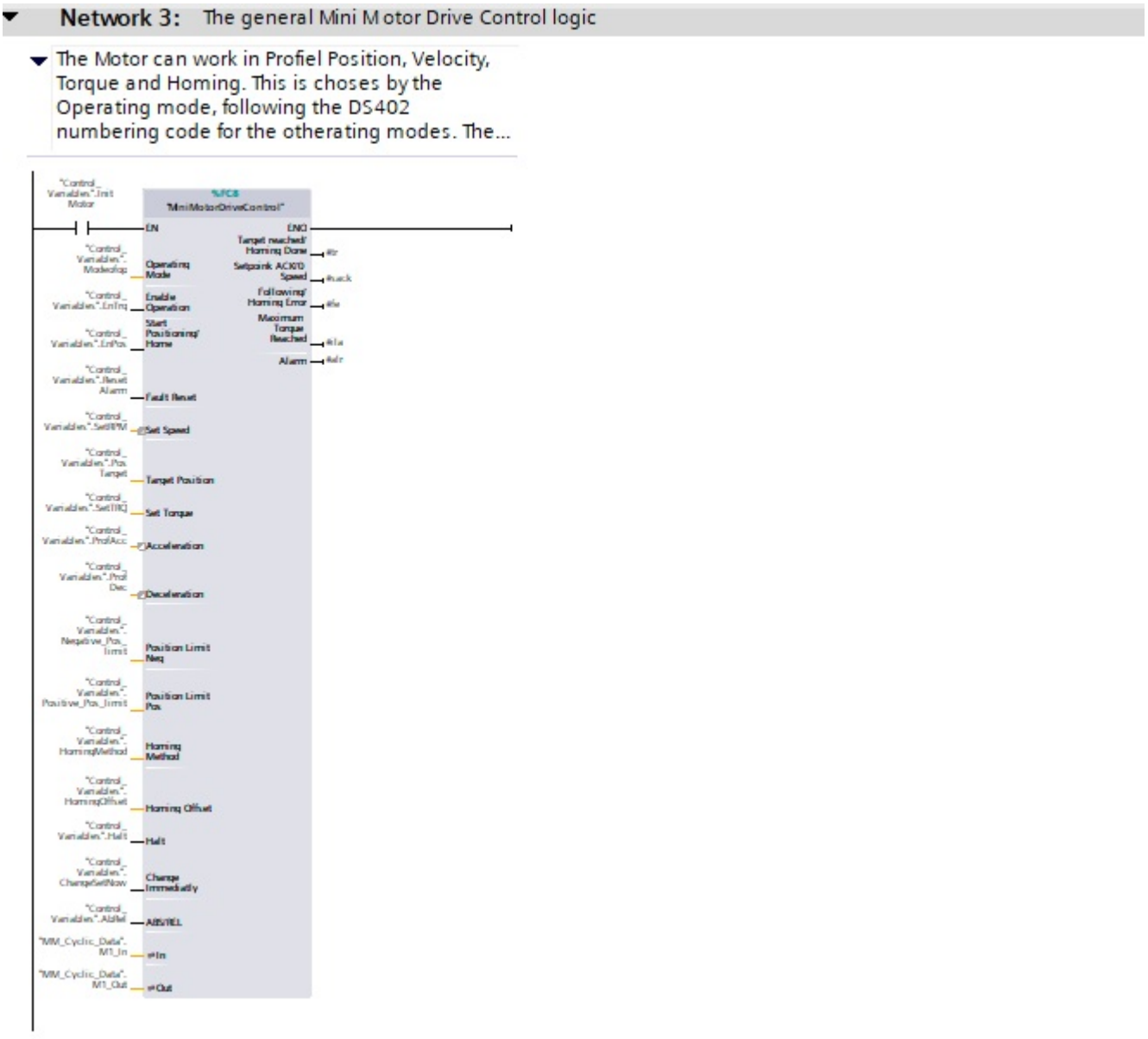


The outputs are:

- Actual Speed: The speed of the motor in rpm
- Actual Position: The position of the motor in encoder pulses, eventually scaled by the user using the motor internal parameters
- Actual Torque: the actual torque used by the motor. In thousands per rated nominal torque, so 1000 = 100% nominal.
- Mode of Operation: The name of the Operating mode the motor is currently in
- Mode of Op Code: The code of the current mode of operation
- Alarm: boolean value if there is or not an error on the drive
- Display Alarm: Text explaining the error
- State Machine: Where the motor is in the DS402 State machine. This is done by reading the Status word bits and the output is in text
- Power Voltage: The voltage reading of the Motor Power Pins.

With this you have a quick look at all the motor variables, that can be used for all kinds of program behaviour.

MiniMotorDriveControl



The MiniMotorDriveControl takes the inputs from the program and navigates the CanOpenDS402 commands automatically across all operating modes of Profile Velocity, Position, Torque and Homing to send the correct commands to the motor.

The block needs the Cyclic Data In and Out to read and write to the motor. The Cyclic data out is then sent out to the motor memory area by the last network in the example.

The Motor has 3 main Inputs.

The input *EN* is used to power the FC and initialize/shutdown the motor. This way the motor will be kept in Ready to Switch On State.

The input *Enable Operation* will put the motor in Enable Operation, making it execute whichever command is given depending on the *modeofop* variable.

- 1: Profile Position
- 3: Profile Velocity
- 4: Profile Torque
- 6: Profile Homing

The input *Start Positioning/Home* is used to start the actual homing or positioning of the motors.

For further information, we refer to the in-depth explanation in the CANopen DS402 specification.

The other inputs set the variable used by the various modes.

Set Speed, *Set Torque*, *Acceleration*, *Deceleration* all contribute to the trajectory, with the velocity, maximum torque that the motor can apply and the acceleration and deceleration, for a trapezoidal profile.

Position Limit Neg and *Position Limit Pos* are used to limit via software the range of movement for the motor. When put at their maximum value they are disabled. These are -2^31 and 2^31-1

Target Position is used to set the target position to be reached by positioning mode when the *Start Positioning/Home* input sees a rising edge.

Change Immediatly and *ABS/REL* are two options for profile positioning. The first one if true it will enable immediate change of target position while the motor is already doing a move, otherwise it will put the new move command in a buffer. The second if true enables relative positioning where the target pos is added to the current position. False is absolute movements.

The *Halt* command PAUSES the current job and when it’s removed the motor will RESUME the previous command.

Home Method defines which homing method the motor will follow (see main manual), and *home offset* is the offset to add to the 0 position.

The outputs of this block are bits that can be used to monitor specific mode of operation behaviour. They are all boolean values self explanatory in their name. Be sure to correctly interpret them depending on the Mode of Operation the motor is in.

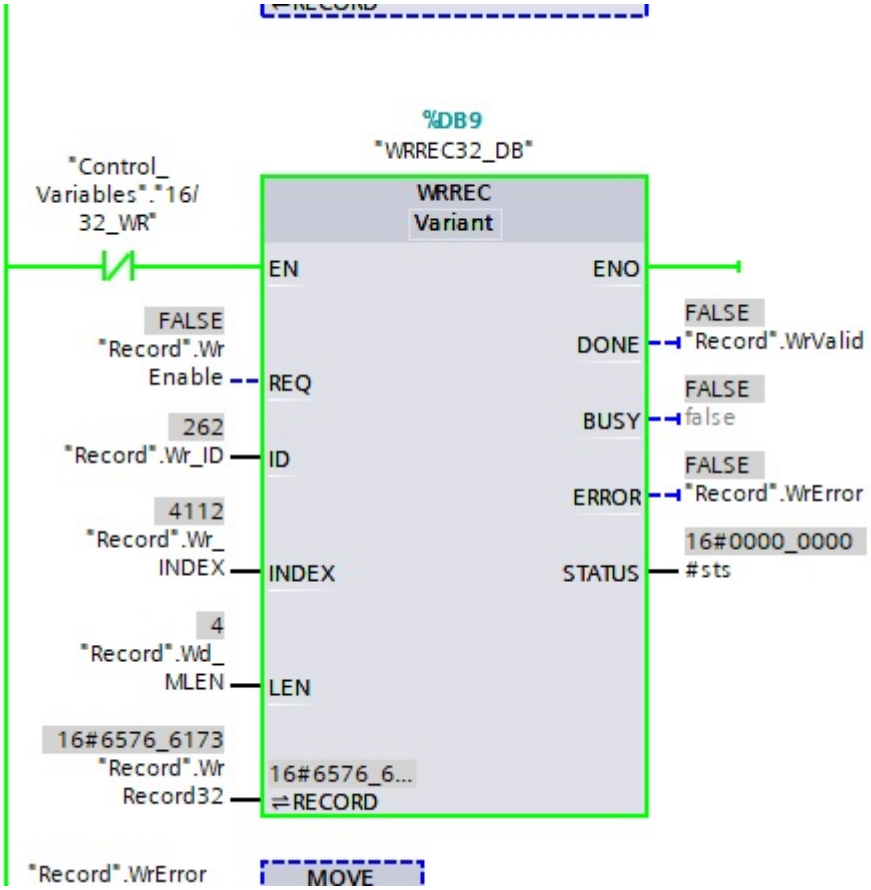
The block also enables the Alarm reset with the command *Fault Reset*

Async Read and Write

All the CANopen (6000h) and Motor Manufacturer (2000h) parameters can be read and written with the RDREC and WRREC by Siemens. For more information about the parameter size, check the main manual.

To save the Manufacturer Parameters (2000h), it can be written the value 65766173h to the index 1010h, like shown in the picture. This command saves all changes, so it can be done last.

Parameters 6000h CANNOT be saved, so they must be loaded up at every startup.



5. PROFIdrive

From firmware 5.004 the Mini Motor Integrated Drive can also be controlled over PROFINET as a **PROFIdrive** drive. PROFIdrive is the drive profile defined by PROFIBUS & PROFINET International (PI): instead of the DS402 objects of the fixed layouts described in the previous chapters, the controller exchanges the **standard telegrams** of the profile, built on control word STW1 and status word ZSW1. It is the natural choice when the PLC program already controls other PROFIdrive drives, or when you want to use the Siemens drive blocks and technology objects.

From firmware 5.006 the drive supports five telegrams, grouped by PROFIdrive application class. The table compares them at a glance: PZD is the number of 16-bit words from the PLC to the drive and back, and the words are listed in the order they appear in the IO data.

Table 2. PROFIdrive telegrams supported by the Mini Motor drive

Telegram	Class	PZD	PLC → drive	Drive → PLC	Typical use
1	AC1, speed	2 / 2	STW1, NSOLL_A (16 bit)	ZSW1, NIST_A (16 bit)	Simple speed control; the Siemens SINA_SPEED block works with it
2	AC1, speed	4 / 4	STW1, NSOLL_B (32 bit), STW2	ZSW1, NIST_B (32 bit), ZSW2	Speed control with finer resolution on setpoint and actual speed
9	AC3, positioning	10 / 5	STW1, SATZANW, STW2, MDI_TARPOS, MDI_VELOCITY, MDI_ACC, MDI_DEC, MDI_MOD	ZSW1, AKTSATZ, ZSW2, XIST_A	Positioning with target, speed and ramps sent by the PLC (MDI)
102	AC4, motion control	6 / 10	STW1, NSOLL_B, STW2, MOMRED, G1_STW	ZSW1, NIST_B, ZSW2, MELDW, G1_ZSW, G1_XIST1, G1_XIST2	Speed axis with the position loop in the controller
105	AC4, motion control with DSC	10 / 10	as telegram 102, plus XERR and KPC	as telegram 102	As telegram 102, with Dynamic Servo Control in isochronous mode



Mini Motor does not provide a PROFIdrive example project yet. This chapter describes the drive settings, the telegrams as defined in the Mini Motor GSDML and the use of the standard Siemens blocks.

5.1. Drive settings

Set the two parameters with the BSI software:

Parameter	Value	Notes
P001 MODE	18 → PROFIdrive	A reboot is needed after the change of mode
P217 Telegram	1 → Telegram 1 2 → Telegram 2 9 → Telegram 9 102 → Telegram 102 105 → Telegram 105	Telegram used at power on: a change takes effect at the next power on

Parameter	Value	Notes
P271 Jog1, P272 Jog2	Speed [rpm], signed	Jog speeds of telegram 9 (STW1 bits 8 and 9)

When the controller configures a different telegram in its project, the drive adapts to it. The telegram actually in use can be read in the PROFIdrive parameter **PNU922**.

Device name and IP address are assigned by the controller as for any PROFINET device, as shown in the Application Example.

5.2. Device configuration

In the hardware catalog choose the **DBS55-XXX PROFIdrive** device access point of the Mini Motor GSDML (see GSDML). The device has one **PROFIdrive Module** in slot 1, with two subslots:

- subslot 1: **Parameter Access Point**, the access point to the drive parameters;
- subslot 2: the **telegram**. The default is **Standard Telegram 1**; to use another one, delete it in the device view and drag the new one from the catalog into the same subslot.

The PROFIdrive access point accepts one controller only (no Shared Device). It supports RT and IRT communication, the MRP client role and a minimum update time of 1 ms.

5.3. Telegrams

Every telegram is a fixed sequence of 16-bit words (PZD), with the structure shown in the table at the beginning of the chapter. The first word is always STW1 from the PLC to the drive and ZSW1 from the drive to the PLC. NSOLL/NIST are speed setpoint and actual speed, XIST and MDI_TARPOS actual and target position, expressed in user units (position scaling P160÷P162).

The speeds are normalized as in the PROFIdrive profile: **16#4000 = 100 %** of the reference speed, which on the Mini Motor drive is **P006 SpeedMax**; negative values reverse the direction. For the 32-bit speeds 100 % is **16#4000_0000**.

5.3.1. Speed telegrams 1 and 2

Table 3. STW1 control word 1 (telegrams 1, 2, 102 and 105)

Bit	Meaning
0	ON (OFF1 when 0)
1	No Coast Stop (OFF2 when 0)
2	No Quick Stop (OFF3 when 0)
3	Enable Operation
4	Enable Ramp Generator
5	Unfreeze Ramp Generator
6	Enable Setpoint

Bit	Meaning
7	Fault Acknowledge (rising edge)
10	Control By PLC

Table 4. ZSW1 status word 1 (telegrams 1, 2, 102 and 105)

Bit	Meaning
0	Ready To Switch On
1	Ready To Operate
2	Operation Enabled
3	Fault Present
4	Coast Stop Not Activated
5	Quick Stop Not Activated
6	Switching On Inhibited
7	Warning Present
8	Speed Error Within Tolerance Range
9	Control Requested
10	Speed Reached Or Exceeded

The switch-on sequence is the standard one of the profile:

1. write **STW1 = 16#047E** (bit 0 at 0, bits 1-6 and 10 at 1) and wait for ZSW1 bit 0 **Ready To Switch On**;
2. write **STW1 = 16#047F**: the drive goes to **Operation Enabled** (ZSW1 bit 2) and follows the speed setpoint;
3. to stop with the ramp write again **16#047E**; bit 2 at 0 stops with the quick stop ramp, bit 1 at 0 releases the motor (coast stop);
4. with a fault ZSW1 bit 3 goes to 1: acknowledge it with a rising edge of STW1 bit 7 (for example 16#04FE, then back to 16#047E).

Bit 10 **Control By PLC** must stay at 1: without it the drive does not accept the commands of the control word.

5.3.2. Position telegram 9

Telegram 9 works in the **MDI** submode (Manual Data Input): the PLC does not select stored motion records, it sends every positioning directly with target, speed, ramps and mode. MDI is selected with bit 15 of **SATZANW** at 1, and the drive confirms it with bit 15 of **AKTSATZ**.

Table 5. MDI words of telegram 9

Word	Type	Meaning
SATZANW	16 bit	Bit 15 = 1: MDI submode (no motion records)
MDI_TARPOS	32 bit	Target position, in user units
MDI_VELOCITY	32 bit	Speed of the positioning, 1 = 1000 user units/min

Word	Type	Meaning
MDI_ACC, MDI_DEC	16 bit	Acceleration and deceleration, 16#4000 = 100 % of the maximum
MDI_MOD	16 bit	Bit 0 = 1 absolute, 0 relative; bits 1 and 2: modulo positioning in positive or negative direction

Table 6. STW1 control word 1, positioning (telegram 9)

Bit	Meaning
0	ON
1	No Coast Stop
2	No Quick Stop
3	Enable Operation
4	Do Not Reject Traversing Task
5	No Intermediate Stop
6	Activate Traversing Task (rising edge)
7	Fault Acknowledge
8	Jog1 ON (speed P271)
9	Jog2 ON (speed P272)
10	Control By PLC
11	Start Homing Procedure

Table 7. ZSW1 status word 1, positioning (telegram 9)

Bit	Meaning
0	Ready To Switch On
1	Ready To Operate
2	Operation Enabled
3	Fault Present
4	Coast Stop Not Activated
5	Quick Stop Not Activated
6	Switching On Inhibited
7	Warning Present
8	Position Error Within Tolerance Range
9	Control Requested
10	Target Position Reached
11	Home Position Set
12	Traversing Task Acknowledgement
13	Drive Stopped

The drive is switched on with the same sequence as the speed telegrams. A positioning is then a traversing task: set SATZANW bit 15, write target, speed, ramps and MDI_MOD, keep bits 4 and 5 of STW1 at 1 and give a rising edge on bit 6. ZSW1 bit 12 acknowledges the task, bit 10 signals that the target has been reached and XIST_A gives the actual position. The homing procedure, with the method set in P152, starts with STW1 bit 11 and ends with ZSW1 bit 11 **Home Position Set**.

5.3.3. Motion control telegrams 102 and 105

With telegrams 102 and 105 the drive works as a speed axis and the position loop runs in the controller: the controller reads the encoder position and sends back the speed setpoint NSOLL_B. These are the telegrams used by the motion control technology objects of the S7-1500 controllers. STW1 and ZSW1 are the same as the speed telegrams.

Table 8. Additional words of telegrams 102 and 105

Word	Type	Meaning
MOMRED	16 bit	Torque reduction
G1_STW	16 bit	Encoder 1 control word
XERR	32 bit	Position deviation, telegram 105 only (DSC)
KPC	32 bit	Position controller gain, telegram 105 only (DSC)
MELDW	16 bit	Message word
G1_ZSW	16 bit	Encoder 1 status word
G1_XIST1	32 bit	Encoder 1 actual position 1, the cyclic position used by the controller
G1_XIST2	32 bit	Encoder 1 actual position 2, the value requested with G1_STW (for example the absolute position or a reference mark)

Telegram 105 adds **Dynamic Servo Control (DSC)**. Besides the speed setpoint, the controller sends the position deviation XERR and the gain KPC, and the drive closes the position loop on them at its own cycle, faster than the controller cycle. DSC needs the isochronous mode.

Both telegrams are meant for the **isochronous mode** on IRT (RT_CLASS_3), even if the GSDML does not make it mandatory. In TIA Portal enable it on the telegram submodule and assign the axis to the isochronous servo organization block. The GSDML allows an application cycle from 1 ms to 8 ms, with minimum input and output times Ti and To of 0.5 ms.

5.4. Using the Siemens blocks

For the speed telegram 1 Siemens provides the **SINA_SPEED** block (FB285) in the **DriveLib** library (Siemens entry 109475044), for S7-1200 and S7-1500 controllers. The block writes STW1 and NSOLL_A and reads ZSW1 and NIST_A, so you do not have to handle the bits by hand:

Input / output	Use with the Mini Motor drive
EnableAxis	Start/stop of the drive (STW1 bit 0)
AckError	Fault acknowledge (STW1 bit 7)
SpeedSp	Speed setpoint in rpm

Input / output	Use with the Mini Motor drive
RefSpeed	Reference speed: the value of P006 SpeedMax set in the drive
ConfigAxis	Default 16#003F: sets STW1 bits 1 to 6 to 1
HWIDSTW, HWIDZSW	Both on the hardware ID of the telegram submodule (subslot 2 of the PROFIdrive Module)
AxisEnabled, Error, ActVelocity	Drive enabled, fault present, actual speed in rpm

The block is designed for SINAMICS drives, but it only uses the words of standard telegram 1, so it works with the Mini Motor drive as well. The Siemens application example *SINAMICS G: Controlling a speed axis with the SINA_SPEED block* (entry 109485727) shows the whole procedure, from the hardware configuration to the block call.

With telegrams 102 and 105 the drive is instead connected to a technology object (speed axis or positioning axis) of an S7-1500 controller, which handles telegram, homing and position loop by itself.

For the full definition of the telegrams and of the drive state machine refer to the PROFIdrive profile specification published by PROFIBUS & PROFINET International (PI).