



# V6 control firmware

## FD1E, FD1xEC, FD2E, FD2xC

### Stepper motor driver control firmware

02/10/2018 first issue  
29/05/2025

## 1. DESCRIPTION

The FD-family drivers represent an innovative range of fully digital dual H-bridge stepper motor drivers, specifically designed to minimize vibrations and optimize power consumption, ensuring maximum motor performance.

The FD1.1E, FD1.1EC, FD1.2EC, FD2.1E, and FD2.1EC models expand interface options by incorporating EtherCAT communication through industrial circular connectors (M8 and M12). The FD2.1AC model supports CANopen communication, while the FD2.1C model utilizes Modbus RTU over RS-485.

Mounted directly on the rear of the motor, FD drivers feature a 12-bit per revolution absolute (single-turn) magnetic encoder for precise motion control.

The V6 firmware introduces advanced functionality beyond the traditional current control method, which maintains a fixed motor current regardless of load (with only current reduction when the motor is idle). It also implements an adaptive torque control method, which calculates the optimal current by measuring the resistive torque applied to the motor. This advanced torque control provides two key benefits:

- **Enhanced Efficiency:** motor and driver operate with higher efficiency, reducing power losses and operating temperatures.
- **Dynamic Torque Control:** if a sudden increase in torque demand exceeds the motor's maximum capacity, unexecuted steps are temporarily accumulated. This allows the motor to slow down and deliver higher torque to overcome obstacles. Once the resistive torque decreases, the motor recovers the accumulated steps without losing positional accuracy. An alarm can be configured to trigger if the following error exceeds a programmable threshold.

FD drivers can be controlled via EtherCAT (CoE), CANopen and Modbus RTU protocols or via standard digital inputs (step/dir, quadrature A/B steps, select/start/stop cycle). Additionally, Modbus RTU communication via RS-232 is available for easy troubleshooting, firmware updates, and parameter reprogramming.

- EtherCAT (CoE, FoE) on FD1E, FD1EC, FD2E, and FD2EC
- CANopen on FD2AC
- Modbus RS-485 on FD2C

- Multipurpose I/O

FD1.1E	FD1.1EC	FD2.1E
2 single-ended DI	4 single-ended DI	4 fast diff. DI
1 pnp DO	2 pnp DO	3 single-ended DI
		3 pnp DO

FD2.1C, FD2.1AC, FD2.1EC

3 single-ended DI  
1 single-ended DI or AI  
2 pnp DO

- Torque Control loop

Adjustable  $I_{MAX}$  (current at maximum torque)  
Adjustable  $I_{MIN}$  (current at no torque)

- 12-bit single-turn absolute encoder on FD1.1E, FD1.1EC, FD1.2EC, FD2.1E

- 14-bit single-turn absolute encoder on FD2.1EC, FD2.1AC, FD2.1C

- 32 programmable cycles, 10 cycles sequences

Cycle or sequence of cycle's selection, start and stop using DI or fieldbus. Configurable speed, acceleration, deceleration, target position with linear, parabolic and s-curve motion profiles. Several movements available.

- Position resolution

Configurable steps per revolution

- Absolute multi-turn position recovery at power on

- firmware and parameters reprogramming via FoE on EtherCAT versions, via SDO block transfer on CANopen versions, via write file on Modbus RTU.

- Protections: over temperature (100 °C), over voltage, under voltage and short circuit alarms

## 2. INDEX

1.	DESCRIPTION .....	1
2.	INDEX .....	2
3.	HARDWARE .....	7
4.	DWLOADER.....	9
4.1.	In-application programming.....	10
4.2.	RAM data upload .....	10
4.3.	User flash program .....	10
4.4.	Data savings .....	11
4.5.	Create binary files.....	11
4.6.	Data modify.....	11
4.7.	Data Exchange.....	16
4.8.	Modbus Data Exchange- Oscilloscope Feature .....	18
5.	CONTROL METHODS .....	19
6.	INPUTS / OUTPUTS .....	20
6.1.	Start/Stop and Input Frequency Signals .....	20
6.2.	Multipurpose Inputs .....	22
6.3.	Aux inputs .....	23
6.4.	Analog input.....	23
6.5.	Outputs .....	24
7.	ALARMS .....	25
7.1.	Steps accumulation limit .....	25
7.2.	Over-Temperature Alarm .....	25
7.3.	Short Circuit Alarm .....	26
7.4.	Over-Voltage Alarm .....	26
7.5.	Data Error .....	26
7.6.	Under Voltage.....	26
7.7.	Encoder error.....	27
7.8.	Alarm Resetting .....	28
8.	RAMP PROFILES .....	29
9.	POWER-ON POSITION RESTORE (QUASI-ABSOLUTE MULTI-TURN).....	30
9.1.	Multi-turn position upload.....	30
9.2.	Multi-turn position corrected by deviation.....	30
9.3.	Position inside the revolution .....	30
10.	ETHERCAT .....	31
10.1.	Introduction .....	31
10.2.	Main specifications.....	32
10.3.	Node address settings .....	32
10.4.	EtherCAT LED indicators .....	33
10.5.	CANopen over EtherCAT (CoE).....	33
10.6.	EtherCAT Slave Controller (ESC) .....	34
10.6.1.	FMMU .....	34

10.6.2.	SyncManagers.....	34
10.7.	EtherCAT functional overview.....	35
10.8.	EtherCAT state machine .....	36
10.9.	SDO.....	38
10.9.1.	SDO download expedited.....	38
10.9.2.	SDO abort.....	39
10.9.3.	SDO download normal .....	40
10.9.4.	SDO upload expedited.....	42
10.9.5.	SDO upload normal .....	43
10.9.6.	SDO information .....	46
10.9.7.	Get OD list.....	47
10.9.8.	Get object description.....	48
10.9.9.	Get entry description .....	49
10.9.10.	Unit types.....	50
10.9.11.	SDO info error .....	51
10.9.12.	Complete access.....	51
10.9.13.	Encapsulated SDO .....	52
10.10.	PDO mapping.....	54
10.11.	Emergency messages .....	55
10.12.	Synchronization .....	56
10.12.1.	Free run.....	56
10.12.2.	SM synchronous .....	56
10.12.3.	Distributed clocks (DC) .....	58
10.13.	Filetransfer over EtherCAT (FoE) .....	59
10.14.	ESC registers.....	61
11.	CANopen .....	70
11.1.	Standards .....	70
11.2.	Communication objects .....	70
11.2.1.	Network Management Objects (NMT).....	71
11.2.2.	Service Data Objects (SDO).....	73
11.2.3.	Download SDO.....	73
11.2.4.	Upload SDO.....	75
11.2.5.	Abort SDO .....	77
11.2.6.	SDO block download .....	78
11.2.7.	Synchronization object (SYNC).....	81
11.2.8.	Emergency object (EMCY).....	81
11.2.9.	Process Data Objects (PDO) .....	81
12.	DSP402 – MOTION CONTROL.....	83
12.1.1.	Modes of operation.....	85
12.1.2.	Profile position (1) .....	86
12.1.3.	Profile velocity (3).....	88
12.1.4.	Homing (6) .....	89
12.1.5.	Interpolated position (7) .....	92

12.1.6.	Cyclic synchronous position (8) .....	94
12.1.7.	Cyclic synchronous velocity (9) .....	95
13.	Object dictionary .....	96
13.1.	CoE communication.....	96
13.1.1.	0x1000, Device type .....	99
13.1.2.	0x1001, Error register .....	100
13.1.3.	0x1003, Pre-defined error.....	100
13.1.4.	0x100C / 0x100D, Guard time and Life time factor .....	100
13.1.5.	0x1010, Store parameters.....	100
13.1.6.	0x1011, Restore default parameters .....	100
13.1.7.	0x1016, Consumer heartbeat time .....	101
13.1.8.	0x1017, Producer heartbeat time .....	101
13.1.9.	0x140x / 0x180x, RPDOx / TPDOx parameters.....	101
13.1.10.	0x160x / 0x1A0x, RPDOx / TPDOx mapping records .....	102
13.1.11.	0x1C12, 0x1C13, SM2 and SM3 PDO assignment records.....	102
13.1.12.	0x1C32, 0x1C33, SM2 and SM3 parameters.....	102
13.1.13.	0x1F50, Download program data record .....	103
13.1.14.	0x1F52, Verify application software .....	103
13.2.	Manufacturer specific .....	104
13.2.1.	0x2003, Delta-stop steps.....	104
13.2.2.	0x2005 / 0x2006, Modbus register arrays .....	104
13.2.3.	0x200C, SDO encapsulated .....	104
13.2.4.	0x2010, Encoder position latch .....	105
13.3.	DSP402 servo drive profile .....	106
13.3.1.	0x603F, Error code.....	108
13.3.2.	0x6040 / 0x6041, Control word and status word .....	108
13.3.3.	0x6060 / 0x6061, Modes of operation.....	109
13.3.4.	0x6062 / 0x6063 / 0x6064 / 0x6065, Positions .....	109
13.3.5.	0x606B / 0x606C, Velocities.....	109
13.3.6.	0x6073 / 0x6078, Motor current .....	109
13.3.7.	0x6079, DC link circuit voltage.....	110
13.3.8.	0x607A, Target position .....	110
13.3.9.	0x607C, Home offset.....	110
13.3.10.	0x607D, Software position limits.....	110
13.3.11.	0x607F, Maximum profile velocity.....	110
13.3.12.	0x6081, Profile velocity .....	110
13.3.13.	0x6083 / 0x6084 / 0x6086, Profile acceleration / deceleration / type.....	110
13.3.14.	0x6098 / 0x6099 / 0x609A, Homing method / speeds / acceleration.....	111
13.3.15.	0x60C0 / 0x60C1 / 0x60C2 / 0x60C4, Interpolation .....	111
13.3.16.	0x60FD, Digital inputs.....	112
13.3.17.	0x60FE, Digital outputs .....	112
13.3.18.	0x60FF, Target velocity .....	112
14.	CYCLES AND SEQUENCES.....	113

14.1.	Cycle and sequence selection.....	114
14.2.	Jog .....	115
14.3.	Positioning.....	116
14.4.	Homing.....	117
14.4.1.	Index.....	117
14.4.2.	Homing simple.....	117
14.4.3.	Homing, time stop and inversion / no-inversion .....	118
14.4.4.	Homing, time stop, inversion / no-inversion and index.....	118
14.4.5.	Mechanical stop .....	119
14.4.6.	Current position.....	119
14.5.	Delta stop .....	119
14.6.	Time delay.....	120
15.	COMMANDS.....	121
15.1.	Rotate Right (ROR), Rotate Left (ROL).....	121
15.2.	Motor Stop (MST) .....	121
15.3.	Move to Position (MVP) .....	121
15.4.	Reference Search (RFS).....	121
15.5.	Reset (RST) .....	122
15.6.	Alarm Reset (ALR) .....	122
15.7.	Disable/Enable Motor Current (DMC/EMC).....	122
15.8.	Disable/Enable Frequency (DFR/EFR) .....	122
15.9.	Enter Programming Mode (PRG) .....	123
16.	MODBUS.....	124
16.1.	Registers addresses .....	124
16.2.	Registers Description.....	127
16.2.1.	Start, Stop .....	127
16.2.2.	Acceleration, Deceleration.....	127
16.2.3.	Current Speed, Position, Cycle.....	127
16.2.4.	Homing Position, Position Offset .....	127
16.2.5.	Select Cycle Sequence.....	128
16.2.6.	Target Version .....	129
16.2.7.	I/O Bits .....	129
16.2.8.	Configuration .....	129
16.2.9.	Modbus Address.....	130
16.2.10.	Execute Function .....	130
16.2.11.	Maximum, Minimum and Limit Currents .....	130
16.2.12.	Data Link status.....	130
16.2.13.	Fatal Error, Error Log.....	131
16.2.14.	Modbus Baud Rate .....	131
16.2.15.	Status Word .....	131
16.2.16.	Cycles Data.....	132
16.2.17.	Sequence .....	132
16.2.18.	Start, Stop and Input Frequency Configuration .....	133

16.2.19.	Multipurpose Inputs Configuration .....	136
16.2.20.	Aux Inputs .....	137
16.2.21.	Analog input value.....	137
16.2.22.	Output.....	137
16.2.23.	Homing, Time Stop, Release and Research Speeds .....	138
16.2.24.	Position Software Limit Switch Up/Down .....	138
16.2.25.	Temperature .....	138
16.2.26.	Temperature Offset .....	138
16.2.27.	K.....	139
16.2.28.	Steps Accumulation Limit.....	139
16.2.29.	Encoder Position, Speed, Latch and Revolution .....	139
16.2.30.	Power On Calibration Limit .....	139
16.2.31.	Time Constant.....	139
16.2.32.	Start Stop Frequency.....	139
16.2.33.	Resolution .....	139
16.2.34.	CANopen Baud Rate, Address and Status .....	140
16.2.35.	Cycles Counters .....	141
16.2.36.	Encoder Status.....	141
16.2.37.	EtherCAT SyncManager2 Period Average and Variance .....	142
16.2.38.	EtherCAT Sync managers properties .....	142
16.2.39.	Main period .....	143
16.2.40.	Supply voltage.....	143
16.2.41.	EtherCAT AL Status register.....	144
16.2.42.	Ports error counters .....	144
16.2.43.	ESC registers read/write command .....	144
16.2.44.	PDO Param and mapping .....	145
16.2.45.	CANopen Lifetime, Consumer and Producer Heartbeat .....	145
16.2.46.	Serial number .....	145
17.	MODBUS PROTOCOL.....	146
17.1.	Read Holding Register (03).....	147
17.1.1.	Master Read Request .....	147
17.1.2.	Slave Read Answer.....	147
17.2.	Write Single Holding Register (06) .....	148
17.2.1.	Master Write Request .....	148
17.2.2.	Slave Write Answer.....	148
17.3.	Write Multiple Holding Register (16).....	149
17.3.1.	Master Write Request .....	149
17.3.2.	Slave Write Answer.....	149
17.4.	Write File Record (21).....	150
17.4.1.	Master Write File Request .....	150
17.4.2.	Slave Write File Answer.....	150
17.5.	Checksum Calculation .....	151

### 3. HARDWARE

FD-family drivers measure motor current using high-performance Hall-effect sensors equipped with fast overcurrent detection. Current control is managed through a dual H-bridge power stage composed of ultra-low on-resistance MOSFETs. The H-bridges operate with 40 kHz PWM modulation, generating two orthogonal sine-wave currents—one for each motor winding.

The period of sine-wave current is divided into a configurable number of steps, allowing resolutions ranging from 400 up to 204'800 steps per revolution for a standard 1.8° bi-phase motor.

The advanced current loop ensures a highly consistent torque/speed characteristic across the entire operating speed range.

Those application characterized by a variable load torque can achieve a significant power consumption reduction and higher torque availability with the torque control loop available in V6.

The load torque is calculated using the magnetic encoder acquisition. This value is continuously processed to determine the best amount of current needed to move the load inside motor operating limits (the sine wave peak current value will range between the programmed values  $I_{MIN}$  and  $I_{MAX}$ ).

Another key feature of V6 is the ability to accumulate steps that cannot be executed due to a sudden increase in resistive torque beyond the motor's maximum capacity. In such cases, V6 maintains maximum motor torque, and when the load torque decreases, the motor recovers the accumulated steps by accelerating and reaching the target position. The transition from following mode to synchronous mode is smooth, achieved through bump-less speed adjustment, ensuring vibration-free operation.

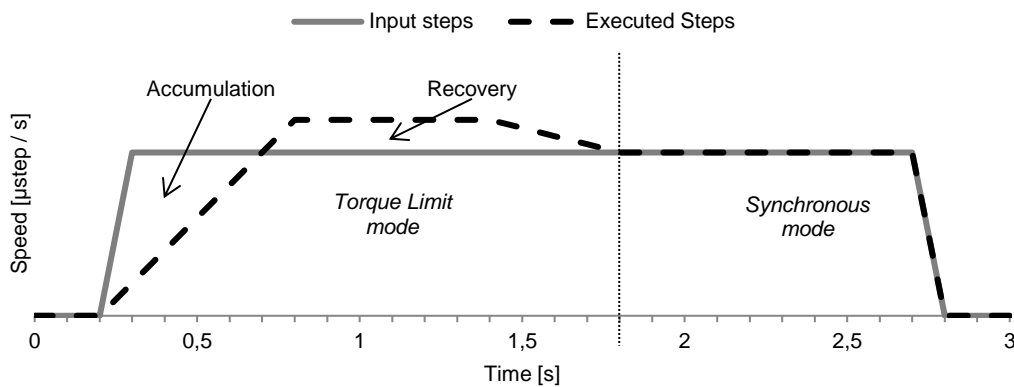


Fig. 1 - Speed profile with step accumulation and recovery during acceleration

In motion applications characterized by high acceleration and inertial loads, traditional stepper systems require sufficient torque margins to ensure that any increase in load does not cause the motor to lose synchronization, resulting in step loss or even a complete stop if operating above the start/stop frequency. In other words, traditional stepper drivers necessitate oversizing the motor and driver to account for such conditions.

With the V6 control firmware, the driver dynamically increases current and torque up to the maximum configured value. In cases where the resistive torque exceeds this limit, the resulting reduction in speed and acceleration is managed by accumulating unexecuted input steps. A configurable alarm threshold monitors the step accumulation. Once the resistive torque decreases, the driver recovers the accumulated steps without any loss of position.

If the accumulated steps exceed the configurable limit, the system triggers the "step accumulation limit" alarm. During accumulation, the red LED remains steadily lit, and the corresponding bits in the status word are set.

The V6 control firmware effectively combines the advantages of stepper systems—such as low cost, simplicity (no need for PID tuning), minimal position overshoot, and a high torque-to-motor size ratio—with the benefits of brushless systems, including high efficiency (adaptive current adjustment based on load) and reliable position retention.

Protections such as over-voltage, over-current and over-temperature are also implemented (low-voltage protection inhibits the driver when the supply voltage is below a minimum allowed value).

Type	Rated voltage	Abs. Enc.	Digital I/O	Analog or digital Inputs	Ether CAT	RS-485	CAN open	RS-232
FD1.1E	24 – 60 V <sub>DC</sub>	12 bits	2 IN 1 OUT		✓			✓
FD1.1EC	24 – 80 V <sub>DC</sub>	12 bits	4 IN 2 OUT		✓			✓
FD1.2EC	24 – 80 V <sub>DC</sub>	12 bits	4 IN 2 OUT		✓			✓
FD2.1E	24 – 130 V <sub>DC</sub>	12 bits	7 IN 3 OUT		✓			✓
FD2.2E	24 – 130 V <sub>DC</sub>	12 bits	5 IN 3 OUT		✓			✓
FD2.1C	24 – 130 V <sub>DC</sub>	14 bits	3 IN 2 OUT	1 IN		✓		✓
FD2.1A	24 – 130 V <sub>DC</sub>	14 bits	3 IN 2 OUT	1 IN			✓	✓
FD2.1EC	24 – 130 V <sub>DC</sub>	14 bits	3 IN 2 OUT	1 IN	✓			✓

Tab. 1 – V6 drivers



## 4. DWLOADER

DwLoader is a PC application designed to program the driver and test its functionality. It is particularly useful for monitoring drive parameters and troubleshooting issues while the driver is controlled by the fieldbus master.

The application requires no installation. Simply copy the contents of the provided CD to a folder on your PC (ensure there are no blank spaces in the folder path). Double-click on “DwLoader.exe” to launch the application, and the main window will appear.

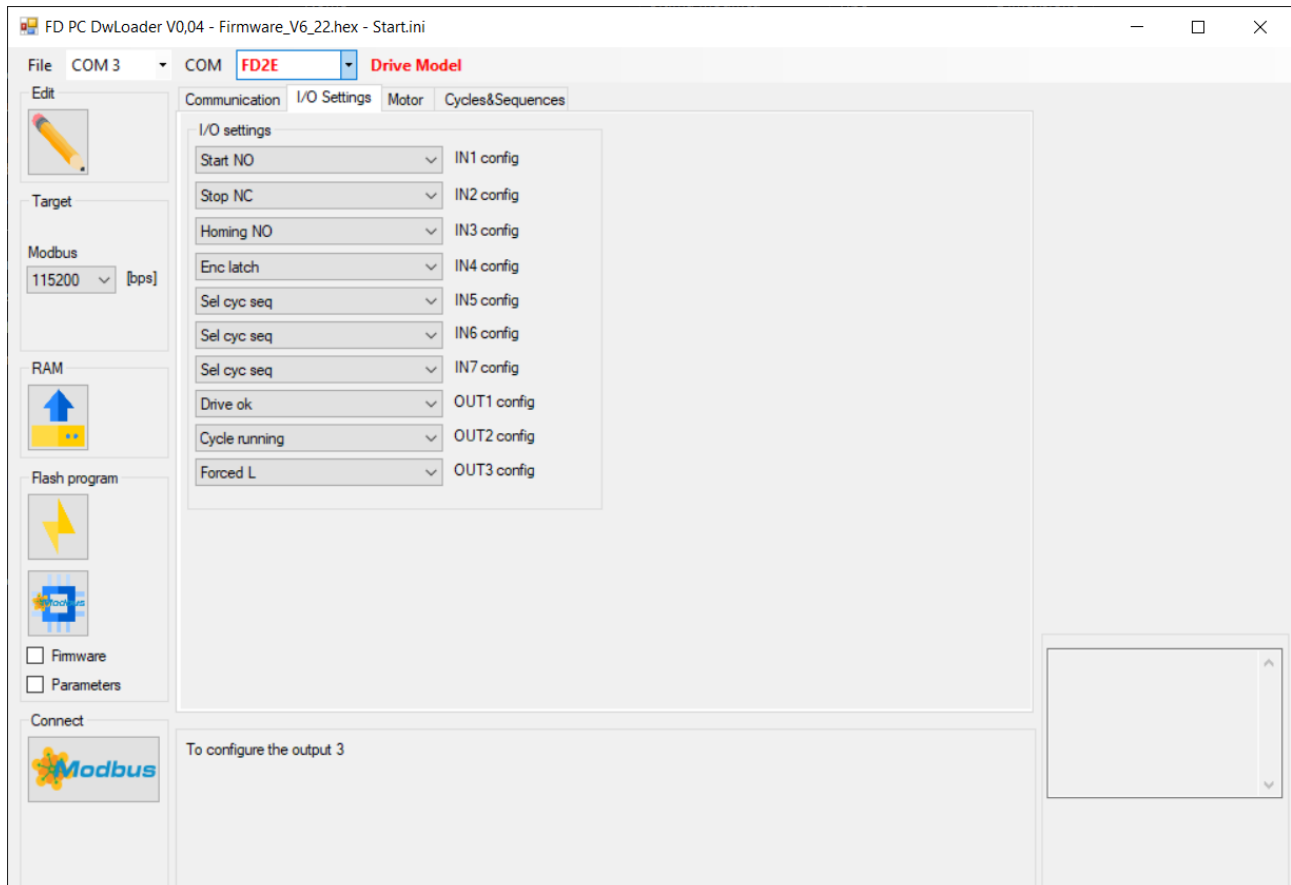


Fig. 2 – DwLoader main window

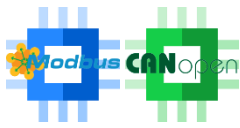
**Note:**

The COM port and the correct driver model in the menu strip of the window need to be selected (use Windows Device Manager to select the proper COM to be used).

The Microsoft .NET Framework 4.0 needs to be present (it is provided into the CD together with the DwLoader).

CANopen DwLoader functionalities work with IXXAT VCI V4 drivers.

## 4.1. In-application programming



In-application programming (IAP) tool allows to re-program firmware and/or parameters using Modbus protocol via RS-485 or RS-232 (blue push-button) or CANopen protocol (green push-button). Ref. to 17.4 Write File Record (21) for Modbus protocol details and ref. to 11.2.6 SDO block download for CANopen protocol details

When IAP takes place, FD drive green and red LEDs flash alternatively.

CANopen and Modbus node address is configured via DIP switches. When all DIP switches are off, since the address zero is forbidden, the address is taken from programmable parameter.

IAP allows to modify the field-bus parameters: baud rate and node address:

- Target text-box identifies the parameters to be used during programming, those which are currently active in the device.
- Communication Modbus/CANopen baud rate and address are the new parameters, which will be programmed and will be active after reset.

*Latest bootloader is version is V0.04.*

## 4.2. RAM data upload



Data upload reads from the target RAM all the data and updates the DwLoader boxes accordingly.

*Note:*

*Data upload combined with In Application Programming are useful when it is needed to re-program a new driver with the same configuration of another existing.*

## 4.3. User flash program



User flash program allows to completely reprogram the driver microprocessor (bootloader, firmware and parameters). Just a common PC and a 9 poles pin-to-pin RS-232 cable (or a simple USB to RS-232 converter) are needed.

1. Power off the driver.
2. Open the cover and set the driver in programming mode by turning on DIP switch 8.
3. For FD1.1EC and FD1.2EC mount back the cover.
4. Power on the driver (FD1.1E, FD1.1EC, FD1.2EC, FD2.1EC, FD2.1AC, FD2.1C: all LEDs off. FD2.1E: red LED will be on; all other LEDs will be all off).
5. Click on "User Flash Program" button. The command window will appear. In case of error, i.e. target does not answer or COM does not exist, a proper message will be shown.
6. Power off the driver (it might take time to discharge the energy accumulated in the capacitors, as the driver is absorbing very low power in programming mode).
7. Remove the programming mode, i.e. DIP switch 8 off.
8. Power on the driver again. Blinking green LED means that the programming procedure occurred properly.

*Note:*

*Folder path cannot contain blank space characters (" ").*

#### 4.4. Data savings



When DwLoader is launched, it loads the parameters from previously used .ini file (if such file does not exist, the Start.ini file inside the DwLoader folder will be used). It is possible to load new parameters from File ⇒ Open and save them from File ⇒ Save / SaveAs. When saving a set of parameters into .ini file, Start.ini is automatically updated.

The actual file in use is displayed in the top bar of DwLoader window.

When closing the program, if any modification from the program start-up settings is present, it will ask to save.

#### 4.5. Create binary files



Via TwinCAT or similar EtherCAT automation studios it is possible to reprogram firmware and or parameters of the drive.

The driver EtherCAT state machine shall be in bootstrap mode. When the drive is in bootstrap mode, the red and green LEDs flash alternatively.

Using DwLoader commands File ⇒ Create Parameters binary and File ⇒ Create Parameters binary it is possible to generate the two files: Firmware\_Bin\_V6\_25.bin and Parameters\_Bin.bin, that will be stored into DwLoader folder.

The file name shall start with “Firmware\_Bin” and “Parameters\_Bin”. E.g., it can be “Parameters\_Bin\_Motor1.bin”.

The password to transfer them via Filetransfer over EtherCAT (FoE) is 0.

#### 4.6. Data modify



Click on the pencil symbol to modify the driver parameters.

Name	Unit	Description
<b>Communication</b>		
RS-232 baud rate	[bps]	Range: 4'800 – 115'200 bps for RS-232
RS-485 address		FD2.1C only. Modbus address from flash memory, used when all DIP switches are off.
CANopen baud rate	[KHz]	FD2.1AC only.
CANopen address		FD2.1AC only. CANopen Node-Id from flash memory, used when all DIP switches are off.
RPDO and TPDO settings		FD2.1AC only. To preconfigure in flash the PDO mapping and PDO settings. When PDOs are preconfigured, the drive can be directly set in Operational and start the process for faster CANopen network startup time.
CANopen error control		FD2.1AC only. Heartbeat and node guarding settings preconfigured in flash for faster CANopen network startup time.

I/O settings		FD1.1E	FD1.xEC	FD2.1E	FD2.1xC
IN1 configuration		0: Disable current	0: Step frequency	0: Step frequency	0: Step frequency
		19: Enable current	1: Start NO	1: Start NO	1: Start NO
		22: Interlock current	2: Start NC	2: Start NC	2: Start NC
		1: Disable frequency	4: not used	4: not used	4: not used
		2: Homing NO	6: Quadrature step A	6: Quadrature step A	6: Quadrature step A
		3: Homing NC	7: Start NO, Stop NC	7: Start NO, Stop NC	7: Start NO, Stop NC
		4: not used	8: Start NC, Stop NO	8: Start NC, Stop NO	8: Start NC, Stop NO
		5: Limit switch up NO	30: Start NO, Stop NC, Sel Cyc Seq +1	30: Start NO, Stop NC, Sel Cyc Seq +1	30: Start NO, Stop NC, Sel Cyc Seq +1
		6: Limit switch up NC	31: Start NC, Stop NO, Sel Cyc Seq +1	31: Start NC, Stop NO, Sel Cyc Seq +1	31: Start NC, Stop NO, Sel Cyc Seq +1
		7: Limit switch dw NO	32: Start NO, Stop NC, Sel Cyc Seq +2	32: Start NO, Stop NC, Sel Cyc Seq +2	32: Start NO, Stop NC, Sel Cyc Seq +2
		8: Limit switch dw NC	33: Start NC, Stop NO, Sel Cyc Seq +2	33: Start NC, Stop NO, Sel Cyc Seq +2	33: Start NC, Stop NO, Sel Cyc Seq +2
		9: Encoder latch			
		10: Step frequency			
		11: Select cycle or sequence			
		12: Delta stop NO			
		13: Delta stop NC			
		14: Quadrature step A			

e-mail: [info@auxind.com](mailto:info@auxind.com)

Tel: (+39) 0522 520312 – Fax, Tel: (+39) 0522 521333

Via Martiri di Cervarolo, 74/3 – 42123 Reggio Emilia, Italy

C.F. / P.I. **01844360352** – R.E.A. di R.E. 228913 –

Reg. Impr. 27689 / 99

**AUXIND**

[www.auxind.com](http://www.auxind.com)

I/O settings		FD1.1E	FD1.xEC	FD2.1E	FD2.1xC
		15: Start NO 16: Start NC 17: Stop NO 18: Stop NC 20: Start NO, Stop NC 21: Start NC, Stop NO 30: Start NO, Stop NC, Sel Cyc Seq +1 31: Start NC, Stop NO, Sel Cyc Seq +1 32: Start NO, Stop NC, Sel Cyc Seq +2 33: Start NC, Stop NO, Sel Cyc +2			
IN2 configuration		0: Disable current 19: Enable current 22: Interlock current 1: Disable frequency 2: Homing NO 3: Homing NC 4: not used 5: Limit switch up NO 6: Limit switch up NC 7: Limit switch dw NO 8: Limit switch dw NC 9: Encoder latch 10: Direction frequency 11: Select cycle or sequence 12: Delta stop NO 13: Delta stop NC 14: Quadrature step B 15: Start NO 16: Start NC 17: Stop NO 18: Stop NC 20: Start NO, Stop NC 21: Start NC, Stop NO 30: Start NO, Stop NC, Sel Cyc Seq +1 31: Start NC, Stop NO, Sel Cyc Seq +1 32: Start NO, Stop NC, Sel Cyc Seq +2 33: Start NC, Stop NO, Sel Cyc Seq +2	0: Direction frequency 1: Stop NO 2: Stop NC 3: Select cycle or sequence 4: not used 6: Quadrature step B 7: Start NO, Stop NC 8: Start NC, Stop NO 30: Start NO, Stop NC, Sel Cyc Seq +1 31: Start NC, Stop NO, Sel Cyc Seq +1 32: Start NO, Stop NC, Sel Cyc Seq +2 33: Start NC, Stop NO, Sel Cyc Seq +2	0: Direction frequency 1: Stop NO 2: Stop NC 3: Select cycle or sequence 4: not used 6: Quadrature step B 7: Start NO, Stop NC 8: Start NC, Stop NO 30: Start NO, Stop NC, Sel Cyc Seq +1 31: Start NC, Stop NO, Sel Cyc Seq +1 32: Start NO, Stop NC, Sel Cyc Seq +2 33: Start NC, Stop NO, Sel Cyc Seq +2	0: Direction frequency 1: Stop NO 2: Stop NC 3: Select cycle or sequence 4: not used 6: Quadrature step B 7: Start NO, Stop NC 8: Start NC, Stop NO 30: Start NO, Stop NC, Sel Cyc Seq +1 31: Start NC, Stop NO, Sel Cyc Seq +1 32: Start NO, Stop NC, Sel Cyc Seq +2 33: Start NC, Stop NO, Sel Cyc Seq +2
IN3 configuration		n. a.	0: Disable current 19: Enable current 22: Interlock current 1: Disable frequency 2: Homing NO 3: Homing NC 4: not used 5: Limit switch up NO 6: Limit switch up NC 7: Limit switch dw NO 8: Limit switch dw NC 9: Encoder latch 26: Encoder latch ris 27: Encoder latch fal 10: Direction frequency 11: Select cycle or sequence 12: Delta stop NO 13: Delta stop NC 15: Start NO 16: Start NC	0: Disable current 19: Enable current 22: Interlock current 1: Disable frequency 2: Homing NO 3: Homing NC 4: not used 5: Limit switch up NO 6: Limit switch up NC 7: Limit switch dw NO 8: Limit switch dw NC 9: Encoder latch 26: Encoder latch ris 27: Encoder latch fal 10: Direction frequency 11: Select cycle or sequence 12: Delta stop NO 13: Delta stop NC 15: Start NO 16: Start NC	0: Disable current 19: Enable current 22: Interlock current 1: Disable frequency 2: Homing NO 3: Homing NC 4: not used 5: Limit switch up NO 6: Limit switch up NC 7: Limit switch dw NO 8: Limit switch dw NC 9: Encoder latch 26: Encoder latch ris 27: Encoder latch fal 10: Direction frequency 11: Select cycle or sequence 12: Delta stop NO 13: Delta stop NC 15: Start NO 16: Start NC

I/O settings		FD1.1E	FD1.xEC	FD2.1E	FD2.1xC
			17: Stop NO 18: Stop NC	17: Stop NO 18: Stop NC	17: Stop NO 18: Stop NC
IN4 configuration		n. a.	0: Disable current 19: Enable current 22: Interlock current 1: Disable frequency 2: Homing NO 3: Homing NC 4: not used 5: Limit switch up NO 6: Limit switch up NC 7: Limit switch dw NO 8: Limit switch dw NC 9: Encoder latch 26: Encoder latch ris 27: Encoder latch fal 10: Direction frequency 11: Select cycle or sequence 12: Delta stop NO 13: Delta stop NC 15: Start NO 16: Start NC 17: Stop NO 18: Stop NC	0: Disable current 19: Enable current 22: Interlock current 1: Disable frequency 2: Homing NO 3: Homing NC 4: not used 5: Limit switch up NO 6: Limit switch up NC 7: Limit switch dw NO 8: Limit switch dw NC 9: Encoder latch 26: Encoder latch ris 27: Encoder latch fal 10: Direction frequency 11: Select cycle or sequence 12: Delta stop NO 13: Delta stop NC 15: Start NO 16: Start NC 17: Stop NO 18: Stop NC	0: Disable current 19: Enable current 22: Interlock current 1: Disable frequency 2: Homing NO 3: Homing NC 4: not used 5: Limit switch up NO 6: Limit switch up NC 7: Limit switch dw NO 8: Limit switch dw NC 11: Select cycle or sequence 15: Start NO 16: Start NC 17: Stop NO 18: Stop NC 30: Analog speed scaling
IN5 configuration		n. a.	n. a.	0-10: not used 11: Select cycle or sequence	n. a.
IN6 configuration		n. a.	n. a.	0-10: not used 11: Select cycle or sequence	n. a.
IN7 configuration		n. a.	n. a.	0-10: not used 11: Select cycle or sequence	n. a.
OUT1 configuration		0: Cycle running, 1: Encoder index 50ms 2: Encoder index Π 3: Position uploaded 4: Axis calibrated 5: Power on pos ok 6: Torque limit 7: Cycle run + /Dstop 8: Forced L 9: Forced H 10: Drive ok 11: Drive alarm	0: Drive ok 1: Drive alarm 8: Forced L 9: Forced H	0: Drive ok 1: Drive alarm 8: Forced L 9: Forced H	0: Drive ok 1: Drive alarm 8: Forced L 9: Forced H
OUT2 configuration		n. a.	0: Cycle running 1: Encoder index 50ms 2: Encoder index Π 3: Position uploaded 4: Axis calibrated 5: Power on pos ok 6: Torque limit 7: Cycle run + /Dstop 8: Forced L 9: Forced H	0: Cycle running 1: Encoder index 50ms 2: Encoder index Π 3: Position uploaded 4: Axis calibrated 5: Power on pos ok 6: Torque limit 7: Cycle run + /Dstop 8: Forced L 9: Forced H	0: Cycle running 1: Encoder index 50ms 2: Encoder index Π 3: Position uploaded 4: Axis calibrated 5: Power on pos ok 6: Torque limit 7: Cycle run + /Dstop 8: Forced L 9: Forced H
OUT3 configuration		n. a.	n. a.	8: Forced L 9: Forced H	n. a.

Motor		
Encoder enable		Selected: activates encoder functionality.
Absolute multi-turn encoder enable		FD2C, FD2AC, FD2EC only. Selected: activates multi-turn absolute encoder functionality.
Position upload		Selected: The exact multi-turn power-off position and current configuration are reloaded during power-on, provided the position deviation is within $\pm 1.8^\circ$ . Deselected: The power-on position is calculated as the power-off position plus the position deviation, provided the deviation is within the power-on calibration limit.
Software limit switch up	[step]	Selected: Enables and configures the software limit switch for increasing positions.
Software limit switch down	[step]	Selected: Enables and configures the software limit switch for decreasing positions.
Disable torque control		Selected: Disables torque control and the accumulation of steps. <i>Note: Current reduction is active when the motor is stopped.</i>
Invert direction		Selected: Reverses the direction of rotation. <i>For example: Changes counterclockwise (CCW) movement for decreasing positions to clockwise (CW) movement for the same.</i>
Starting boost		Selected: the motor starts with a higher current to overcome static friction and load inertia. This feature anticipates torque control, allowing for higher accelerations. After the boost phase, the current demand reverts to torque-controlled values.
Select 32 cycles		When cycles or sequences are selected using a binary combination of digital inputs: Selected: digital inputs select cycles [0-31]. Deselected: digital inputs select sequences [0-9] and cycles [10-31]. <i>Note: an input configured as Sel. Cyc. Seq. overrides fieldbus selection</i>
Auto full step		Selected: Maximizes the motor's current RMS (Root Mean Square), increasing motor torque. Deselected: The current remains sinusoidal, resulting in smoother motion and reduced motor heating.
Maximum motor current	[mA]	Specifies the peak value of the sine wave phase current. <i>Note: The correct value depends on motor ratings and duty cycle. Typically, it is <math>\sqrt{2}</math> times the motor rated current.</i>
Minimum motor current	[mA]	Specifies the minimum peak value of the sine wave phase current. <i>Note: Typically, it is approximately half of the maximum motor current.</i>
Step accumulation limit	[step]	Defines the maximum number of accumulated steps (limited to 10 revolutions).
Time constant	[256 x 50 s]	Preset to 1. <i>Important: Do not modify.</i>
Reference resolution num.		Configures the number of steps per revolution using the formula: $\text{step}_{\text{REV}} = \frac{50 \cdot \text{Res}_{\text{NUM}}}{\text{Res}_{\text{DEN}}}$ Res <sub>NUM</sub> and Res <sub>DEN</sub> range from 1 to 65'535. The corresponding resolution is limited from 400 to 204'800 $\mu\text{step/rev}$ .
Reference resolution den.		

Cycles		
Type	[1 – 7]	1: Jog 2: Relative positioning 3: Homing 4: Delta stop 5: Absolute positioning 7: Time delay
Speed	[step / s]	Range: 1 – 300'000 step/s <i>Note: if higher angular velocities [rev/s] are needed, slightly decrement the resolution [step/rev]</i>
Position	[step] or [ms]	Motor steps to be executed in a positioning relative cycle. Position destination in a positioning absolute cycle. Maximum motor steps to be executed in a delta stop cycle. Time delay in a time delay cycle.
Direction	[0-1]	When the direction is not inverted: 0: CW rotation to increasing positions 1: CCW rotation to decreasing positions
Delta stop	[step]	Number of steps between the activation of delta stop input and the end of movement
Arrows for cycle	<< >>	To configure the movement data of 32 programmable cycles.

selection		
<b>Sequences</b>		
Sequence parameters [0, ..., 19]		When a sequence is started, the motor will execute its parameters. Each of them can be one of the following: 0 – 31: Cycle number to be executed in sequence 254: Stop sequence 255: Loop sequence (restart the sequence from parameter number 0). When no stop and no loop are present in a sequence, the next sequence will be executed.
Arrows for sequence selection	<< >>	To configure the 10 programmable sequences.
<b>Ramp profile</b>		
Type		Linear, parabolic or S-curve.
Start / stop frequency	[step]	Selected: it enables and set the start / stop frequency. Deselected: automatic.
Acceleration	[kstep / s <sup>2</sup> ]	e.g., 15 stands for an acceleration of 15'000 step / s <sup>2</sup> .
Deceleration	[kstep / s <sup>2</sup> ]	e.g., 15 stands for a deceleration of 15'000 step / s <sup>2</sup> .
<b>Homing</b>		
Homing position	[step]	The position loaded in position counter at the end of a homing cycle.
Homing		Homing methods
Position offset	[step]	The encoder magnet is mounted in a random angular position. This parameter modifies the zero-encoder position.
Power-on calibration limit	[step]	Programmable power-on position deviation to assess the axis calibration and recalculate the multi-turn axis position.
Research speed	[step / s]	Speed to research the homing switch.
Release speed	[step / s]	Speed to release from the homing switch.
Time stop	[ms]	Delay time between research and release speed in homing cycles.

Tab. 2 - DwLoader parameters

## 4.7. Data Exchange

The Connect Modbus button initiates continuous communication with the target device using the Modbus RTU protocol.

Once connected, the Data Exchange window appears, allowing real-time monitoring of parameters such as position, speed, alarms, and temperature. It also enables command execution and RAM data exchange, among other functions.

A faster blinking green LED on the target indicates that communication is active.

The screenshot shows the 'Modbus data exchange - Target FD1: firmware V6.01' window. It features a top status bar with real-time data: Position 10457, Speed 0, ENC Pos 10460, ENC Speed 0, Curr. Cyc 0, Sel. Cyc 0, Temp 42 [°C], Mot Curr 187 [mA], Alarm Drive ok, and VDC 45,00. Below this, there are sections for GEN. DATA (485\_Address, Scan Interval, Target Address), CYCLE 0 COMMANDS (Rotate, Move, Stop, Ref. Search), Status Word (Drive Ready, Alarm Active, etc.), I/O Bits (IN1 Homing, OUT1 Cycle running), Configuration (Encoder, S-curve, etc.), Data link status (PDI, Enhanced link detection), Encoder Status (Parity Error, Magnetic Decrement), Application Layer (Unknown), Ports Error Counters (P0-P3 invalid frames), and Sync manager. A 'START' button is prominently displayed in the center. The bottom section shows Tx/Rx strings and ESC access parameters.

Fig. 3 - Modbus data exchange

Before clicking Connect Modbus, make sure to verify the baud rate in the DwLoader window. If the baud rate is incorrect, communication will fail, and a timeout message will be displayed.



The Connect CANopen button opens the Data Exchange window, which allows you to configure PDOs and test the communication protocol. This feature is compatible with IXXAT USB-to-CAN converter libraries.

The screenshot displays the CANopen Data Exchange software interface. The window is titled "CANopen Data Exchange" and contains various configuration panels. On the left, there's a "Network Management Objects (NMT)" section with buttons like "Start Remote Node", "Stop Remote Node", "Enter Pre-Operational", "Reset Node", and "Reset Communication". Below that is "Service Data Objects (SDO)" with a dropdown for "Device type" and buttons for "Initiate SDO Download" and "Initiate SDO Upload". The main area is divided into several sections: "Save parameters" with a "Save" button; "Modes of operation (0x6060)" with a dropdown set to "0: Undefined"; "Status word (0x6041)" and "Control word (0x6040)" with checkboxes for various states like "Ready to switch on", "Switched on", "Operation enable", "Fault", "Voltage enable", "Quick stop", "Switch on disable", "Torque limit", "Remote", "Target reached", "Operation mode", and "Position upload"; "Power disabled" and "Power enabled" sections with buttons like "Start", "Not ready to switch on", "Switch on disabled", "Ready to switch on", "Switched on", "Operation enabled", "Quick stop active", "Shut down", "Switch on", "Disable voltage", "Quick stop", "Disable operation", "Enable operation", and "Fault reset"; "Fault" section with "Fault reaction active" and "Fault" buttons; "Target velocity (0x60FF)", "Target position (0x607A)", "Profile velocity (0x6081)", "Position actual value (0x6063)", "Position demand value (0x6062)", "Velocity actual value (0x606C)", and "Velocity demand value (0x606B)" sections with input fields and "Up"/"Dw" buttons; "Dig I/O (0x2005, 12)" section with checkboxes for "IN1/2 Step Frequency", "IN3/4 Direction", "IN5 Disable Current", "IN6 Disable Frequency", "IN7", "IN8", "OUT9 Drive OK", and "OUT10 Cycle Run"; and "Position address record (0x2000)" section with a list of addresses from 0 to 19. At the bottom, there's a "Process Data Objects (PDO)" section with a table for RPDO1-RPDO4 and TPDO1-TPDO4, a "COB-ID" field set to 525, and a "Type" dropdown set to "0: Acyclic, synchronous". To the right of the PDO table is a log window showing a list of messages with columns for Time, TX/RX, COBID, DLC, Type, and data bytes. The log shows several "Init SDO Up confirmed" messages. At the bottom right, there's a "CANopen" logo.

Fig. 4 – CANopen data exchange

## 4.8. Modbus Data Exchange - Oscilloscope Feature

By clicking the Oscilloscope button, you can visualize a graphical representation of all Modbus register values with a time resolution of 1 ms.

This feature allows, for example, simultaneous drive control via EtherCAT or digital I/O while plotting selected data in real time using DwLoader over RS-232.

Start Command checkbox:

When enabled, this option allows the Time Analysis function to issue a Start command after 50 ms of data sampling. The data buffer retains approximately 50 samples collected prior to the Start command.

Write to File checkbox:

When checked, this option creates a .txt file in the current folder containing the sampled time and data values.

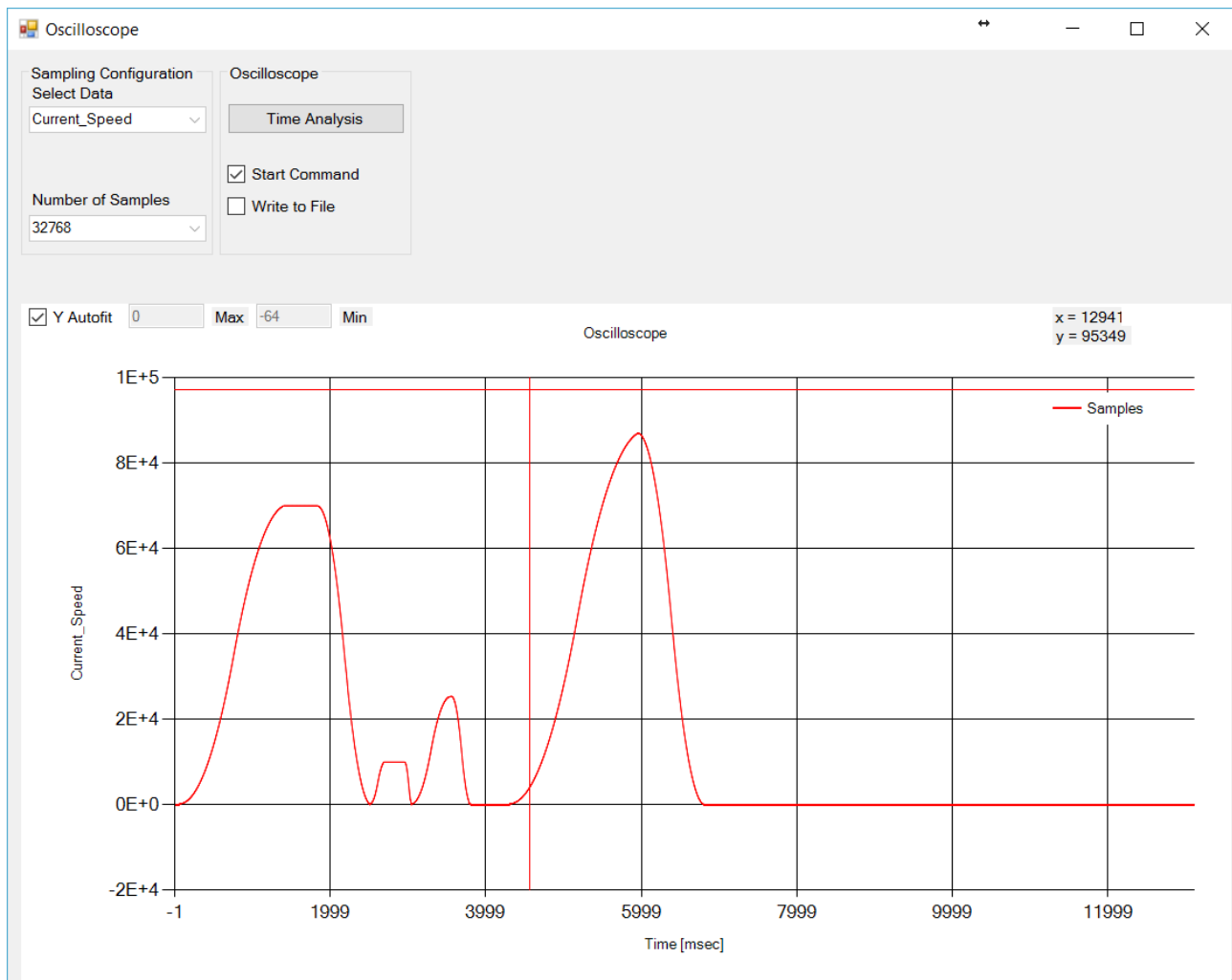


Fig. 5 – Oscilloscope

Note:

Modbus baud rate cannot be lower than 115'200 bps.

Because many records are overlapped in time and then reconstructed, Number of Samples does not reflect the exact number of samples plotted. During the reconstruction, if more than 1% of samples are missing, record will be discarded. Otherwise, missing samples will be interpolated.

Serial port needs to be set with minimum latency (check Device Manager ⇒ Ports (COM & LPT) ... ⇒ Advanced properties).

## 5. CONTROL METHODS

FD drives can be controlled from an external system using following methods:

### 1. Step / dir or quadrature steps A/B

Using two digital signals from a motion controller, the system generate acceleration and deceleration ramps based on a given position or velocity setpoint.

Step/Dir mode: with each rising edge of the step signal, the motor rotates a configurable angle in either the clockwise (CW) or counterclockwise (CCW) direction, as determined by the direction signal.

Quadrature mode: the sequence of transitions between signals A and B generates both count pulses and direction information, controlling motor rotation accordingly.

### 2. Start / stop and sequence selection

Digital inputs can be used to select, start, and stop up to 32 predefined movements (referred to as cycles) or 10 sequences of cycles. Control can be managed by a simple system such as a PLC or pushbuttons.

The drive internally generates the motion profile, which may include linear, parabolic, or S-curve/jerk-limited ramps.

Motion profiles can be stored in Flash memory for persistent use or dynamically updated in RAM via fieldbus communication for real-time adjustments.

### 3. Modbus RTU

Using the Modbus RTU protocol, external systems can configure, select, start, and stop cycles and sequences.

### 4. CANopen

FD drives with the “A” suffix support CANopen communication, with following modes of operation:

- Profile position,
- Profile velocity,
- Homing mode,
- Interpolated position.

All Modbus registers are accessible as CANopen objects.

### 5. EtherCAT

FD drives with “E” suffix implement CAN over EtherCAT (CoE), with following modes of operation:

- Profile position,
- Profile velocity,
- Homing mode,
- Interpolated position,
- Cyclic synchronous position (CSP),
- Cyclic synchronous velocity (CSV).

All Modbus registers are accessible as EtherCAT objects.

## 6. INPUTS / OUTPUTS

FD drives have configurable inputs and outputs.

Type	Digital Inputs	Digital Outputs	Analogue or digital Inputs
FD1.1E	2	1	-
FD1.1EC, FD1.2EC	4	2	-
FD2.1E	5	3	-
FD2.1C, FD2.1AC, FD2.1EC	3	2	1

Tab. 3 – Inputs / Outputs

FD1E inputs and outputs are limited in number and all the possible configurations are available for them.

FD1EC, FD2E, FD2EC inputs are separated in groups, normally configured as:

- first two inputs are step/dir modes or start/stop.
- other two inputs are multipurpose, such as: homing, limit switch, delta stop, enable or disable current, etc.
- remaining inputs are for cycle or sequence selection.

FD2E OUT1 is used for signaling the alarm status (normally closed or normally opened). OUT2 is multipurpose, i.e. cycle running, axis calibrated, etc. OUT3 is used as torque off feedback.

The status of all the inputs can be monitored from Modbus register IO\_BITS or EtherCAT/CANopen object 60FD Digital inputs.

The outputs can be forced high or low using Modbus register OUT\_1\_CNF, OUT\_2\_CNF, or EtherCAT/CANopen object 60FE Digital outputs.

### 6.1. Start/Stop and Input Frequency Signals

IN1 and IN2 can be configured as step/dir, quadrature A/B input frequencies, start/stop NO/NC and other extra functionalities such as cycle or sequence selection. Refer to table below:

Function	Drive input	Description	FD1.1E	FD1.1EC, FD1.2EC, FD2.1E, FD2.1xC
			Input config. reg. value	
Step frequency	IN1	Every pulse produces a motor step. Maximum input frequency: 300 kHz	10	0
Direction frequency	IN2	Active: Up-counting (CW). Inactive: Down-counting (CCW). <i>Note: Applying "invert direction" reverses the motor's sense of rotation.</i>	10	0
Quadrature steps	IN1, IN2	The position counter counts up or down at every signal edge based on the level of the other signal.	14	6
Start	FD1.1E: IN1, IN2 FD1.xEC, FD2.1E, FD2.1xC: IN3, IN4	Starts the selected cycle or sequence.	15: NO 16: NC	15: NO 16: NC
	FD1.xEC, FD2.1E, FD2.1xC: IN1		1: NO 2: NC	1: NO 2: NC
Stop	FD1E: IN1, 2 FD1.xEC, FD2.1E, FD2.1xC: IN3, IN4	Stops the running cycle or sequence.	17: NO 18: NC	17: NO 18: NC
	FD1.xEC, FD2.1E, FD2.1xC: IN2		1: NO 2: NC	1: NO 2: NC

Start, /Stop	IN1, IN2	Active: Starts the selected cycle or sequence. Inactive: Stops it.	20: Start NO, Stop NC 21: Start NC, Stop NO	7: Start NO, Stop NC 8: Start NC, Stop NO
Start, /Stop, Sel cyc seq +1	IN1, IN2	Active: Increments the "Sel Cyc Seq" register by 1 and starts the selected cycle. Inactive: Stops it. <i>Note: Intended for separate starts, such as "Start CW" and "Start CCW"</i>	30: NO 31: NC	30: NO 31: NC
Start, /Stop, Sel cyc seq +2	IN1, IN2	Active: Increments the "Sel Cyc Seq" register by 2 and starts the selected cycle. Inactive: Stops it. <i>Note: Intended for separate starts, such as "Start CW" and "Start CCW"</i>	32: NO 33: NC	32: NO 33: NC

Tab. 4 - Inputs description

**Notes:**

Fieldbus start commands are always available and are managed with higher priority than input step signals. During the execution of a command, step signals are disabled and are re-enabled once the movement is complete.

In step/dir and quadrature step modes, acceleration and deceleration ramps are not provided by the drive itself. Instead, the motion controller supplying the signals must manage these ramps.

Quadrature step signals allow the connection of an encoder master to multiple drives, enabling synchronized movement among the drives and the encoder. These signals can be enabled or disabled using the disable frequency signal, and their speed can be adjusted using configurable resolution.

Hardware and software limit switches are inactive in step/dir and quadrature step modes.

When the input is configured for step frequency or quadrature steps, measures should be taken to minimize external disturbances, such as using shielded cables. With the exception of delta stop, direction input, and encoder latch configurations, all other input configurations include a digital filter and a Schmitt trigger with a maximum delay of 2.1 ms.

## 6.2. Multipurpose Inputs

Following functions are implemented:

Function	Drive: Input	Description	FD1.1E	FD1.1EC, FD1.2EC, FD2.1E, FD2.1xC
			Input config. reg. value	
Disable current	FD1.1E: IN1, IN2 FD1.xEC, FD2.1E, FD2.1xC: IN3, IN4	When disable current is active, it frees the motor shaft. However, using the magnetic encoder, the drive can continue tracking the shaft's position. Once current is re-enabled, the position command and status word bits are restored.	0	0
Enable current	FD1.1E: IN1, 2 FD1.xEC, FD2.1E, FD2.1xC: IN3, 4		19	19
Interlock current	FD1.1E: IN1, 2 FD1.xEC, FD2.1E, FD2.1xC: IN3, 4	When using the EtherCAT and CANopen fieldbuses, the motor current is managed by state machine of DSP-402, hence enable/disable current inputs might be in conflict with state machine status. Interlock can be used as: Active: motor current is controlled via DSP-402 Inactive: motor current is off.	22	22
Disable frequency	FD1.1E: IN1, 2 FD1.xEC, FD2.1E, FD2.1xC: IN3, 4	Active: it inhibits all movements initiated by fieldbus commands or input steps. Despite this, the motor continues to generate holding torque.	1	1
Homing	FD1.1E: IN1, 2 FD1.xEC, FD2.1E, FD2.1xC: IN3, 4	Uses the homing sensor to establish the reference position.	2: NO 3: NC	2: NO 3: NC
Hardware limit switches	FD1.1E: IN1, 2 FD1.xEC, FD2.1E, FD2.1xC: IN3, 4	Stops motor movements in the direction of the activated switch. These switches can also serve as homing sensors during homing cycles. They are not active in input step modes.	5: Up NO 6: Up NC 7: Down NO 8: Down NC	5: Up NO 6: Up NC 7: Down NO 8: Down NC
Encoder position latch	FD1.1E: IN1, 2 FD1.xEC, FD2.1E, FD2.1xC: IN3	Rising and falling edges of the configured input trigger a rapid interrupt, which captures and latches the multi-turn encoder position. The latched positions can be accessed via the ENC_LATCH_RIS and ENC_LATCH_FAL registers.	9	9
Encoder position latch rising edge	FD1.xEC, FD2.1E, FD2.1xC: IN3	Only rising edges capture the multi-turn encoder position. The latched positions can be accessed via the ENC_LATCH_RIS.	n.a.	26
Encoder position latch falling edge	FD1.xEC, FD2.1E, FD2.1xC: IN3	Only falling edges capture the multi-turn encoder position. The latched positions can be accessed via the ENC_LATCH_FAL.	n.a.	27
Cycle, sequence selection	FD1.1E: IN1, 2 FD1.xEC, FD2.1E, FD2.1xC: IN3, 4	Binary selection of the cycle/sequence	11	11
Delta stop	FD1.1E: IN1, 2 FD1.xEC, FD2.1E, FD2.1xC: IN3, 4	Refer to delta stop cycle description.	12: NO 13: NC	12: NO 13: NC

Tab. 5 – Multipurpose inputs configuration

A digital low pass filter combined with a Schmitt trigger with 2.1 ms maximum delay has been applied to these two input channels in order to reduce noise effects. The low pass filter is removed when these inputs are set as delta stop, encoder latch or direction input frequency (setup time = 200 µs).

### 6.3. Aux inputs

FD2.1E is provided with additional inputs: IN5, IN6, IN7

Function	Drive: Input	Description	Config. Reg. value
Cycle, sequence selection	FD2.1E: IN5, IN6, IN7	Binary selection of the cycle/sequence	11

Tab. 6 – IN5, IN6, IN7 configuration

Other inputs configurations are available upon request.

*A digital low pass filter combined with a Schmitt trigger with 2.1 ms maximum delay has been applied to this input channel in order to reduce noise effects.*

### 6.4. Analog input

FD2.1C, FD2.1AC, FD2.1EC are equipped with IN4 that can be configured as digital or analog input. When used as analog input, several functions are available upon request.

Function	Drive: Input	Description	Config. Reg. value
Speed scaling	FD2.1xC: IN4	Except when operating in interpolation or CSP mode, the speed at which movements are executed is proportional to the value of the analog input.	30

Tab. 7 - Analog input configuration

## 6.5. Outputs

OUT1 is a digital output. When used to monitor the drive status, it is recommended to configure it in normally closed mode (i.e., 0 = Drive ok). However, it can also be configured as normally open (i.e., 1 = Alarm), depending on the application requirements.

On FD1.1E, which has only one output, this output is used for all available functions.

On models such as FD1.xEC, FD2.1E, and FD2.1xC, which are equipped with multiple outputs, different outputs are assigned to different functions and may have varied configurations.

Function	Drive: Output	Description	Config. Reg. value
Drive ok	FD1.1E: OUT1	The output drive ok is high and it opens in case of alarm (recommended)	10
	FD1.xEC, FD2.1E, FD2.1xC: OUT1		0
Alarm	FD1.1E: OUT1	The output alarm is normally opened and it high in case of alarm	11
	FD1.xEC, FD2.1E, FD2.1xC: OUT1		1
Running	FD1.1E: OUT1 FD1.xEC, FD2.1E, FD2.1xC: OUT2	The output is high during running cycles and sequences, opened when stopped.	0
Encoder index 50 ms	FD1.1E: OUT1 FD1.xEC, FD2.1E, FD2.1xC: OUT2	The output is high for 50 ms when the encoder position corresponds to the zero-encoder position. <i>Note: the zero-encoder position can be adjusted using the POSITION_OFFSET register</i>	1
Encoder index $\Pi$	FD1.1E: OUT1 FD1.xEC, FD2.1E, FD2.1xC: OUT2	The output is high for half revolution and low for the other half. The transition from low to high occurs at zero-encoder position. <i>Note: the zero-encoder position can be adjusted using the POSITION_OFFSET register</i>	2
Position uploaded	FD1.1E: OUT1 FD1.xEC, FD2.1E, FD2.1xC: OUT2	The output is high if the power-off position is restored exactly after power-on (ref. to 9 POWER-ON POSITION RESTORE (QUASI-ABSOLUTE MULTI-TURN)).	3
Axis calibrated	FD1.1E: OUT1 FD1.xEC, FD2.1E, FD2.1xC: OUT2	The output is high if the homing cycle completed successfully.	4
Power on position ok	FD1.1E: OUT1 FD1.xEC, FD2.1E, FD2.1xC: OUT2	The output is high if the power-off position is within the limits defined by the POWER_ON_CALIBRATION_LIMIT register after power-on (ref. to 9 POWER-ON POSITION RESTORE (QUASI-ABSOLUTE MULTI-TURN)).	5
Torque limit	FD1.1E: OUT1 FD1.xEC, FD2.1E, FD2.1xC: OUT2	The output is high when the motor cannot follow the demand position.	6
Cycle running + not delta stop	FD1.1E: OUT1 FD1.xEC, FD2.1E, FD2.1xC: OUT2	This output activates at every start movement, similar to normal cycle running signal, and deactivates at the end of the movement, except during delta stop cycles. If delta stop cycle is running, the output deactivates only if delta stop input gets active during movement. If the sensor is not detected, the motor stops after reaching the position parameter steps, but the output remains active. A host or PLC managing this output should implement a time-out for the detection of signal deactivation.	7
Forced	FD1.1E: OUT1 FD1.xEC, FD2.1E,	The output can be forced to control or actuate an external device.	8: L = Opened 9: H = High



	FD2.1xC: OUT2		
ECAT frame IRQ	FD1.1E: OUT1 FD1.xEC, FD2.1E, FD2.1EC: OUT1, OUT2	The output turns on when the frame is received and turns off after RPDOs are read from the ESC	31
ECAT DC SYNC	FD1E: OUT1 FD1.xEC, FD2.1E, FD2.1EC: OUT1, OUT2	The output turns on when a SYNC signal is received and turns off after TPDOs are written into the ESC	32

Tab. 8 - Digital outputs

FD2E implements also OUT3, which can be used as torque feedback (refer to torque off description).

## 7. ALARMS

In normal operation Modbus, CANopen, EtherCAT error registers values are zero, status word alarm bit is off, output drive ok is closed, the status green LED is flashing and the alarm red LED is off.

If any of below alarm condition is triggered, status word, output, Modbus, CANopen and EtherCAT error registers and LEDs signal the anomaly, motor gets immediately de-energized and the shaft freed.

In case of alarm, the red LED number of consequent blinks identifies the Modbus ERR\_FAT register value (when steady lit, it expresses 1, step accumulation limit).

### 7.1. Steps accumulation limit

Modbus ERR\_FAT = 1  
EtherCAT/CANopen error register = 0x81  
EtherCAT/CANopen error code = 0x8611

FD drivers are equipped with a magnetic encoder that measures the position of a shaft-mounted magnet.

When the motor cannot execute the ordered steps—for example, due to insufficient torque to overcome the load or a mechanical blockage—the difference between the rotor position and the rotating magnetic field position increases. The current control loop continues to produce maximum motor torque, accumulating the unexecuted steps while attempting to achieve the commanded position. If the difference between commanded and executed steps exceeds a programmable threshold, the "step accumulation limit" alarm is triggered, and the drive stops.

This threshold of alarm is defined in Modbus as the ACCUMULATION\_LIMIT register and in EtherCAT/CANopen as 0x6065 following error window. The limit represents the maximum allowable position deviation, expressed in steps, and is application-dependent. It is limited at 10 revolutions (128'000 steps when configured with a resolution of 12'800 steps/rev), default value is one revolution.

When the load torque is high enough to cause loss of synchronism with the commanded steps, a position lag take place, but the motor continues to provide maximum output torque to win the load. As soon as this happens, the red LED turns on, and the torque limit bits are set high (Modbus status word bit 4, EtherCAT/CANopen status word bit 8).

Until, eventually, the step accumulation limit is reached and then:

- The motor is de-energized.
- The Modbus ERR\_FAT register is set to 1.
- The EtherCAT/CANopen status word fault bit is activated (error register = 0x81, error code = 0x8611).
- The red LED remains steadily lit, and the Drive ok output turns off.

### 7.2. Over-Temperature Alarm

Modbus ERR\_FAT = 2  
EtherCAT error register = 0x09  
EtherCAT/CANopen error code = 0x4310

FD drives are equipped with a temperature sensor. If the temperature exceeds 100 °C, an over-temperature alarm is triggered, and the drive stops.

The motor's temperature is influenced by factors such as current settings, applied voltage, duty cycle, and rotational speed. Power dissipation in the motor's copper coils due to the Joule effect increases quadratically with motor current. Therefore, to prevent overheating, it is important to avoid oversizing the current settings.

As the rotational speed increases, the frequency of the magnetic field also increases, bringing power dissipation through eddy currents

(Foucault currents) in the iron. To minimize the risk of overheating, do not oversize the speed reduction ratio of the gearbox.

In the event of an over-temperature alarm:

- The motor is de-energized
- The Modbus ERR\_FAT register is set to 2.
- The EtherCAT status word fault bit is activated (error register = 0x09, error code = 0x4310).
- The red LED flashes twice, followed by a 2 s pause and the Drive ok output turns off.

### 7.3. Short Circuit Alarm

Modbus ERR\_FAT = 3  
EtherCAT error register = 0x03  
EtherCAT/CANopen error code = 0x2300

This is a latched hardware fault that occurs when the current exceeds the drive's maximum allowable value, such as during a short circuit in the motor wiring.

In the event of a short circuit alarm:

- The motor is de-energized.
- The Modbus ERR\_FAT register is set to 3.
- The EtherCAT status word fault bit is activated (error register = 0x03, error code = 0x2300).
- The red LED flashes three times, followed by a 2 s pause and the Drive ok output turns off.

*Note:*

*This hardware alarm can only be reset by turning the device off and then back on.*

### 7.4. Over-Voltage Alarm

Modbus ERR\_FAT = 4  
EtherCAT error register = 0x05  
EtherCAT/CANopen error code = 0x3210

This is a latched hardware fault that occurs due to over-voltage on the input DC power supply. This alarm may be triggered by rapid deceleration of a large inertial load combined with an insufficient external power supply capacitor.

In the event of an over-voltage alarm:

- The motor is de-energized.
- The Modbus ERR\_FAT register is set to 4.
- The EtherCAT status word fault bit is activated (error register = 0x05, error code = 0x3210).
- The red LED flashes four times, followed by a 2 s pause and the Drive ok output turns off.

FD overvoltage alarms are set as:

- FD1.1E, FD1.xEC 88 V<sub>DC</sub>,
- FD2.1E, FD2.xC 135 V<sub>DC</sub>.

### 7.5. Data Error

Modbus ERR\_FAT = 5  
EtherCAT error register = 0x21  
EtherCAT/CANopen error code = 0x6320

This alarm is triggered at power-on if incorrect factory settings are detected in the flash memory. The main reason of this fault is a failure during flash programming. This alarm cannot be reset and requires contacting the supplier for assistance.

In the event of this alarm:

- The motor is de-energized.
- The Modbus ERR\_FAT register is set to 5.
- The EtherCAT status word fault bit is activated (error register = 0x21, error code = 0x6320).
- The red LED flashes five times, followed by a 2 s pause and the Drive ok output turns off.

### 7.6. Under Voltage

Modbus ERR\_FAT = 7  
EtherCAT error register = 0x05  
EtherCAT/CANopen error code = 0x3220

Logic power supply configuration:

$V_{EXT}$  powers the drive logic. Drives have different implementations:

FD1.1EC, FD2.1E, FD2.1xC logic is powered exclusively from  $V_{EXT}$ .

FD1.1E and FD1.2EC logic is powered by both  $V_{EXT}$  and  $V_{POW}$  in parallel, enabling operation even with  $V_{POW}$  alone when needed.

Provided no other alarms (e.g., step accumulation limits) are already active and the drive is powered, when  $V_{POW}$  drops below  $17 V_{DC}$ , movement is halted, and an under-voltage alarm is triggered.

Once the motor power supply  $V_{POW}$  is restored above  $20 V_{DC}$ , new Modbus enable motor current or CANopen/EtherCAT fault reset + shutdown + enable operation transitions shall be ordered. As long as  $V_{EXT}$  persists, if the axis was calibrated via a homing cycle, the calibration remains valid even after power recovery.

In the event of this alarm:

- The motor is de-energized.
- Modbus ERR\_FAT register is set to 7.
- The EtherCAT status word fault bit is activated (error register = 0x05, error code = 0x3220).
- The red LED flashes 7 times, followed by a 2 s pause and the "Drive OK" output turns off.

## 7.7. Encoder error

Modbus ERR\_FAT = 11, 12

EtherCAT error register = 0x21

EtherCAT/CANopen error code = 0x7320, 7321

To ensure proper torque control, the drive undergoes precise encoder and motor tuning during manufacturing, which cannot be performed by the user.

Any attempt to mount or dismount the board can alter its position from the factory-tuned alignment. As this tuning is highly precise, even small deviations can render it invalid.

Improper motor mounting can also invalidate the tuning. For example, if the motor shaft is fixed to the hub first, and then the motor flange screws are tightened, the shaft may be pulled back into the motor. This reduces the distance between the encoder IC and the magnet on the motor shaft, leading to tuning invalidation and potential damage to the motor and encoder IC.

Certain motors are equipped with a spring mechanism that restores the shaft to its original position. In motors with glued bearings, any misalignment requires returning the motor to Auxind for repair.

The encoder IC monitors the distance to the magnet by measuring the magnetic field. If the communication with the encoder fails or the magnetic field is out of range, the encoder signals an error.

ENC\_STATUS register can be used to verify encoder status.

Two levels of alarm are available. When ERR\_FAT is equal to 11 or error code is equal to 0x7320, the alarm can be reset, the motor can run, even if the performances might be bad. When ERR\_FAT is equal to 12 or error code is equal to 0x7321 the magnetic field sensed from the encoder IC is out of range, the alarm cannot be reset, and the motor shall be returned for repair.

In the event of this alarm:

- The motor is de-energized.
- Modbus ERR\_FAT register is set to 11/12
- The EtherCAT status word fault bit is activated (error register = 0x21, error code = 0x7320/7321).
- The red LED flashes 11/12 times, followed by a 2 s pause and the "Drive OK" output turns off.

## 7.8. Alarm Resetting

In order to reset software alarms, the following possibilities are offered:

- Fault reset (EtherCAT/CANopen bit 7 of 0x6040, control word)
- Alarm Reset (Modbus EXE\_FUN = 15)
- Disable and enable current (using multipurpose inputs configured as disable/enable current, or by EXE\_FUN = 16 then EXE\_FUN = 17)
- System reset (EXE\_FUN = 14) (refresh of RAM data from flash)
- ROL or ROR (EXE\_FUN = 1 or 2) (refresh of RAM data from flash)
- Turn V<sub>EXT</sub> off and on (refresh of RAM data from flash)

Fault/Alarm reset, disable/enable current are preferred, because after a System Reset, ROL/ROR and V<sub>EXT</sub> off/on the alarm is reset, but the RAM memory will be re-initialized to flash programmed data, as it happens during a normal power on.

Status	LEDs	Error register
Drive ok	Red LED OFF Green LED blinking: 5 Hz communication ON 0.5 Hz communication OFF	Error = 0
Programming mode <i>Bootstrap</i>	Red and green LEDs blinking alternatively: 5 Hz communication ON 0.5 Hz communication OFF	Error = 0 Modbus status word bit 18 high
Alarm	Red LED blinking (number of blinks represents the alarm code) Green LED same as Drive ok status	Error ≠ 0
Encoder error	Green LED short blink at 0,25 Hz Red LED flashes 11/12 times, followed by a 2 s pause	Error = 0 ENC_STATUS ≠ 0

Tab. 9 – Diagnostic LEDs and Errors

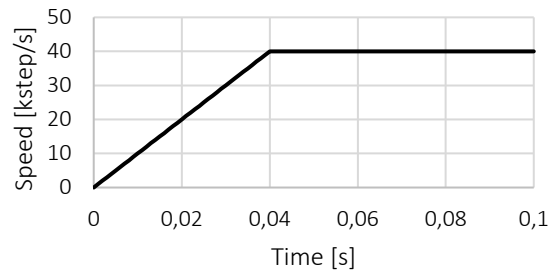
## 8. RAMP PROFILES

V6 self generates linear, parabolic and s-curved speed ramps.

Linear ramp:

ACCELERATION and DECELERATION registers represent the speed increment / decrement every milli-second.

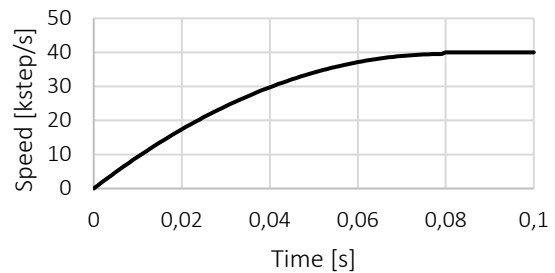
$$t_{acc} = \frac{V}{acc}$$



Parabolic ramp:

ACCELERATION and DECELERATION registers represent the speed increment / decrement per milli-second at zero speed. Increment and decrement are reduced at higher speeds.

$$t_{acc} = \frac{2V}{acc}$$

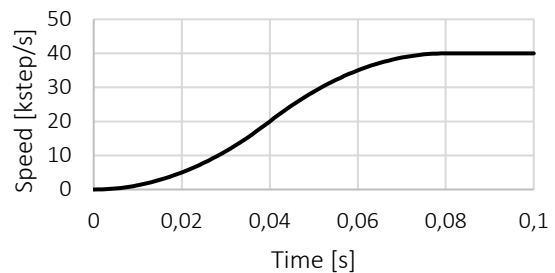


Parabolic ramps, characterized by lower speed variations at higher speeds, are preferred in high speed and inertial load applications.

S-curve ramps:

ACCELERATION and DECELERATION registers represent the speed increment / decrement per milli-second at half speed. Increment and decrement are reduced at lower and higher speeds.

$$t_{acc} = \frac{2V}{acc}$$

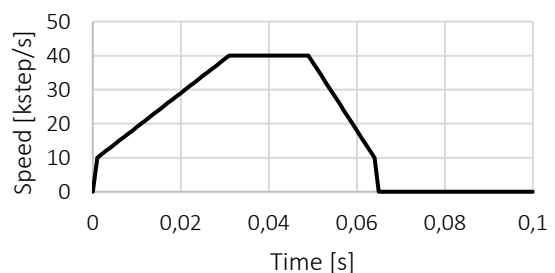


Being the acceleration and deceleration a continuous function of time, motion is characterized by very smooth speed variations with the benefit of mechanical vibrations, wear and tear reduction.

Above graphs represent speed – expressed as thousands of steps per second – as a function of time. For all of them regime speed of 40'000 step/s and acceleration of 1'000'000 step/s<sup>2</sup> have been used (ACCELERATION = 1000).

Start Stop Frequency defines the minimum frequency used in the ramp for both acceleration and deceleration calculation. This value must be set into the START\_STOP\_FREQ register and it shall be enabled setting the bit 9 of CONFIG register. Lower speeds are always available even if a higher start stop frequency is used.

In the graph regime speed of 40'000 step/s, start stop frequency of 10'000 step/s, acceleration of 1'000'000 μstep/s<sup>2</sup> and deceleration of 2'000'000 step/s<sup>2</sup>.



When high acceleration is requested, starting boost functionality will allow a fast increment of motor current at motor start.

## 9. POWER-ON POSITION RESTORE (QUASI-ABSOLUTE MULTI-TURN)

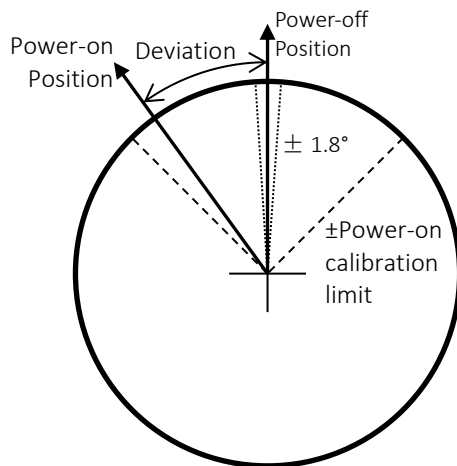


Fig. 6 - Position restore angles

The drive implements the quasi-absolute multi-turn functionality, i.e., provide position data over multiple revolutions of a shaft. Unlike fully absolute multi-turn encoders, which continuously and permanently track the exact position even during power loss, quasi-absolute encoders rely on calculations to reconstruct the absolute multi-turn position after a power cycle.

Before the driver is completely powered off, V6 saves the single-turn encoder position and the drive status.

During the subsequent power on, the system can compare the newly detected single-turn encoder position with the previously saved position. This comparison allows the recovery of the multi-turn position and the drive status from the prior power cycle.

The following options are available for this process:

### 9.1. Multi-turn position upload

Setting bit 11 POS\_UPLOAD of CONFIG register the exact power-off position and status can be recovered at power-on.

Stepper motors have a detent torque, which moves the rotor in the position of minimum reluctance when the drive is off (if the detent torque is bigger than the resistant torque). Thanks to the power off position saving, when the drive is turned on, it compares the new absolute encoder position with the saved one. If the motor movement is inside  $\pm 1.8^\circ$  threshold, it energizes the motor with the exact position before power off: the rotating field position and the multi-turn position registers will be exactly the same, as if it has never been powered off, Modbus STATUS\_WORD bit 11 AX\_CALIBRATED, and bit 6 TARGET\_POS are restored and bit 10 POS\_UPLOADED is set; EtherCAT 0x6041 bit 14 position uploaded is set, bit 12 homing attained in homing mode and bit 10 target reached in profile position mode are restored.

### 9.2. Multi-turn position corrected by deviation

With bit 11 POS\_UPLOAD of CONFIG register set, but position deviation bigger than  $\pm 1.8^\circ$  threshold, or with bit 11 POS\_UPLOAD of CONFIG register reset, a second verification takes place: the position deviation will be compared with the programmable POWER\_ON\_CALIBRATION\_LIMIT register. If the deviation is inside the programmed limit, the multi-turn position and the current configuration will be corrected using such deviation; status word AX\_CALIBRATED bit will be restored, POWER\_ON\_POS\_OK bit will be set and POS\_UPLOADED bit will be reset.

### 9.3. Position inside the revolution

When both previous verification results are negative (the motor has been moved when off beyond power-on calibration limit), the new position value will be the absolute encoder position inside the revolution (i.e. not a multi-turn position), the AX\_CALIBRATED and POS\_UPLOADED bits will be reset.

It is possible to verify this feature moving the shaft when the motor is off: once powered on in a position beyond the POWER\_ON\_CALIBRATION\_LIMIT the multi-turn position is lost.

Setting POWER\_ON\_CALIBRATION\_LIMIT to zero and resetting bit 11 POS\_UPLOAD of CONFIG register, the power on position will always be the absolute encoder position inside the revolution.

## 10. ETHERCAT

This chapter describes the installation, setup, range of functions and EtherCAT protocols for FD Auxind drives, whose code includes the suffix "E".

### 10.1. Introduction

EtherCAT is an open high-speed fieldbus system that conforms to Ethernet (IEEE 802.3) for real-time distributed control, now standardized as IEC 61158.

EtherCAT is a registered trademark of Beckhoff Automation GmbH (Germany). EtherCAT technology is protected by patents.

Each node achieves a short cycle time by transmitting Ethernet frames at high speed. A mechanism that allows sharing clock information enables high-precision synchronization control with low communications jitter.

The EtherCAT communication speed is up to 100 Mbps full duplex and can include a maximum of 65,535 stations in a single network configuration without using switches.

The frame is originated by the master and it is transmitted to the first slave in the network, traditionally connected in a ring topology. Each slave mounts an EtherCAT Slave Controller (ESC) that processes "on-the-fly" the frame. Each node in the network reads and writes the data from the frame as it passes through them. The frame is then sent back to the master, that can collect all the data written from the slaves.

Following graph describes a network of EtherCAT slaves in a ring topology. The Master controls the traffic in the network by initiating the transactions.

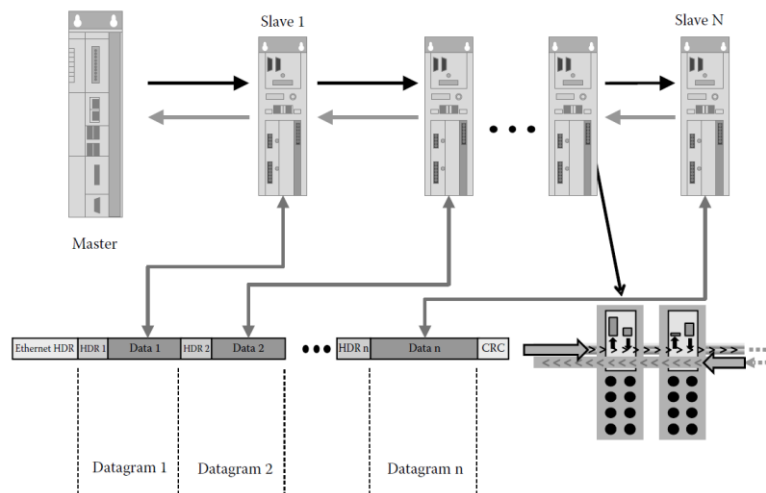


FIGURE 18.1 EtherCAT typical topology, with the *on-the-fly* frame processing.

Fig. 7 - EtherCAT ring topology

The frames only delay by a fraction of a micro-second in each node. Via EtherCAT, the entire network can be addressed with just one frame.

EtherCAT telegram is encapsulated in an Ethernet frame and includes one or more EtherCAT datagrams destined to the EtherCAT slaves. Each datagram is a command that consists of a header, data and a working counter. The header and data are used to specify the operation that the slave must perform, and the working counter is updated by the slave to let the master to know that a slave has processed the command.

A control system requires the following in periodic time intervals:

- Inputs: messages from the slaves to the master such as status word, position actual value, speed, analogue or digital inputs, etc.
- Outputs: messages from the master to the slave with control word, target position, target speed, etc.

The specific nature of the data transferred via the network depends on the mode of operation of the slave drive.

## 10.2. Main specifications

Item	Specification
Communications standard	IEC 61158 Type 12, IEC 61800-7 CiA 402 Drive Profile
Physical layer	100BASE-TX (IEEE802.3)
Connectors	FD1.1E: Circular connectors M8, 4 poles, A-coded, female FD1.1EC, FD1.2EC, FD2.1E, FD2.xEC: Circular connectors M12, 4 poles, D-coded, female
Communications media	Ethernet Category 5 (100BASE-TX) or higher (twisted-pair cable with double, aluminum tape and braided shielding) is recommended.
Communications distance	Distance between each node: 100 m max.
Process data	Configurable PDO mapping
Mailbox (CoE)	SDO requests, SDO responses, and SDO information
Distributed clock (DC)	Synchronization in DC mode. 1 ms, 2 ms, 4 ms
CiA402 Drive Profile	Cyclic synchronous position mode (CSP) Cyclic synchronous velocity mode (CSV) Profile position mode (PP) Profile velocity mode (PV) Interpolated position mode (IP) Homing mode (HM)

Tab. 10 - EtherCAT main specifications

## 10.3. Node address settings

The node address DIP switches (1-7) are used to set the EtherCAT node address.

DIP switch settings	Description
0	The EtherCAT master sets the node address on E2PROM
1 to 127	DIP switch setting is used as node address

Tab. 11 - node addressing

Configured Station Alias Register 0x0012:

If DIP switches value is 0 the address is taken from register 0x0012, loaded from EEPROM, written from EtherCAT master. If DIP switches value is different than 0, the DIP switches value is copied to configured station alias register during initialization phase.

Requesting ID mechanism:

FD drives use AL Status Code register 0x0134 for Device Identification value. The EtherCAT master can request the Device Identification value by setting bit 5 of AL Control register (0x0120.5 = ID Request). FD drive loads the current Device Identification Value to the AL Status Code register and set bit 5 of AL Status register (0x0130.5 = ID loaded) to indicate that its Device Identification Value is loaded. A changed Device Identification value, e.g. because the DIP switches are set to a different value, is loaded only with a new ID Request by the EtherCAT Master.



## 10.4. EtherCAT LED indicators

Name	Color	Status	Description
Run	Green	Flickering	Bootstrap
		Off	Init
		Blinking	Pre-operational
		Single flash	Safe-operational
		On	Operational
Link/Activity IN/OUT	Green	Off	Link not established in physical layer
		On	Link established in physical layer, without activity
		Flickering	In operation

Tab. 12 - EtherCAT LEDs

Flickering is 50 ms on, 50 ms off.

Blinking is 200 ms on, 200 ms off.

## 10.5. CANopen over EtherCAT (CoE)

The structure of CANopen application over EtherCAT is represented in the following graph:

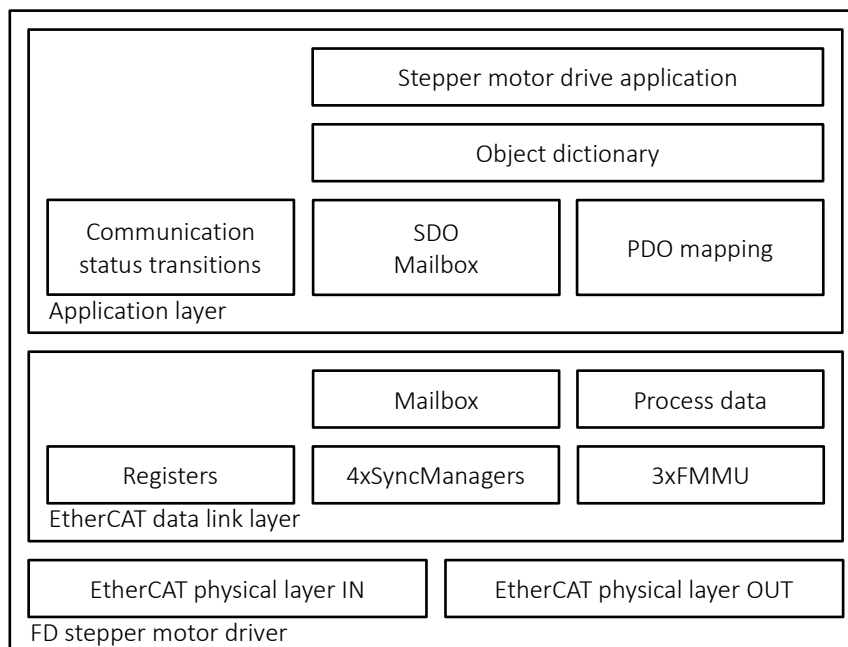


Fig. 8 - CoE structure

Normally, multiple protocols can be transmitted using EtherCAT. The IEC 61800-7 (CiA 402) drive profile is used.

The object dictionary in the application layer contains parameters and application data as well as information on the PDO mapping between the process data servo interface and stepper motor drive application.

The process data object (PDO) consists of objects in the object dictionary that can be mapped to the PDO. The contents of the process data are defined by the PDO mapping.

Process data communications cyclically read and write the PDO. Mailbox communications (SDO), instead, uses asynchronous message communications where all objects in the object dictionary can be read and written.

## 10.6. EtherCAT Slave Controller (ESC)

FD1E, FD1EC, FD2E and FD2EC are equipped with LAN9252 from Microchip, which is used as a 2 port ESC with dual integrated ethernet PHY. Each PHY contains a full-duplex 100BASE-TX transceiver and support 100 Mbps operation. The LAN9252 supports HP Auto-MDIX, allowing the use of direct connect or cross-over LAN cables.

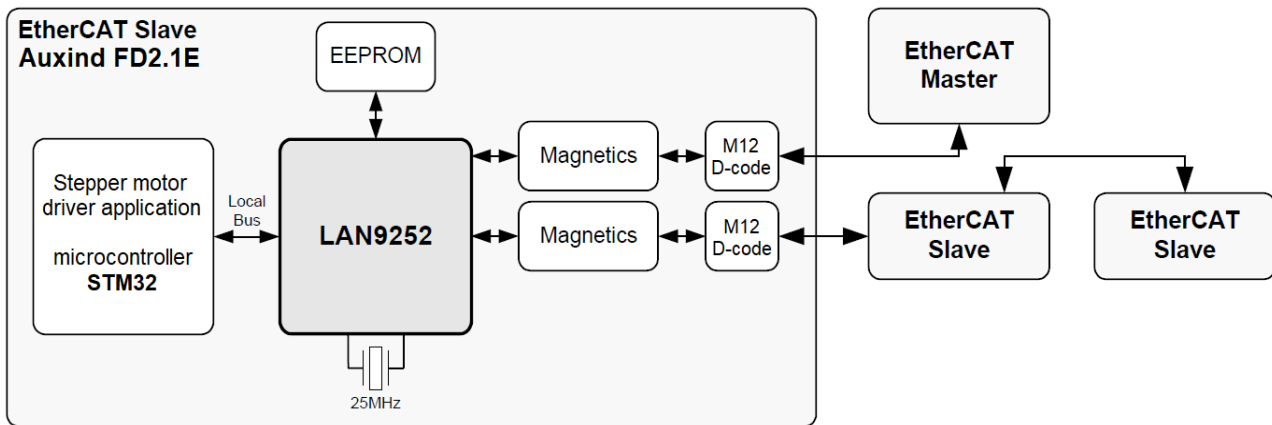


Fig. 9 - EtherCAT slave block diagram

The LAN9252 includes 4 KB of Dual Port memory (DPRAM) and 3 Fieldbus Memory Management Units (FMMUs). Each FMMU performs the task of mapping logical addresses to physical addresses.

The EtherCAT slave controller includes 4 SyncManagers to allow the exchange of data between the EtherCAT master and the stepper motor driver application. Each SyncManager direction and mode of operation is configured by the EtherCAT master.

Two modes of communication are available: buffered mode or mailbox mode.

In the buffered mode, both the local microcontroller and EtherCAT master can write to the device concurrently. The buffer within the LAN9252 will always contain the latest data. If newer data arrives before the old data can be read out, the old data will be dropped. This mode is used for PDO.

In mailbox mode, access to the buffer by the local microcontroller and the EtherCAT master is performed using handshakes, guaranteeing that no data will be dropped. This mode is used for SDO.

### 10.6.1. FMMU

Fieldbus Memory Management Units (FMMU) convert logical addresses into physical addresses by the means of internal address mapping. Thus, FMMUs allow to use logical addressing for data segments that span several slave devices: one datagram addresses data within several arbitrarily distributed ESCs. Each FMMU channel maps one continuous logical address space to one continuous physical address space of the slave.

### 10.6.2. SyncManagers

LAN9252 is equipped with 4 KB of DPRAM. Stepper motor driver application and the master access this common memory area to communicate. SyncManagers enable consistent and secure data exchange between the EtherCAT master and the application and they are capable to generate interrupts to inform both sides of changes.

FD stepper motor drivers are equipped with 4 SyncManagers, the communication direction (in / out) and mode (mailbox / buffered) are configured as:

SyncManager 0 = master mailbox output  
 SyncManager 1 = master mailbox input  
 SyncManager 2 = master buffered output  
 SyncManager 3 = master buffered input

The mailbox mode implements a handshake mechanism for data exchange, so that no data will be lost. Each side, EtherCAT master or drive application, will get access to the buffer only after the other side has finished its access. At first, the producer writes to the buffer. Then, the buffer is locked for writing until the consumer has read it out. Afterwards, the producer has write access again, while the buffer is locked for the consumer. The mailbox mode is typically used for application layer protocol (SDO).

The mailbox mode only allows alternating reading and writing. This assures all data from the producer reaches the consumer. The mailbox mode uses just one buffer of the configured size. At first, after initialization/activation, the mailbox is writeable. Once it is written completely, write access is blocked, and the buffer can be read out by the other side. After it was completely read out, it can be written again. The time it takes to read or write the mailbox does not matter.

On the other hand, the buffered mode allows both sides, EtherCAT master and drive application, to access the communication buffer at any time. The consumer always gets the latest consistent buffer which was written by the producer, and the producer can always update the content of the buffer. The buffered mode allows writing and reading data simultaneously without interference. If the buffer is written faster than it is read out, old data will be dropped. The buffered mode is typically used for cyclic process data (PDO).

The buffered mode is also known as 3-buffer-mode. Physically, 3 buffers of identical size are used for buffered mode. The start address and size of the first buffer is configured in the SyncManager configuration. The addresses of this buffer have to be used by the master and the local application for reading/writing the data. Depending on the SyncManager state, accesses to the first buffer (0) address range are redirected to one of the 3 buffers.

One buffer of the three is allocated to the producer (for writing), one buffer to the consumer (for reading), and the third buffer keeps the last consistently written data of the producer.

## 10.7. EtherCAT functional overview

Each port receives an Ethernet frame, performs frame checking and forwards it to the next port. Time stamps of received frames are generated when they are received. The Loop-back function of each port forwards Ethernet frames to the next logical port if there is either no link at a port, or if the port is not available, or if the loop is closed for that port. The Loop-back function of port 0 forwards the frames to the EtherCAT Processing Unit. The loop settings can be controlled by the EtherCAT master.

Packets are forwarded in the following order: Port 0 → EtherCAT Processing Unit → Port 1.

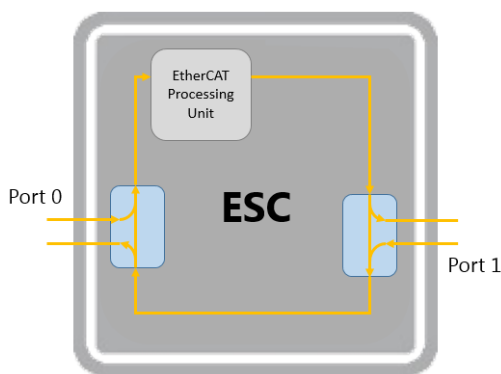


Fig. 10 - EtherCAT slave controller

The EtherCAT Processing Unit (EPU) receives, analyses and processes the EtherCAT data stream. The main purpose of the EtherCAT Processing unit is to enable and coordinate access to the internal registers and the memory space of the ESC, which can be addressed both from the EtherCAT master and from the drive application. Data exchange between master and slave application is comparable to a dual-ported memory (process memory), enhanced by special functions.

Distributed Clocks (DC) allow for precisely synchronized generation of output signals and input sampling, as well as time stamp generation of events.

## 10.8. EtherCAT state machine

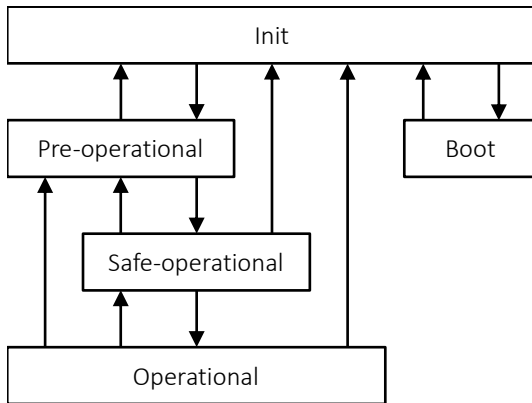


Fig. 11 - EtherCAT state machine

The EtherCAT State Machine (ESM) of the EtherCAT slave is controlled by the EtherCAT Master and it is responsible for the coordination of master and slaves at start up and during operation.

In the following table, the available services in each state (RPDO are the PDO received: written from the master, read from the slave, e.g. control word; TPDO are the PDO transmitted: read from the master, written from the slave, e.g. status word).

State	SDO	RPDO	TPDO	Description
Bootstrap	X	X	X	In bootstrap state, the drive is able to accept a new firmware or parameters downloaded with the FoE protocol
Init	X	X	X	Communications are being initialized. Communications are not possible. The master uses init state to initialize a set of configuration register. Mailbox sync managers are also configured in this state.
Pre-operational	✓	X	X	Only mailbox communications are possible in this state. This state is entered after initialization has been completed and mailbox has been setup. It is used to initialize network settings (configuration of SM and PDO mapping)
Safe-operational	✓	X	✓	In this state, PDO transmissions (not reception) are possible in addition to mailbox communications. DC mode cyclic communications can be used to send information such as status from the drive.
Operational	✓	✓	✓	This is a normal operating state. DC mode cyclic communications can be used to control the motor

Tab. 13 - EtherCAT states

State changes are requested by the master. The master requests a write to AL control register, which results in a register event in the drive. The drive responds to the change in 0x0120, AL control register through 0x0122, AL status register after a successful or failed state change. If the requested state change failed, the drive responds with the error flag set in AL status and with AL status code register that details the error occurred.

Requests of state transitions through 0x0120, AL control register are expressed as per following table:

Parameter	Data type	Value
State	4 bits	1: init 2: pre-operational 3: boot 4: safe-operational 8: operational
Acknowledge	1 bit	0: parameter Change of slave AL status register remains unchanged 1: reset parameter Change of slave AL status register
ID request	1 bit	0: ID is not requested 1: request of ID
reserved	2 bits	
reserved	1 Byte	

Tab. 14 - AL control

The drive answers via 0x0130, AL status register:

Parameter	Data type	Value
State	4 bits	1: init 2: pre-operational 3: boot 4: safe-operational 8: operational
Change	1 bit	0: state transition successful 1: state transition not successful
ID loaded	1 bit	0: AL status code value does not represent ID 1: AL status code value represents drive ID
reserved	2 bits	
reserved	1 Byte	

Tab. 15 - AL status

In case of not successful state transitions, the drive provides the reason on 16 bits, 0x0134, AL status code register

Code	Description	Current state	Resulting state
0x0000	No error	Any	Any
0x0011	Invalid request state change	I→S, I→O, P→O, O→B, S→B, P→B	I + error, P + error, S + error
0x0012	Unknown requested state	Any	I + error, P + error, S + error
0x0013	Boot not supported	I→B	I + error
0x0016	Invalid mailbox configuration	I→P	I + error
0x0017	Invalid sync manager configuration	P→S, S→O	Current state + error
0x001B	Sync manager watchdog	O, S	S + error
0x001D	Invalid output configuration	O, S, P→S	P + error
0x001E	Invalid input configuration	O, S, P→S	P + error
0x0036	Invalid SYNC0 cycle time	P→S	P + error
0x0037	Invalid SYNC1 cycle time	P→S	P + error
0x0061	Device identification value updated	P	P + error

Tab. 16 - AL status code

## 10.9. SDO

With the SDO services the entries of the object dictionary can be read or written.

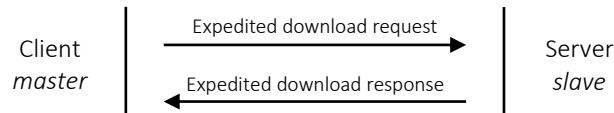
The first Byte of the first segment specifies the flow control information (download, upload, size, expedited, etc.), the next three Bytes specifies the index and sub-index, the last bytes are size or data.

When the data size exceeds the mailbox size, other segments shall be transmitted. In such segments, the first Byte specifies control information and remaining Bytes, the data.

The receiver, confirms each segment or each block of segments, so that a peer-to-peer communication (client/server) takes place.

### 10.9.1. SDO download expedited

Following graph shows the primitives between client and server in case of a successful single SDO download expedited sequence:



All the information can be grouped in standard SDO frame part, made of 8 Bytes.

The message sent from the master to the slave:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	0x0A
	Address	Word	Slave address
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x02, SDO request
SDO	Size indicator	1 b	0x01, Data set size is specified
	Transfer type	1 b	0x01, Expedited transfer
	Data set size	2 b	0x00, 4 Bytes 0x01, 3 Bytes 0x02, 2 Bytes, 0x03, 1 Byte
	Complete access	1 b	0x00, entry addressed with index and sub-index will be downloaded 0x01, complete object will be downloaded, sub-index shall be zero (sub-index 0 included) or one (sub-index 0 excluded)
	Command specifier	3 b	0x01, download request
	Index	Word	Index of the object
	Sub-index	Byte	Sub-index of the object 0 or 1 if complete access is used
	Data	4 Bytes	Data of the object, unused Bytes shall be 0

Tab. 17 - SDO request

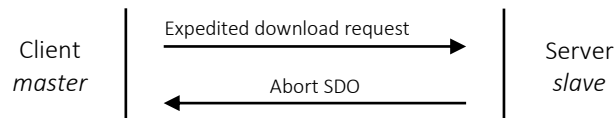
The successful answer from the slave to the master:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	0x0A
	Address	Word	0x00
	Channel	6 b	0x00
	Priority	2 b	0x00, lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x03, SDO response
SDO	Size indicator	1 b	0x00
	Transfer type	1 b	0x00
	Data set size	2 b	0x00
	Complete access	1 b	0x00
	Command specifier	3 b	0x03, download response
	Index	Word	Index of the object
	Sub-index	Byte	Sub-index of the object 0 or 1 if complete access is used
	reserved	4 Bytes	

Tab. 18 - SDO response

### 10.9.2. SDO abort

The unsuccessful answer is through an Abort SDO transfer, whose primitives are represented in the following graph:



Frame part	Data field	Type	Description
Mailbox header	Length	Word	0x0A
	Address	Word	0x00
	Channel	6 b	0x00
	Priority	2 b	0x00, lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x02, SDO request
SDO	Size indicator	1 b	0x00
	Transfer type	1 b	0x00
	Data set size	2 b	0x00
	Complete access	1 b	0x00
	Command specifier	3 b	0x04, abort transfer request
	Index	Word	Index of the object
	Sub-index	Byte	Sub-index of the object
	Abort code	4 Bytes	Abort code specified in following table

Tab. 19 - SDO abort

The implemented abort codes are:

Name	Code	Meaning
TOGGLE_NOT_CHANGED	0x05030000	Toggle bit not changed
COMMAND_NOT_VALID	0x05040001	Client/server command specifier not valid or unknown
INVALID_BLOCK_SIZE	0x05040002	Block size not supported
UNSUPPORTED	0x06010000	Unsupported access to an object
WRITE_ONLY	0x06010001	Attempt to read a write only object
READ_ONLY	0x06010002	Attempt to write a read only object
SUBIX_ZERO_NOT_ZERO	0x06010003	Sub-index cannot be written, sub-index 0 must be 0 for write access
COMPLETE_ACC_NOT_SUPP	0x06010004	Complete access not supported for this object
OBJ_NOT_EXIST	0x06020000	The object does not exist in the object dictionary
OBJ_NOT_MAPPABLE	0x06040041	The object cannot be mapped into the PDO
MAPPING_TOO_LONG	0x06040042	The number and length of the objects to be mapped would exceed the PDO length
PARAM_INCOMPATIBLE	0x06040043	General parameter incompatibility reason
GEN_INTERN_INCOMP	0x06040047	General internal incompatibility in the device
STORE_FAILED	0x06060000	Error in parameters storing
DATA_TYPE LENGHT NOT_OK	0x06070010	Data type does not match, length of service does not match
SUB_IX_NOT_EXIST	0x06090011	Sub-index does not exist
WRITE_RANGE_EXCEEDED	0x06090030	Value range of parameter exceeded (only for write access)
VALUE_TOO_HIGH	0x06090031	Value of parameter written too high
VALUE_TOO_LOW	0x06090032	Value of parameter written too low
GENERAL_ERROR	0x08000000	General error
NO_TRANSFER_GENERAL	0x08000020	
NO_TRANSFER_LOCAL	0x08000021	
NOT_IN_THIS_STATE	0x08000022	Data cannot be transferred or stored to the application because of present device state

Tab. 20 - SDO abort codes

### 10.9.3. SDO download normal

With this type of service, the length is variable, doing so the data can be transferred all in one frame if the mailbox is big enough, otherwise via segmented transfers.

The SDO download normal request sent from the master to the slave is:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	$n \geq 0x0A$ , length of mailbox service data
	Address	Word	Slave address
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x02, SDO request
SDO	Size indicator	1 b	0x01, Data set size is specified
	Transfer type	1 b	0x00, normal transfer
	Data set size	2 b	0x00
	Complete access	1 b	0x00, entry addressed with index and sub-index will be downloaded 0x01, complete object will be downloaded, sub-index shall be zero (sub-index 0 included) or one (sub-index 0 excluded)
	Command specifier	3 b	0x01, download request
	Index	Word	Index of the object
	Sub-index	Byte	Sub-index of the object 0 or 1 if complete access is used
	Complete size	4 Bytes	Complete data size of the object
	Data	n-10 Bytes	If $((\text{Length}-10) \geq \text{complete size})$ : data of the object



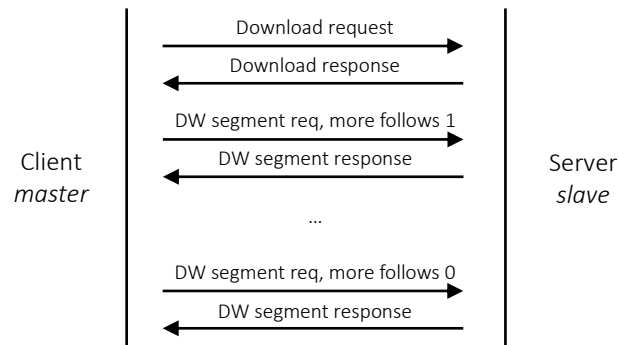
			If ((Length-10) < complete size): first data part of the object, Download SDO Segment is following
--	--	--	--

Tab. 21 - SDO download request

The attribute types and coding of SDO download normal response are the same as of SDO download expedited response.

When the dimension of the downloaded object is larger than the dimension of the mailbox, segmented transfers take place.

In case of segment, following primitive is implemented:



The SDO segment request sent from the master to the slave is:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	$n \geq 0x0A$ , length of mailbox service data
	Address	Word	Slave address
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x02, SDO request
SDO	More follows	1 b	0x00: Download SDO Segment is following 0x01: last Download SDO Segment
	SegData size	3 b	Defines how much of the last 7 Bytes (which always has to be send) contain data (only applicable if Length is 0x0A, otherwise shall be 0x00): 0x00: 7 Bytes 0x01: 6 Bytes 0x02: 5 Bytes 0x03: 4 Bytes 0x04: 3 Bytes 0x05: 2 Bytes 0x06: 1 Byte 0x07: 0 Bytes
	Toggle	1 b	Shall toggle with every Download SDO Segment Request, starting with 0x00
	Command specifier	3 b	0x00: Download Segment Request
	Data	n-3 Bytes	data of the object

Tab. 22 - SDO segment download request

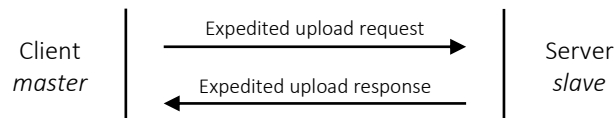
The SDO segment response sent from the slave to the master is:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	0x0A, length of mailbox service data
	Address	Word	0x00
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x03, SDO response
SDO	reserved	4 b	0x00
	Toggle	1 b	Shall be the same as for the corresponding Download SDO Segment Request
	Command specifier	3 b	0x01: Download Segment Response
	Data	7 Bytes	0x00

Tab. 23 - SDO segment download response

#### 10.9.4. SDO upload expedited

Following graph shows the primitives between client and server in case of a successful single SDO upload expedited sequence:



All the information can be grouped in standard SDO frame part, made of 8 Bytes.

The message sent from the master to the slave:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	0x0A
	Address	Word	Slave address
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x02, SDO request
SDO	Reserved	4 b	0x00
	Complete access	1 b	0x00, entry addressed with index and sub-index will be uploaded 0x01, complete object will be uploaded, sub-index shall be zero (sub-index 0 included) or one (sub-index 0 excluded)
	Command specifier	3 b	0x02, upload request
	Index	Word	Index of the object
	Sub-index	Byte	Sub-index of the object 0 or 1 if complete access is used
	Reserved	4 Bytes	0x00

Tab. 24 - SDO upload request

The successful answer from the slave to the master:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	0x0A
	Address	Word	0x00
	Channel	6 b	0x00
	Priority	2 b	0x00, lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x03, SDO response
SDO	Size indicator	1 b	0x01, size of data in data set size specified
	Transfer type	1 b	0x01, expedited transfer
	Data set size	2 b	0x00, 4 Bytes 0x01, 3 Bytes 0x02, 2 Bytes, 0x03, 1 Byte
	Complete access	1 b	0x00, entry addressed with index and sub-index will be downloaded 0x01, complete object will be uploaded, sub-index shall be zero (sub-index 0 included) or one (sub-index 0 excluded)
	Command specifier	3 b	0x02, upload response
	Index	Word	Index of the object
	Sub-index	Byte	Sub-index of the object 0 or 1 if complete access is used
	Data	4 Bytes	Data of the object

Tab. 25 - SDO upload response

### 10.9.5. SDO upload normal

With this type of service, the length is variable, doing so the data can be transferred all in one frame if the mailbox is big enough, otherwise via segmented transfers.

The SDO upload normal request sent from the master to the slave is the same as SDO upload expedited request:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	0x0A
	Address	Word	Slave address
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x02, SDO request
SDO	Reserved	4 b	0x00
	Complete access	1 b	0x00, entry addressed with index and sub-index will be uploaded 0x01, complete object will be uploaded, sub-index shall be zero (sub-index 0 included) or one (sub-index 0 excluded)
	Command specifier	3 b	0x02, upload request
	Index	Word	Index of the object
	Sub-index	Byte	Sub-index of the object 0 or 1 if complete access is used
	Reserved	4 Bytes	0x00

Tab. 26 - SDO upload request

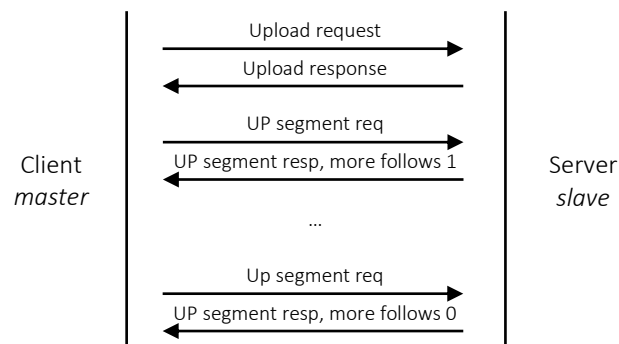
The SDO upload normal response from the slave to the master, if the message fits in the mailbox is:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	$n \geq 0x0A$ , length of mailbox service data
	Address	Word	0x00
	Channel	6 b	0x00
	Priority	2 b	0x00, lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x03, SDO response
SDO	Size indicator	1 b	0x01
	Transfer type	1 b	0x00, normal transfer
	Data set size	2 b	0x00
	Complete access	1 b	0x00, entry addressed with index and sub-index will be downloaded 0x01, complete object will be uploaded, sub-index shall be zero (sub-index 0 included) or one (sub-index 0 excluded)
	Command specifier	3 b	0x02, upload response
	Index	Word	Index of the object
	Sub-index	Byte	Sub-index of the object 0 or 1 if complete access is used
	Complete size	4 Bytes	Complete size of data of the object
	Data	n-10 Bytes	If $((\text{Length}-10) \geq \text{Complete Size})$ : data of the object If $((\text{Length}-10) < \text{Complete Size})$ : first data part of the object, upload SDO segment is following

Tab. 27 - SDO upload response

When the dimension of the uploaded object is larger than the dimension of the mailbox, segmented transfers take place.

In case of segment, following primitive is implemented:



The upload SDO segment request sent from the master to the slave is:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	0x0A
	Address	Word	Slave address
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x02, SDO request
SDO	Reserved	4 b	0x00
	Toggle	1 b	Shall toggle with every upload SDO segment request, starting with 0x00
	Command specifier	3 b	0x03: Upload Segment Request
	Reserved	7 Bytes	

Tab. 28 - SDO upload segment request

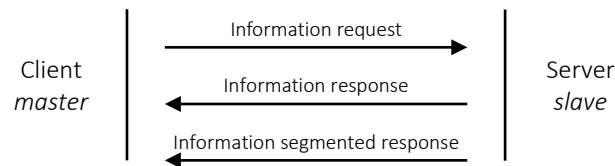
The upload SDO segment response sent from the slave to the master is:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	n ≥ 0x0A, length of mailbox service data
	Address	Word	0x00
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x03, SDO response
SDO	More follows	1 b	0x00: Upload SDO Segment is following 0x01: last Upload SDO Segment
	SegData Size	3 b	Defines how much of the last 7 Bytes (which always has to be send) contain data (only applicable if Length is 0x0A, otherwise shall be 0x00): 0x00: 7 Bytes 0x01: 6 Bytes 0x02: 5 Bytes 0x03: 4 Bytes 0x04: 3 Bytes 0x05: 2 Bytes 0x06: 1 Byte 0x07: 0 Bytes
	Toggle	1 b	Shall be the same as for the corresponding Upload SDO Segment Request
	Command specifier	3 b	0x00: Upload Segment Response
	Data	n-3 Bytes	Data part of the object

Tab. 29 - SDO upload segment response

### 10.9.6. SDO information

With SDO information services the object dictionary of a server can be read by a client. The primitives of the SDO information services are mapped to the primitives of the mailbox transmission services.



Frame part	Data field	Type	Description
Mailbox header	Length	Word	n > 0x06: length of the mailbox service data
	Address	Word	0x00
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x08, SDO information
SDO info header	Opcode	7 b	0x01, get OD list request 0x02, get OD list response 0x03, get object description request 0x04, get object description response 0x05, get entry description request 0x06, get entry description response 0x07, SDO info error request
	Incomplete	1 b	0: last SDO information fragment 1: SDO information fragments will follow
	reserved	8 b	
	Fragments left	Word	Number of fragments which follow
SDO info service data	Data	Byte[n-6]	SDO information service data

Tab. 30 - SDO information

### 10.9.7. Get OD list

Get OD list request is as follows:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	n = 0x08: length of the mailbox service data
	Address	Word	Slave address
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x08, SDO information
SDO info header	Opcode	7 b	0x01, get OD list request
	Incomplete	1 b	0x00
	Reserved	8 b	0x00
	Fragments left	Word	0x00
SDO info service data	List type	Word	0x00: get number of objects in the 5 different lists 0x01: all objects of the object dictionary shall be delivered in response The other options are not supported.

Tab. 31 - Get OD request

The response, containing the number of entries or the list is:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	n >= 0x08: length of the mailbox service data
	Address	Word	0x00
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x08, SDO information
SDO info header	Opcode	7 b	0x02, get OD list response
	Incomplete	1 b	0: last SDO information fragment 1: SDO information fragments will follow
	Reserved	8 b	0
	Fragments left	Word	Number of fragments which follow SDO info header will be sent with every fragment, SDO info data will be sent fragmented
SDO info service data	List type	Word	0x00: get number of objects in the 5 different lists 0x01: all objects of the object dictionary shall be delivered in response The other options are not supported.
	Index list	Words [n-8/2]	5 words with the length of the list types if list type is 0. List of object indexes if list type is 1.

Tab. 32 - Get OD response

The SDO info service data of the get OD list response is segmented and the data are fragmented.

### 10.9.8. Get object description

Get object description request is as follows:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	n = 0x08: length of the mailbox service data
	Address	Word	Slave address
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x08, SDO information
SDO info header	Opcode	7 b	0x03, get object description request
	Incomplete	1 b	0x00
	reserved	8 b	0x00
	Fragments left	Word	0x00
SDO info service data	Index	Word	Index of the requested object description

Tab. 33 - Get OD request

The response is:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	n > 0x0A: length of the mailbox service data
	Address	Word	0x00
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x08, SDO information
SDO info header	Opcode	7 b	0x04, get object description response
	Incomplete	1 b	0: last SDO information fragment
	reserved	8 b	0x00
	Fragments left	Word	Number of fragments which follow
SDO info service data	Index	Word	Index of object
	Data type	Byte	Reference to data type list
	Max sub-index	Byte	Maximum number of sub-indexes of the object
	Object code	Byte	7: variable 8: array 9: record
	Name	Char[n-12]	Name of the object

Tab. 34 - Get OD response

Data type list implemented:

Index (hex)	Type
0x0002	Integer 8
0x0003	Integer 16
0x0004	Integer 32
0x0005	Unsigned 8
0x0006	Unsigned 16
0x0007	Unsigned 32
0x0009	Visible string
0x001B	Unsigned 64

Tab. 35 - Data type list



### 10.9.9. Get entry description

Get entry description request is as follows:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	0x0A: length of the mailbox service data
	Address	Word	Slave address
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x08, SDO information
SDO info header	Opcode	7 b	0x05, get entry description request
	Incomplete	1 b	0x00
	Reserved	8 b	0x00
	Fragments left	Word	0x00
SDO info service data	Index	Word	Index of the requested object description
	Sub-index	Byte	Sub-index of the requested object description
	Value info	Byte	The value info includes which elements shall be in the response: b3: unit type b4: default value b5: minimum value b6: maximum value

Tab. 36 - Get entry description request

The response is as follow:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	n >= 0x10: length of the mailbox service data
	Address	Word	0x00
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x08, SDO information
SDO info header	Opcode	7 b	0x06, get entry description response
	Incomplete	1 b	0: last SDO information fragment 1: SDO information fragments will follow
	Reserved	8 b	0
	Fragments left	Word	Number of fragments which follow
SDO info service data	Index	Word	Index of object
	Sub-index	Byte	Sub-index of the requested object description
	Max sub-index	Byte	Maximum number of sub-indexes of the object
	Value info	Byte	The value info includes which elements shall be in the response: b3: unit type b4: default value b5: minimum value b6: maximum value
	Data type	Word	Index of the data type of the object
	Bit length	Word	Bit length of the object
	Object access	Word	b0: read in preop b1: read in safeop b2: read in op b3: write in preop b4: write in safeop

			b5: write in op b6: mappable in RPDO b7: mappable in TPDO
	Data	Byte[n-16]	If requested: The unit type of the object (double word) The default value (same data type of the object) The minimum value (same data type of the object) The maximum value (same data type of the object) If the length is greater than the described response parameter, the description is following (array of char)

Tab. 37 - Get entry description response

### 10.9.10. Unit types

Unit types are 32 bits information, defined in ETG.1004, organized as per following scheme:

31	24	23	16	15	8	7	0	
Prefix				Numerator		Denominator		reserved
MSB								LSB

Prefixes implemented are:

Name	Power	Code
-	10 <sup>0</sup>	0x00
kilo	10 <sup>3</sup>	0x03
centi	10 <sup>-2</sup>	0xFE
milli	10 <sup>-3</sup>	0xFD
nano	10 <sup>-9</sup>	0xF7

Tab. 38 - unit prefix

Following notation index, expressed as numerator and denominator, are implemented:

Name	Notations	Code
-	-	0x00
seconds	s	0x03
Ampere	A	0x04
Hertz	Hz	0x20
Volt	V	0x26
Celsius	°C	0x2D
seconds square	s <sup>2</sup>	0x57

Tab. 39 - Unit notations

### 10.9.11. SDO info error

This service is used from the slave to signal to the master that something is wrong in its request.

Frame part	Data field	Type	Description
Mailbox header	Length	Word	0x0A: length of the mailbox service data
	Address	Word	0x00
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x08, SDO information
SDO info header	Opcode	7 b	0x08, SDO info error request
	Incomplete	1 b	0x00
	Reserved	8 b	0x00
	Fragments left	Word	0x00
SDO info service data	Abort code	4 Bytes	Abort code

Tab. 40 - SDO info error

### 10.9.12. Complete access

Compared to CAN, where the number of Bytes for SDO protocol is limited to 8 only, EtherCAT allows larger mailbox size. Auxind drives implements 128 Bytes in reception on SyncManager 0 and 128 Bytes in transmission on SyncManager 1 (headers included).

Thanks to larger mailbox a complete record can be read/written within a single service, this is achieved through complete access: all sub-indexes of an object are uploaded or downloaded with a single SDO service, while with CAN it is possible to read a single sub-index at a time. Consequently, complete access minimizes boot-up time of the device.

Following rules are used for the correct alignment of the data:

1. Sub-index 0 is padded to 16 bits
2. Sub-index of bit data type (BOOLEAN and BIT1 to BIT7) will be filled over the byte border, the next non-bit data type will start at the next BYTE address e.g. object 0x10F3 SI4 = BOOL, SI5 = UINT16 → Bit-Offset of SI5 = 48 (not 41)
3. To define gaps in the byte stream of a complete access, it is allowed to define gap entries (Data\_Type = 0, Bit\_Length = gap size in bits) The value of gaps shall be 0.
4. Sub-indexes that don't exist will not need space
5. The complete access can start with sub-index 0 or sub-index 1, other sub-indexes are not allowed
6. All objects that fit in the mailbox are accessible by SDO with complete access
7. With segmented SDO transfer also bigger objects are accessible by SDO with complete access
8. Objects with dynamic entries (i.e. "OCTET\_STRING", "UNICODE\_STRING", "ARRAY\_OF\_\*" or "VISIBLE\_STRING") cannot be accessed by complete access.
9. Objects that cannot be accessed return the error code 0x06010000 (Unsupported Access) or 0x6010004 (Complete access not supported).
10. The length of the responded Complete Access data shall either match to the current byte length of the object or to the Bit Size of the object description.
11. The current maximum sub-index may exceed the number of entry descriptions of the offline object dictionary.
12. The SDO download complete access data length shall always match the full current object size (defined by sub-index 0).
13. Entries of data type string have a fixed length in the entry description to be used for complete access. The string itself can be shorter. Unused Bytes are filled with 0.

SDO Complete Access Example with SubIndex 1			SDO Complete Access Example with SubIndex 0		
Sub-index	Data type	Bit offset	Sub-index	Data type	Bit offset
1	BOOLEAN	0	0	UNSIGNED8	0
2	BIT2	1	1	BOOLEAN	16
3	BIT3	3	2	BIT2	17
4	0 = Gap, Length = 2	6	3	BIT3	19
10	UNSIGNED8	8	4	0 = Gap, Length = 2	22
11	UNSIGNED16	16	10	UNSIGNED8	24
12	UNSIGNED32	32	11	UNSIGNED16	32
13	UNSIGNED8	64	12	UNSIGNED32	48
14	OCTET-STRING[4]	72	13	UNSIGNED8	80
15	VISIBLE-STRING[7]	104	14	OCTET-STRING[4]	88
16	INTEGER32	160	15	VISIBLE-STRING[7]	120
17	UNSIGNED64	192	16	INTEGER32	176

Tab. 41 - Complete access examples

### 10.9.13. Encapsulated SDO

Since SDOs are treated with lower priority in the driver firmware compared to PDOs, issues may arise under certain conditions. Specifically, when operating with very low cycle times (e.g., 1 or 2 ms) and a large quantity of data mapped to PDOs, the drive might not have enough time to handle all SDO traffic effectively.

When working in CSP or CSV modes, only a few objects typically need to be mapped, such as the control word, status word, target position/velocity, position actual value, and velocity actual value. However, if additional data needs to be exchanged at high frequencies, encapsulated SDOs can be utilized to manage this efficiently.

Encapsulated SDOs allow the entire object dictionary to be accessed via PDOs. To implement encapsulated SDOs, you need to map two 8 Bytes objects:

- 0x200C:1 Encapsulated SDO request (RPDO)
- 0x200C:2 Encapsulated SDO response (TPDO)
- 

The exchange of information follows the standard SDO download/upload expedited process:

Data field	Type	Description
Size indicator	b0	0x01, to specify dataset size
Transfer type	b1	0x01, Expedited transfer
Data set size	b2, b3	0x00, 4 Bytes 0x01, 3 Bytes 0x02, 2 Bytes, 0x03, 1 Byte
Toggle	b4	The download of the content will happen when this bit commutates
Command specifier	b5, b6, b7	0x01, download request
Index	2 Bytes	Index of the object
Sub-index	1 Byte	Sub-index of the object
Data	4 Bytes	Data of the object, unused Bytes shall be 0

Tab. 42 - Encapsulated SDO download expedited request

Data field	Type	Description
Size indicator	b0	0x00
Transfer type	b1	0x00
Data set size	b2, b3	0x00
Toggle	b4	Request toggle
Command specifier	b5, b6, b7	0x03, download response
Index	Word	Index of the object
Sub-index	Byte	Sub-index of the object
reserved	4 Bytes	0

Tab. 43 - Encapsulated SDO download expedited

Data field	Type	Description
Size indicator	b0	0x00
Transfer type	b1	0x00
Data set size	b2, b3	0x00
Reserved	b4	0x00
Command specifier	b5, b6, b7	0x02, upload request
Index	2 Bytes	Index of the object
Sub-index	1 Byte	Sub-index of the object
reserved	4 Bytes	0

Tab. 44 - Encapsulated SDO upload expedited request

Data field	Type	Description
Size indicator	b0	0x01, Data set size is specified
Transfer type	b1	0x01, Expedited transfer
Data set size	b2, b3	0x00, 4 Bytes 0x01, 3 Bytes 0x02, 2 Bytes, 0x03, 1 Byte
Reserved	b4	0x00
Command specifier	b5, b6, b7	0x02, upload response
Index	2 Bytes	Index of the object
Sub-index	1 Byte	Sub-index of the object
Data	4 Bytes	Data of the object, unused Bytes are 0

Tab. 45 - Encapsulated SDO upload expedited response

If something goes wrong an abort message is transmitted.

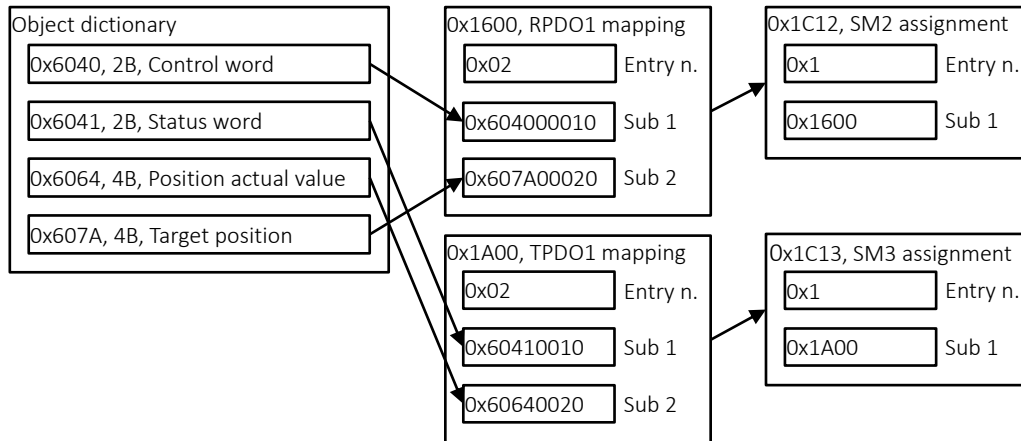
Data field	Type	Description
Size indicator	b0	0x00
Transfer type	b1	0x00
Data set size	b2, b3	0x00
Reserved	b4	0x00
Command specifier	b5, b6, b7	0x04, Abort
Index	2 Bytes	Index of the object
Sub-index	1 Byte	Sub-index of the object
Data	4 Bytes	Abort code

Tab. 46 - Encapsulated SDO abort response

## 10.10. PDO mapping

The process data objects, PDOs, are used to transfer data during cyclic communications in real-time. PDO can be reception PDO (RPDO), which receive data from the controller, or transmission PDO, (TPDO), which send status from the drive to the host controller. The EtherCAT application layer can hold multiple objects to enable process data. The contents of the process data are described in the PDO mapping object and the SyncManager PDO assignment object, addressed in following indexes via SDO:

- 0x1600 – 0x1603 Receive PDO 1 to 4 mapping
- 0x1A00 – 0x1A03 Transmit PDO 1 to 4 mapping
- 0x1C12, 0x1C13 SyncManager 2 and 3 PDO assignment



Mapping parameter configured in RPDO and TPDO mapping is made of:

31	16	15	8	7	0
Index		Sub-index		Number of bits	
MSB				LSB	

To change the content of these objects the following procedure shall be done:

1. set sub-index 0 = 0. The object is considered disabled when sub-index 0 has the value 0.
2. configure the mapping information in the mapping entries sub-index 1 ... 8
3. set sub-index 0 = number of used mapping entries

Each PDO mapping record can also be configured using complete access in a single SDO download service.

FD drives implement 4 RPDO, and 4 TPDO each of them contains a maximum number of 8 objects for a total of 8 Bytes, and a total limit of 4\*64 Bytes in reception and 4\*64 Bytes in transmission.

The object mapping can be changed only when the EtherCAT communications state is in pre-operational. Since the mappings is not saved in EEPROM, the master shall transmit the configuration each time the drive is turned on.

## 10.11. Emergency messages

Emergency messages are triggered by the occurrence of drive internal errors. The transmission is executed via the mailbox interface (SyncManager 1).

The message is structured as follows:

Frame part	Data field	Type	Description
Mailbox header	Length	Word	0x0A
	Address	Word	0x00, Master
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x03, CoE
	Counter	3 b	1 – 7
	reserved	1 b	0x00
CoE header	Number	9 b	0x00
	reserved	3 b	0x00
	Service	4 b	0x01, Emergency
Emergency	Error code	Word	Ref. to object 0x603F
	Error register	Byte	Ref. to object 0x1001
	Data	5 Bytes	Error code 0000 – 9FFF: manufacturer specific error field Error code A000 – EFFF: diagnostic data Error code A000 – EFFF: manufacturer specific error field
	reserved	n-10 Bytes	0x00

Tab. 47 - Emergency messages

Refer to 0x603F, error code for the list of implemented codes.

## 10.12. Synchronization

Following synchronization modes are available:

- Free run

Drive's application is not synchronized with EtherCAT

- Synchronous with sync manager (SM2) event

Drive's application is synchronized with SM2 event. SM2 events are based on the time an EtherCAT frame is received. This time can jitter in the range of a few micro-seconds due to the EtherCAT Master implementation (delay in Stack, PHY & MAC Delay, etc).

- Synchronous with DC SYNC event

Slave's application is synchronized to the SYNC0 or SYNC1 event, which are based on the distributed clocks (DC) unit. The jitter could be reduced to a few nano-seconds.

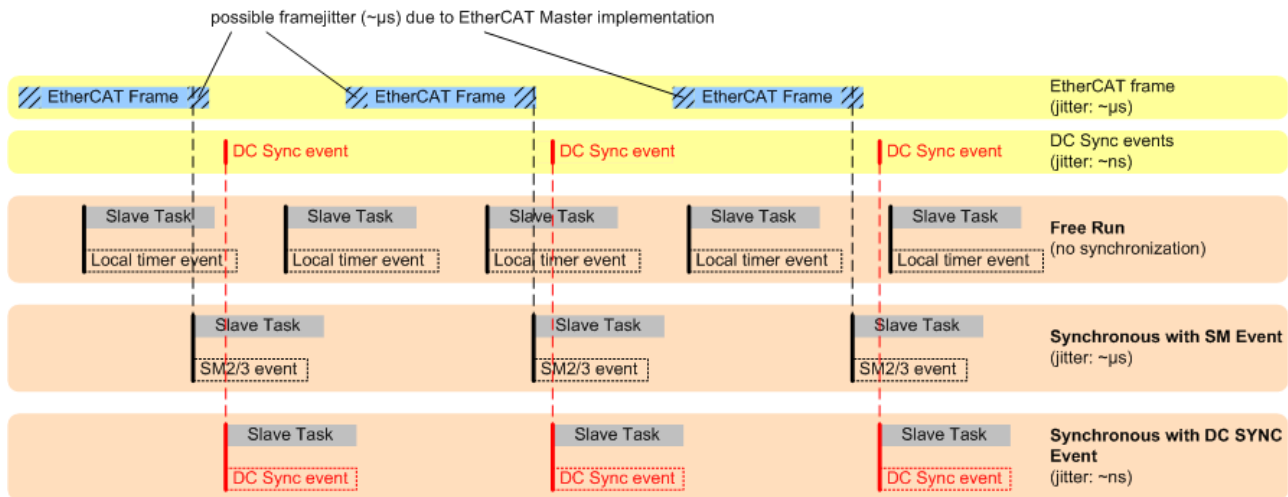


Fig. 12 - EtherCAT frames synchronization

The different synchronization types can be identified by the different combinations of the sub-indexes of 0x1C32 and 0x1C33.

### 10.12.1. Free run

When the drive is controlled using I/O's, it is possible to connect EtherCAT for changing parameter's and monitoring. In this condition there is no need of synchronization with EtherCAT master and the data exchange can be asynchronous.

Nevertheless, this type of functioning is not recommended, synchronization with sync manager (SM2) event is preferred.

In order to operate in free run, objects 0x1C32:1, SM2 synchronization type and 0x1C33:1, SM3 synchronization type shall be written equal to zero.

### 10.12.2. SM synchronous

Drive's process data handling is started when the frame on SM2 is received. This leads to inaccuracy, because:

- cyclic frames are received by the slaves with the same jitter, which affects the master in sending them.
- even with no jitter, due to finite hardware propagation delays the last slaves of the ring topology will receive the cyclic frames later with respect to the first ones



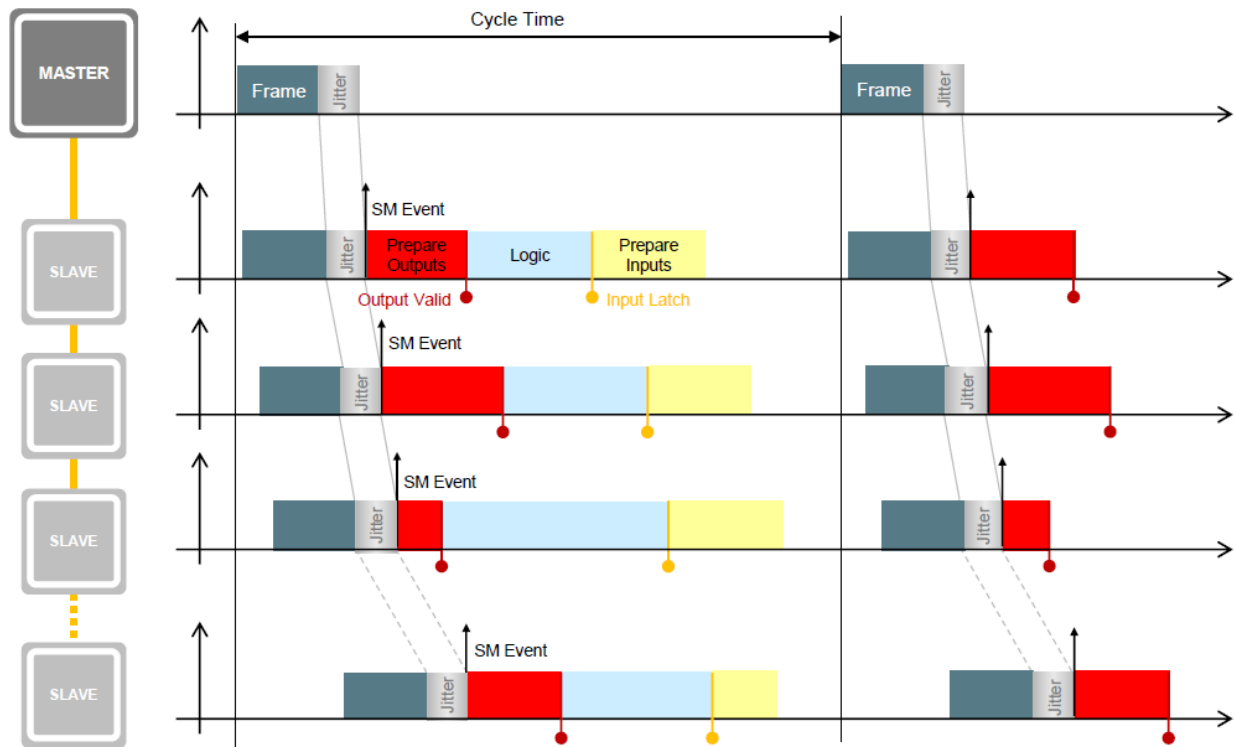


Fig. 13 - Cyclic frames propagation delay

FD drives can withstand jitter on SM event, provided that it is less than a quarter of SM2 cycle time (time period between two frames). This is achieved thanks to internal PLL, capable to create a new time base that is used in cyclic synchronous modes (CSP and CSV). Because modern CPUs have a jitter of approx. 5  $\mu$ s, this mode of functioning is acceptable in most of the cases.

Nevertheless, the propagation delay, that each node of the ring introduces, introduces a frame time shift, which, with a high number of slaves, can cause a not well synchronized network. To give an idea, for simplicity the total slave forwarding delay and the cable propagation delay associated with a single slave can be considered of approx. 1  $\mu$ s.

Considerations on the need to use more complex synchronization methods depends on the number of slaves and application requirements.

On the slave point of view “prepare outputs” is the reading of SM2, once RPDO data is valid, it is immediately processed, (e.g. on cyclic synchronous velocity mode (CSV) 0x60FF, target velocity is set on drive frequency generator). Feedbacks are updated and latched, then written on SM3.

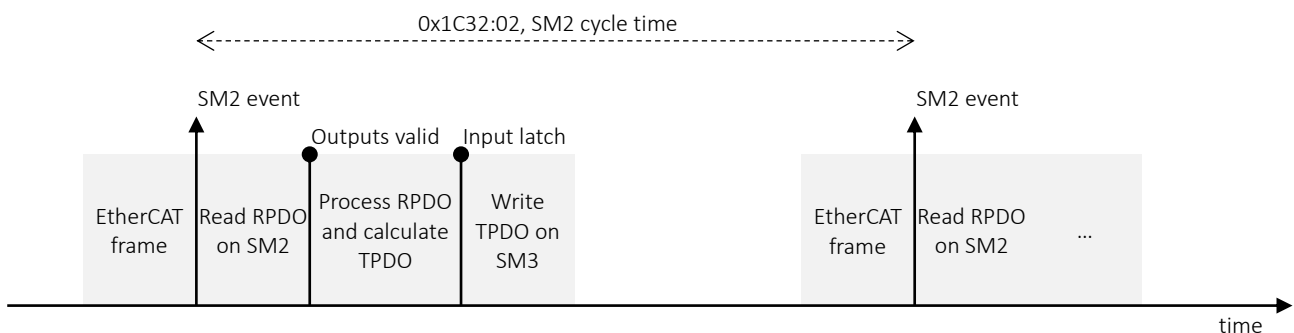


Fig. 14 - EtherCAT frame SM2 time calculation

The cycle time, which is the time period between every frame can be explicit writing object 0x1C32:02, SM2 cycle time, but anyhow the internal PLL is able to calculate autonomously the correct period of SM2.

Reading the object 0x1C32:02, SM2 cycle time allows to monitor the calculated PLL period.

Object 0x1C32:05, SM2 minimum cycle time express the minimum period in ns, that for FD1E and FD2E is 1 ms. This value might depend upon the quantity of Bytes that are mapped into the PDO's.

### 10.12.3. Distributed clocks (DC)

A mechanism named distributed clock (DC) is used to synchronize EtherCAT communications. The DC mode is used to perform highly accurate control in a multi-axis system.

In DC mode, the master and slaves are synchronized by sharing the same clock. Interrupts (SYNC0 and SYNC1) are generated in the slaves at precise intervals based on this clock.

The communications cycle is determined by setting the SYNC0 signal output cycle. The jitter can be reduced to a few nano-seconds.

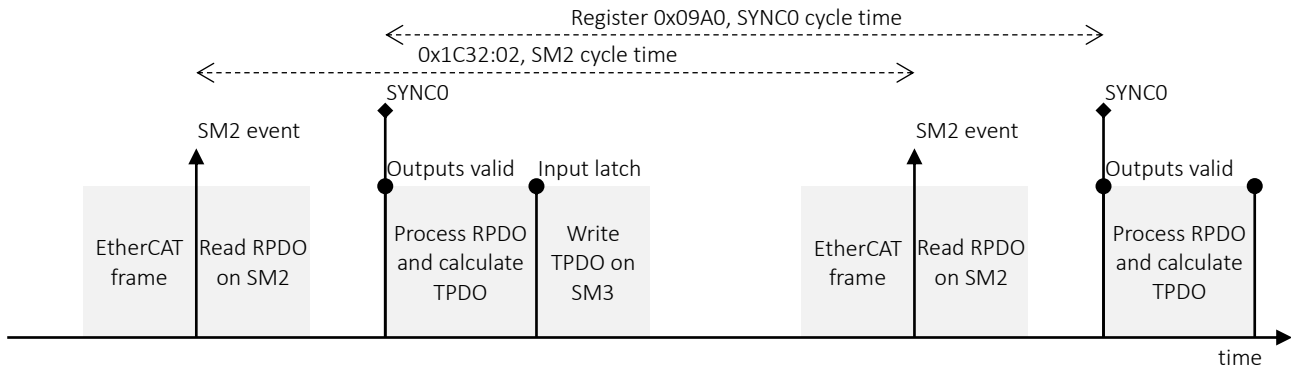


Fig. 15 - EtherCAT frame DC sync time calculation

DC synchronization is more complicated than SM synchronization, but it offers better performances. The SM2 event is affected by jitter of EtherCAT frames ( $\sim\mu\text{s}$ ), while SYNC0's jitter is very small ( $\sim\text{ns}$ ). Furthermore, with DC synchronization all the slaves share a common time base using distributed PLL, that produces the same network SYNC signal to each slave, while on the contrary EtherCAT frames suffers of propagation delay.

It is important that the master transmits the EtherCAT frame with sufficient time before SYNC0 occurs, to allow the slave to read it, otherwise the drive application would not have RPDO data to process when the SYNC arrives.

DC synchronization is activated from the master writing on ESC register

0x1C32:1, SM2 synchronization type is equal to 2, i.e. synchronous with DC SYNC0.

0x1C32:2, SM2 cycle time is taken from master's settings to 0x09A0, SYNC0 cycle time ESC register

### 10.13. Filetransfer over EtherCAT (FoE)

FoE protocol is used for conveying files over EtherCAT. The state machine receives and transmits EtherCAT PDUs with transfer command (Write means load file to slave, Read means load file to master). The primitives of the FoE services are mapped to the primitives of the mailbox transmissions services.

Firmware and parameters can be downloaded into the drive thanks to write request.

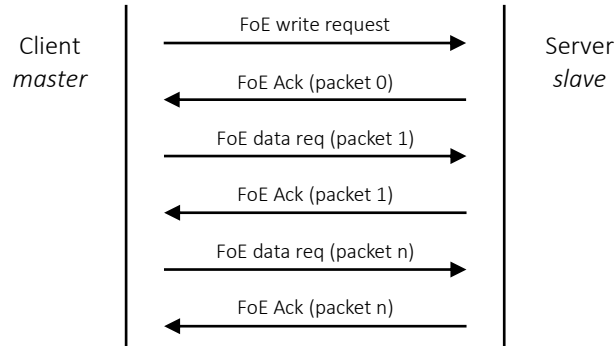


Fig. 16 - FoE write sequence with success

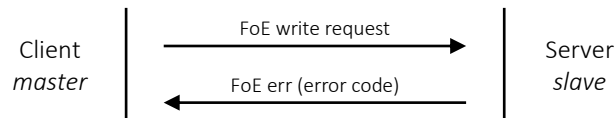


Fig. 17 - FoE write sequence with error

Frame part	Data field	Type	Description
Mailbox header	Length	Word	Length of the mailbox service data
	Address	Word	Slave address
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x04, FoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
FoE header	OpCode	Byte	0x02: Write request
	reserved	Byte	0x00
Write header	Password	DWord	0x00, password unused
	File Name	Char [n-6]	Name of the file to be written. It can start with: "Firmware_Bin" or "Parameters_Bin"

Tab. 48 - FOE write request

Frame part	Data field	Type	Description
Mailbox header	Length	Word	Length of the mailbox service data
	Address	Word	Slave address
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x04, FoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
FoE header	OpCode	Byte	0x03: Data request
	reserved	Byte	0x00
Data header	Packet number	DWord	1... 0xFFFFFFFF
	Data	Byte [n-6]	File data

Tab. 49 - FOE data request

Frame part	Data field	Type	Description
Mailbox header	Length	Word	Length of the mailbox service data
	Address	Word	Slave address
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x04, FoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
FoE header	OpCode	Byte	0x04: Ack request
	reserved	Byte	0x00
Ack header	Packet number	DWord	0: Acknowledge of write request 1...0xFFFFFFFF acknowledge of data request

Tab. 50 - FOE Ack request

Frame part	Data field	Type	Description
Mailbox header	Length	Word	Length of the mailbox service data
	Address	Word	Slave address
	Channel	6 b	0x00
	Priority	2 b	0x00, Lowest priority
	Type	4 b	0x04, FoE
	Counter	3 b	1 – 7, counter of mailbox services
	reserved	1 b	0x00
FoE header	OpCode	Byte	0x05: Error request
	reserved	Byte	0x00
Error header	Error code	DWord	1...0xFFFFFFFF error code
	Error Text	Char [n-6]	Optional error description

Tab. 51 - FOE error request

Error code	Meaning
0x8000	Not defined
0x8001	Not found
0x8002	Access denied
0x8003	Disk full
0x8004	Illegal
0x8005	Packet number wrong
0x8006	Already exists
0x8007	No user
0x8008	Bootstrap only
0x8009	Not bootstrap
0x800A	No rights
0x800B	Program error

Tab. 52 - FOE error codes

## 10.14. ESC registers

Refer to LAN9252 datasheet for further information.

Offset	Size [Bytes]	Bits	Name	R/W	Description
ESC Information					
0x0000	1	7:0	EtherCAT Controller Type	RO	0xC0 = Microchip
0x0001	1	7:0	EtherCAT Controller Revision	RO	0x02
0x0002	2	15:0	EtherCAT Controller Build	RO	0x0000
0x0004	1	7:0	Supported FMMUs	RO	0x03
0x0005	1	7:0	Supported SyncManagers	RO	0x04
0x0006	1	7:0	Process Data RAM Size	RO	0x04
0x0007	1	7:6	Port 3 Configuration	RO	00: Not implemented
		5:4	Port 2 Configuration	RO	01: Not configured
		3:2	Port 1 Configuration	RO	11: MII/RMII
		1:0	Port 0 Configuration	RO	11: MII/RMII
0x0008	2	11	Fixed FMMU/SyncManager Configuration	RO	0: Variable configuration
		10	EtherCAT Read/Write Command Support	RO	0: Supported
		9	EtherCAT LRW Command Support	RO	0: Supported
		8	Enhanced DC SYNC Activation	RO	1: Available
		7	Separate Handling of FCS Errors	RO	1: Supported, frame with wrong FCS and additional nibble will be counted separately in Forwarded RX Counter
		6	Enhanced Link Detection MII	RO	1: Available
		5	Enhanced Link Detection EBUS	RO	0: Not available
		4	Low Jitter EBUS	RO	0: Not available, standard jitter
		3	Distributed Clocks (width)	RO	1: 64-bit
		2	Distributed Clock	RO	1: Available
		0	FMMU Operation	RO	0: Bit oriented
Station Address					
0x0010	2	15:0	Configured Station Address	R/W	0x0000, This field contains the address used for node addressing (FPxx commands)
0x0012	2	15:0	Configured Station Alias Address	RO	0x0000, This field contains the alias address used for node addressing (FPxx commands). The use of this alias is activated by the Station Alias bit of the ESC DL Control Register. EEPROM value is only taken over at first EEPROM load after lower-on reset.
Write Protection					
0x0020	1	0	Write Register Enable	R/W	If write protection is enabled, this register must be written in the same Ethernet frame (value is a don't care) before other writes to this station are allowed. Write protection is still active after this frame (if the Write Register Protection Register is not changed)
0x0021	1	0	Write Register Protection	R/W	0: Protection disabled 1: Protection enabled Note: Registers 0000h-0F0Fh are write protected, except for 0030h.
0x0030	1	0	ESC Write Register Enable	R/W	If ESC write protection is enabled, this register must be written in the same Ethernet frame (value is a don't care) before other writes to this station are allowed. ESC write protection is still active after this frame (if the ESC Write Register Protection Register is not changed)
0x0031	1	0	ESC Write Register Protection	R/W	0: Protection disabled 1: Protection enabled Note: All areas are write protected, except for 0030h.
Data Link Layer					
0x0040	1	7:0	Write: ESC Reset ECAT	W	A reset is asserted after writing 52h ("R"), 45h ("E"),

Offset	Size [Bytes]	Bits	Name	R/W	Description
					and 53h ("S") in this register with 3 consecutive commands.
		1:0	Read: Reset Procedure Progress	R	01: After writing 52h 10: After writing 45h (if 52h previously written) 00: Else
0x0100	4	DL Control Register			
		24	Station Alias	R/W	Default 0: Ignore station alias 1: Alias can be used for all configured address command types (FPRD, FPWR, etc.)
		19	EBUS Low Jitter	R/W	Default 0: Normal jitter 1: Reduced jitter
		18:16	RX FIFO Size/RX Delay Reduction (ESC delays start of forwarding until FIFO is at least half full)		111: Default
		13:12	Loop Port 2	R/W	Default 00: Auto. 01: Auto Close. 10: Open. 11: Closed.
		11:10	Loop Port 1	R/W	Default 00: Auto. 01: Auto Close. 10: Open. 11: Closed.
		9:8	Loop Port 0	R/W	Default 00: Auto. 01: Auto Close. 10: Open. 11: Closed.
		1	Temporary Use of Register 0101h Settings	R/W	0: Permanent Use 1: Temporarily use for ~1 s, then revert to previous settings.
		0	Forwarding Rule	R/W	0: EtherCAT frames are processed, Non-EtherCAT frames are forwarded without processing Default 1: EtherCAT frame are processed, Non-EtherCAT frames are destroyed. The source MAC address is changed for every frame (SOURCE_MAC[1] is set to 1 - locally administered address) regardless of the forwarding rule.
0x0108	2	15:0	Physical Read/Write Offset		Default: 0 Offset of R/W commands (FPRW, APRW) between Read address and Write address. RD_ADR- ADR and WR_ADR = ADR + R/W-offset.
0x0110	2	DL Status Register			
		11	Communication on Port 1	RO	0: No stable communication 1: Communication established
		10	Loop Port 1	RO	0: Open. 1: Closed.
		9	Communication on Port 0	RO	0: No stable communication 1: Communication established
		8	Loop Port 0	RO	0: Open. 1: Closed.
		5	Physical Link on Port 1	RO	0: No link 1: Link detected
		4	Physical Link on Port 0	RO	0: No link 1: Link detected
		2	Enhanced Link Detection	RO	0: Deactivated for all ports 1: Activated for at least one port
		1	PDI Watchdog Status	RO	0: Watchdog expired 1: Watchdog reloaded
		0	PDI Operational/EEPROM Loaded	RO	0: EEPROM not loaded, PDI not operational (no access

Offset	Size [Bytes]	Bits	Name	R/W	Description
			Correctly		to Process Data RAM) 1: EEPROM loaded correctly, PDI operational (access to Process Data RAM)
Application Layer					
0x0120	2	AL control register			
		4	Error Ind Ack	R/W	0: No Ack of Error Ind in AL status register 1: Ack of Error Ind in AL status register
		3:0	Initiate State Transition of Device State Machine	R/W	1: Request Init State 2: Request Pre-Operational State 3: Request Bootstrap State 4: Request Safe-Operational State 8: Request Operational State
0x0130	2	AL status register			
		4	Error Ind	RO	0: Device is in state as requested or Flag cleared by command 1: Device has not entered requested state or changed state as a result of a local action
		3:0	Actual State of the Device State Machine	RO	1: Init State 2: Pre-Operational State 3: Bootstrap State 4: Safe-Operational State 8: Operational State
0x0134	2	15:0	AL status code register	RO	Default: 0
0x0138	1	RUN LED Override register			
		4	RUN Override	R/W	0: Override disabled 1: Override enabled
		3:0	RUN LED Code	R/W	0x0: Off 0x1-0xC: Flash 1x-12x 0xD: Blinking 0xE: Flickering 0xF: On
Interrupts					
0x0200	2	15:0	EtherCAT Event Mask	R/W	ECAT event masking of the ECAT Event Request register Events for mapping into the ECAT event fields of EtherCAT frames. 0: Corresponding ECAT Event Request register bit is not mapped 1: Corresponding ECAT Event Request register bit is mapped
0x0210	2	EtherCAT Event Request			
		7	SyncManager Ch.3 Status Mirror	RO	This bit mirrors the value of the SyncManager Ch.x Status. 0: No Sync Ch.x Event 1: Sync Ch.x Event Pending
		6	SyncManager Ch.2 Status Mirror	RO	
		5	SyncManager Ch.1 Status Mirror	RO	
		4	SyncManager Ch.0 Status Mirror	RO	
		3	AL Status Event	RO	0: No change in AL Status 1: AL Status Change Note: This bit is cleared by reading the AL Status Register from EtherCAT.
		2	DL Status Event	RO	0: No change in DL Status 1: DL Status Change Note: This bit is cleared by reading the ESC DL Status Register from EtherCAT.
		0	DC Latch Event	RO	0: No change on DC Latch Inputs 1: At least one change on DC Latch Inputs Note: This bit is cleared by reading the DC Latch event times from EtherCAT for EtherCAT controlled Latch Units, so that the LATCH0 Status Register/LATCH1 Status Register indicates no event.
Error Counters					

Offset	Size [Bytes]	Bits	Name	R/W	Description
0x0300	2	15:8	Port 0 RX Error Counter	R/WC	Counting is stopped when 0xFF is reached. This register is cleared if any one of the RX Error Counter Registers is written.
		7:0	Port 0 Invalid Frame Counter	R/WC	
0x0302	2	15:8	Port 1 RX Error Counter	R/WC	
		7:0	Port 1 Invalid Frame Counter	R/WC	
0x0308	1	7:0	Port 0 Forwarded RX Error Counter	R/WC	
0x0309	1	7:0	Port 1 Forwarded RX Error Counter	R/WC	
0x030C	1	7:0	ECAT Processing Unit Error Counter	R/WC	Counting is stopped when 0xFF is reached. This field counts the errors of frames passing the Processing Unit (e.g., FCS error or datagram structure error).
0x0310	1	7:0	Port 0 Lost Ling Counter	R/WC	Counting is stopped when 0xFF is reached. This counter only counts if port loop is Auto or Auto-Close. This register is cleared if any one of the Lost Link Counter Registers is written.
0x0311	1	7:0	Port 1 Lost Ling Counter	R/WC	
Watchdogs					
0x0400	2	15:0	Watchdog Divider	R/W	Default: 0x09C2 Number of 25MHz ticks (minus 2) that represents the basic watchdog increment. (default value is 100 us = 2498)
0x0410	2	15:0	Watchdog Time PDI	R/W	Default: 0x03E8 Number of basic watchdog increments (default value with Watchdog Divider of 100 us results in 100 ms watchdog). The watchdog is disabled if Watchdog Time PDI is set to 0000h. Watchdog is restarted with every PDI access.
0x0420	2	15:0	Watchdog Time Process Data	R/W	Default: 0x03E8 Number of basic watchdog increments (default value with Watchdog Divider of 100 us results in 100 ms watchdog). There is one watchdog for all SyncManagers. The watchdog is disabled if Watchdog Time PDI is set to 0x0000. The watchdog is restarted with every write access to the SyncManagers with the Watchdog Trigger Enable bit set.
0x0440	2	15:0	Watchdog Status of Process Data	RO	(triggered by SyncManagers) 0: Watchdog Process Data expired 1: Watchdog Process Data is active or disabled
0x0442	1	7:0	Watchdog Counter Process Data	R/WC	Counting is stopped when FFh is reached. Counts if Process Data Watchdog expires. This field is cleared if one of the Watchdog counters (0442h-0443h) is written.
EEPROM Interface					
0x0500	1	EEPROM Configuration			
		1	Force ECAT Access	R/W	0: Do not change 1: Reset
		0	PDI EEPROM Control	R/W	0: No 1: Yes (PDI has EEPROM control) EtherCAT controls the SII EEPROM interface if the PDI EEPROM Control bit of the EEPROM Configuration Register is 0 and the Access to EEPROM bit of the EEPROM PDI Access State Register is 0. Otherwise, PDI controls the EEPROM interface.
0x0501	1	0	EEPROM PDI Access State	RO	0: Do not change 1: Reset
EEPROM Control/Status Register					
0x0502	2	15	Busy	RO	0: EEPROM interface is idle 1: EEPROM interface is busy
		14	Error Write Enable	RO	0: No error



Offset	Size [Bytes]	Bits	Name	R/W	Description
					1: Write Command without Write enable
		13	Error Acknowledge/Command	RO	0: No error 1: Missing EEPROM acknowledge or invalid command
		12	EEPROM Loading Status	RO	0: EEPROM loaded, device information okay 1: EEPROM not loaded, device information not available (EEPROM loading in-progress or finished with a failure)
		11	Checksum Error in ESC Configuration Area	RO	0: Checksum okay 1: Checksum error
		10:8	Command Register	R/W	Write: Initiate command Read: Currently executed command 000: No command/EEPROM idle (clear error bits) 001: Read 010: Write 100: Reload Others: RESERVED / invalid commands (no not issue)
		7	Selected EEPROM Algorithm	RO	1: 2 address bytes (32Kbit- 4Mbit EEPROMs)
		6	Supported Number of EEPROM Bytes	RO	0: 4 Bytes 1: 8 Bytes
		5	EEPROM Emulation	RO	0: Normal operation (I2C interface used) 1: PDI emulates EEPROM (I2C not used)
		0	ECAT Write Enable	R/W	0: Write requests are disabled 1: Write requests are enabled
0x0504	4	31:0	EEPROM Address	R/W	
0x0508	4	31:0	EEPROM Read/Write Data	R/W	Only lower two Bytes can be written. All four Bytes can be read.
<b>MII Management Interface</b>					
0x0510	2	MII Management Control/Status			
		15	Busy	RO	0: MI control state machine is idle 1: MI control state machine is active
		14	Command Error	RO	0: Last command was successful 1: Invalid command or write command without write enable Note: Cleared with a valid command or by writing "00" to Command Register.
		13	Read Error	R/W	0: No read error 1: Read error occurred (PHY or register bi available)
		9:8	Command Register	R/W	Write: Initiate command. Read: Currently executed command Commands: 00: No command / MI Idle (clear error bits) 01: Read 10: Write 11: RESERVED (do not issue)
		7:3	PHY Address Offset	RO	Default: 0x0000
		2	MI Link Detection	RO	0: Not available 1: MI Link Detection Active
		1	Management Interface Control	RO	0: ECAT control only Default 1: MPDI control possible (MII Management ECAT Access State Register and MII Management PDI Access State Register)
		0	Write Enable	R/W	0: Write Disabled 1: Write Enabled
0x0512	1	4:0	Phy Address	R/W	Default: 0x0000
0x0513	1	4:0	Address of PHY Register to be Read/Written	R/W	
0x0516	1	0	Access to MII Management (ECAT)	R/W	0: ECAT enables PDI takeover of MII management control 1: ECAT claims exclusive access to MII management

Offset	Size [Bytes]	Bits	Name	R/W	Description
0x0517	1	1	Force PDI Access State	R/W	0: Do not change Access to MII Management (PDI) bit 1: Reset Access to MII Management (PDI) bit
		0	Access to MII Management (PDI)	RO	0: ECAT has access to MII management 1: PDI has access to MII management
0x0518	1	5	Port x Lost Link Counter	R/WC	0: No Update 1: PHY Configuration was Updated Note: Cleared by writing any value to at least one of the PHY Port Status Registers.
		4	Port x Link Partner Error	RO	0: No Error Detected 1: Link Partner Error
		3	Port x Read Error	R/WC	0: No Read Error Detected 1: Read Error has Occurred Note: Cleared by writing any value to at least one of the PHY Port Status Registers.
		2	Port x Link Status Error	RO	0: No Error 1: Link Error, Link Inhibited
		1	Port x Link Status	RO	(100 Mbit/s, Full-Duplex, Auto-negotiation) 0: No Link 1: Link Detected
		0	Port x Physical Link	RO	(PHY Status Register 1.2) 0: No Physical Link 1: Physical Link Detected
		<b>FMMU</b>			
LAN9252 includes 3 FMMUs. Each FMMU is described in 16 Bytes, starting at 0x0600. FMMU0 base address 0x0600; FMMU1 base address 0x0610; FMMU2 base address 0x0620. The subsequent FMMU registers will be referenced as an offset from these various base addresses. The variable “x” is used in the following descriptions to represent FMMUs 0 through 2.					
+ 0x0	4	31:0	Logical Start Address	R/W	Logical start address within the EtherCAT address space.
+ 0x4	2	15:0	Length	R/W	Offset from the first logical FMMU byte to the last FMMU Byte + 1 (e.g., if two bytes are used, then this parameter shall contain 2).
+ 0x6	1	2:0	Logical Start Bit	R/W	Logical starting bit that shall be mapped (bits are counted from least significant bit (0) to most significant bit (7)).
+ 0x7	1	2:0	Logical Stop Bit	R/W	Last logical bit that shall be mapped (bits are counted from least significant bit (0) to most significant bit (7)).
+ 0x8	2	15:0	Physical Start Address	R/W	Mapped to logical start address
+ 0xA	1	2:0	Physical Start Bit	R/W	Physical starting bit as target of logical start bit mapping (bits are counted from least significant bit (0) to most significant bit (7)).
+ 0xB	1	FMMU Type			
		0	Write Access Mapping	R/W	0: Ignore mapping for write accesses 1: Use mapping for write accesses
		1	Read Access Mapping	R/W	0: Ignore mapping for read accesses 1: Use mapping for read accesses
+0xC	1	0	FMMU Activation	R/W	0: FMMUx Deactivated 1: FMMUx Activated. FMMUx checks logical addressed blocks to be mapped according to the configured mapping.
<b>SyncManager</b>					
LAN9252 includes 4 SyncManagers. Each SyncManager is described in 8 Bytes, starting at 0800h. SyncManager 0 base address 0x0800; SyncManager 1 base address 0x0808; SyncManager 2 base address 0x0810; SyncManager 3 base address 0x0818. The subsequent SyncManager registers will be referenced as an offset from these various base addresses. The variable “x” is used in the following descriptions to represent SyncManagers 0 through 3. Configuration registers can only be written if SyncManager x is disabled via the SyncManager Enable/Disable bit of the SyncManager x Activate Register.					
+ 0x0	2	15:0	Physical Start Address	R/W	Specifies the first byte that will be handled by SyncManager x.
+ 0x2	2	15:0	Length	R/W	Number of bytes assigned to SyncManager x. This field shall be greater than 1, otherwise the

Offset	Size [Bytes]	Bits	Name	R/W	Description
					SyncManager is not activated. If set to 1, only Watchdog Trigger is generated, if configured.
+0x4	1	SyncManager x Control Register			
		6	Watchdog Trigger Enable	R/W	0: Disabled 1: Enabled
		5	Interrupt in PDI Event Request Register	R/W	0: Disabled Default 1: Enabled
		4	Interrupt in ECAT Event Request Register	R/W	0: Disabled Default 1: Enabled
		3:2	Direction	R/W	00: Read: EtherCAT master read access 01: Write: ECAT master write access
		1:0	Operation Mode	R/W	00: Buffered (3 buffer mode) 10: Mailbox (single buffer mode)
+ 0x5	1	SyncManager x Status Register			
		7	Write Buffer in Use	RO	
		6	Read Buffer in Use	RO	
		5:4	Buffer Status	RO	00: 1. buffer 01: 2. buffer 10: 3. buffer 11: No buffer written
		3	Mailbox Status	RO	0: Mailbox Empty 1: Mailbox Full
		1	Interrupt Read	RO	0: Interrupt cleared after first byte of buffer was written 1: Interrupt after buffer was completely and successfully read
		0	Interrupt Write	RO	0: Interrupt cleared after first byte of buffer was read 1: Interrupt after buffer was completely and successfully written
+ 0x6	1	SyncManager x Activate Register			
		7	Latch Event PDI	R/W	0: No 1: Generate latch events if stepper application issues a buffer exchange or if it accesses buffer start address.
		6	Latch Event ECAT	R/W	0: No 1: Generate latch event if EtherCAT master issues a buffer exchange.
		1	Repeat Request	R/W	A toggle of Repeat Request indicates that a mailbox retry is needed (primarily used in conjunction with ECAT Read Mailbox)
		0	SyncManager Enable/Disable	R/W	0: Disable: Access to memory without SyncManager control 1: Enable: SyncManager is active and controls memory area set in configuration.
Distributed Clocks- Receive Times					
0x0900	4	31:0	Receive Time Port 0	R/W	Write:
0x0904	4	31:0	Receive Time Port 1	R/W	A write access to register 090xh with BWR, APWR (any address) or FPWR (configured address) latches the local time of the beginning of the receive frame (start first bit of preamble) at each port. Read: Local time of the beginning of the last receive frame containing a write access to this register. Note: The time stamps cannot be read in the same frame in which this register was written.
Distributed Clocks- Time Loop Control Unit					
0x0910	8	System Time			
		63:0	Read Access	RO	Local copy of the System Time when the frame passed

Offset	Size [Bytes]	Bits	Name	R/W	Description
					the reference clock (i.e., including System Time Delay). Time latched at beginning of the frame (Ethernet SOF delimiter).
		31:0	Write Access	W	Written value will be compared with the local copy of the system time. The result is an input to the time control loop.
0x0918	8	63:0	Receive Time EtherCAT Processing Unit	RO	Local time of the beginning of a frame (start first bit of preamble) received at the ECAT Processing Unit containing a write access to Receive Time Port 0 Register (0900h).
0x0920	8	63:0	System Time Offset	R/W	Difference between local time and System Time. Offset is added to local time.
0x0928	4	31:0	System Time Delay	R/W	Delay between Reference Clock and the ESC
0x092C	4	31	System Time Difference	RO	0: Local copy of System Time greater than or equal to received System Time 1: Local copy of System Time smaller than received System Time
		30:0		RO	Mean difference between local copy of System Time and received System Time values.
...			Speed counter...		
Distributed Clocks- Cyclic Unit Control					
0x0980	1	Cyclic Unit Control Register			
		5	Latch In Unit 1		Default 0: EtherCAT master controlled 1: Stepper application controlled
		4	Latch In Unit 0		
		0	Sync Out Unit Control	R/W	
Distributed Clocks- SYNC Out Unit					
0x0981	1	Activation			
		7	SyncSignal Debug Pulse (Vasili Bit)	R/W	Default 0: Deactivated 1: Immediately generate a single debug ping on SYNC0 and SYNC1 according to bits 2 and 1 of this register.
		6	Near Future Configuration (approx.)	R/W	Default 0: 1/2 DC width future 1: 2.1 s future
		5	Start Time Plausibility Check	R/W	0: Disabled. SyncSignal generation if Start Time is reached. 1: Immediate SyncSignal generation if Start Time is outside Near Future Configuration (approx.).
		4	Extension of Start Time Cyclic Operation	R/W	0: No extension 1: Extend 32-bit written Start Time to 64-bit
		3	Auto-activation	R/W	0: Disabled 1: Auto-activation enabled. Sync Out Unit Activation is set automatically after Start Time is written.
		2	SYNC1 Generation	R/W	0: Deactivated 1: SYNC1 pulse is generated
		1	SYNC0 Generation	R/W	0: Deactivated 1: SYNC0 pulse is generated
		0	Sync Out Unit Activation	R/W	0: Deactivated 1: Activated
0x0982	2	15:0	Pulse Length of Sync Signals	RO	Default 0. In units of 10ns, a value of 0 is used for Acknowledge Mode: SyncSignal will be cleared by reading the SYNC0 Status Register/SYNC1 Status Register.
0x0984	1	Activation Status			
		2	Start Time Cyclic Operation plausibility check result when Sync Out Unit was activated	RO	0: Start Time was within near future 1: Start Time was out of near future
		1	SYNC1 Activation State	RO	0: First SYNC1 pulse is not pending 1: First SYNC1 pulse is pending

Offset	Size [Bytes]	Bits	Name	R/W	Description
		0	SYNC0 Activation State	RO	0: First SYNC0 pulse is not pending 1: First SYNC0 pulse is pending
0x098E	1	0	SYNC0 State for Acknowledge Mode	RO	SYNCx, in Acknowledge Mode, is cleared when read by local application.
0x098F	1	0	SYNC1 State for Acknowledge Mode	RO	
0x0990	8	63:0	Start Time Cyclic Operation	R/W	Write: Start time (System Time) of cyclic operation in ns. Read: System time of next SYNC0 pulse in ns.

Tab. 53 - ESC registers

## 11. CANopen

### 11.1. Standards

FD1 and FD2 drives with suffix A implement following CANopen standards:

- CiA Draft Standard 301 V4.02
- CiA Draft Standard Proposal 402 V2.0

### 11.2. Communication objects

FD supports CAN standard frames with 11-bits identifier field. The default profile ID-allocation scheme consists of a functional part, which determines the object priority and a Node-ID-part, which allows to distinguish between devices of the same functionality. This allows a peer-to-peer communication between a single master device and up to 127 slave devices. It also supports the broadcasting of non-confirmed NMT and SYNC objects. Broadcasting is indicated by a Node-ID of zero.

Bit	10	9	8	7	6	5	4	3	2	1	0
COB-ID	Function Code				Node-ID						

Tab. 54 - COB-ID

CANOPEN\_ADDRESS (Node-ID) can selected via binary combination of DIP switches or, if DIP switches are all off, address zero, the address is taken from flash memory parameter. Dynamic modifications of Node-ID are not supported.

Object	Function code	Resulting COB-ID
NMT	0000	0
SYNC	0001	128 (0x80)
EMERGENCY	0001	129 (0x81) – 255 (0xFF)
PDO1 (tx)	0011	385 (0x181) – 511 (0x1FF)
PDO1 (rx)	0100	513 (0x201) – 639 (0x27F)
PDO2 (tx)	0101	641 (0x281) – 767 (0x2FF)
PDO2 (rx)	0110	769 (0x301) – 895 (0x37F)
PDO3 (tx)	0111	897 (0x381) – 1023 (0x3FF)
PDO3 (rx)	1000	1025 (0x401) – 1151 (0x47F)
PDO4 (tx)	1001	1153 (0x481) – 1279 (0x4FF)
PDO4 (rx)	1010	1281 (0x501) – 1407 (0x57F)
SDO (tx)	1011	1409 (0x581) – 1535 (0x5FF)
SDO (rx)	1100	1537 (0x601) – 1663 (0x67F)

Tab. 55 - Communication objects

### 11.2.1. Network Management Objects (NMT)

The Network Management (NMT) is node oriented and follows a master-slave structure. NMT objects are used for executing NMT services, which refers to CAN network, i.e. the communication aspects. Through NMT services, nodes communication is initialized, started, monitored, reset or stopped. All nodes are regarded as NMT slaves. A NMT slave is uniquely identified in the network by its Node-ID, a value in the range of [1 – 127]. NMT requires that one device in the network fulfils the function of the NMT Master.

FD can have four NMT status:

- 0: Initializing,
- 4: Stopped,
- 5: Operational,
- 127: Pre-operational.

FD boots up in initializing status, when the initialization is complete it enters in pre-operational status and transmits the boot-up heartbeat.

COB-ID	RTR	DLC	Data 0
0x700 + NodeID	0	1	0

Tab. 56 - NMT boot-up

The drive status can be read using NMT node guard or using heartbeat mechanisms:

The guarding is achieved through transmitting guarding requests (Node guarding protocol) by the NMT Master.

COB-ID	RTR	DLC
0x700 + NodeID	1	0

Tab. 57 - NMT guarding requests

The FD answers its status as:

COB-ID	RTR	DLC	Data 0
0x700 + NodeID	0	1	Toggle + status

Tab. 58 - NMT guarding answer

The only byte of data in slave answer is as follows:

- Bit 7 toggles (0 after communication reset),
- Bits 6 – 0 contains the NMT status

Life guarding is a service to be used for guarding the NMT master from the NMT slave. The slave uses the objects 0x100C, guard time multiplied by 0x100D, life time factor to calculate the node life time. If the NMT Slave is not guarded within its life time, the NMT Slave stops the motor, resets the CAN objects, NMT status becomes PRE-OPERATIONAL, drive status becomes SWITCH ON DISABLED.

If guard time and life time factor are 0 (default values), the NMT Slave does not guard the NMT Master. Guarding starts for the slave when the first remote-transmit-request for its guarding identifier is received with guard time and life time factor different than zero.

The heartbeat protocol defines control of NMT status without need of remote frames. FD transmits a heartbeat message cyclically. One or more heartbeat consumers receive the indication. The relationship between producer and consumer is configurable via object 0x1017, producer heartbeat time.

If producer heartbeat time is zero, heartbeat is disabled. Otherwise, it defines the period between each heartbeat production in milliseconds. The heartbeat message from the slave is as follows:

COB-ID	RTR	DLC	Data 0
0x700 + NodeID	0	1	Status

Tab. 59 - Heartbeat

Note:

It is not allowed for one device to use both error control mechanisms Guarding Protocol and Heartbeat Protocol at the same time. If the heartbeat producer time is different than 0 the heartbeat protocol is used only.

To modify the slave NMT status following services are implemented and always active:

- Start Remote Node: this service sets the state to operational.

COB-ID	RTR	DLC	Data 0	Data 1
0x0	0	2	0x1	NodeID

Tab. 60 - Start remote node

This service is unconfirmed (no answer from the slave).

- Stop Remote Node: this service sets the state to stop.

COB-ID	RTR	DLC	Data 0	Data 1
0x0	0	2	0x2	NodeID

Tab. 61 - Stop remote node

This service is unconfirmed (no answer from the slave).

- Enter Pre-Operational: this service sets the state to pre-operational.

COB-ID	RTR	DLC	Data 0	Data 1
0x0	0	2	0x80	NodeID

Tab. 62 - Enter pre-operational

This service is unconfirmed (no answer from the slave).

- Reset Node: this service sets the state from any state to the reset application sub-state. After completion of the service, the state of the selected remote nodes will be pre-operational.

COB-ID	RTR	DLC	Data 0	Data 1
0x0	0	2	0x81	NodeID

Tab. 63 - Reset node

- Reset Communication: this service sets the state from any state to the reset communication sub-state. After completion of the service, the state of the selected remote nodes will be PRE\_OPERATIONAL.

COB-ID	RTR	DLC	Data 0	Data 1
0x0	0	2	0x82	NodeID

Tab. 64 - Reset communication

Note:

Reset and Reset Communication are theoretically unconfirmed, but a boot-up message is generated, because of the transition from initializing to pre-operational.



Objects	Initialization	Pre-operational	Operational	Stop
PDO			✓	
SDO		✓	✓	
SYNC		✓	✓	
EMCY		✓	✓	
NMT	✓	✓	✓	✓

Tab. 65 - Objects active per status

SDOs are active only in operational and pre-operational status.

PDOs are active only in operational status and they can be configured only in pre-operational.

### 11.2.2. Service Data Objects (SDO)

With Service Data Objects (SDOs) the access to all the entries of FD object dictionary is provided.

As these entries may contain data of various size and data type, SDOs can be used to transfer multiple data sets (each containing an arbitrary large block of data) from a client to a server and vice versa.

The client can control via a multiplexor (index and sub-index of the Object Dictionary) which data set is to be transferred. The contents of the data set are defined within the object dictionary.

Basically, a SDO is transferred as a sequence of segments. Prior to transferring the segments there is an initialization phase where client and server prepare themselves for transferring the segments. For SDOs, it is also possible to transfer a data set of up to four bytes during the initialization phase. This mechanism is called an expedited transfer.

SDO Block Upload is not supported.

For all transfer types it is the client that takes the initiative for a transfer. The owner of the accessed object dictionary is the server of the SDO. Either the client or the server can take the initiative to abort the transfer of a SDO.

By means of a SDO a peer-to-peer communication channel between two devices is established.

### 11.2.3. Download SDO

The following services can be used:

- Initiate SDO Download
- Download SDO Segment

Through these services the client of a SDO downloads data to the server, i.e. the FD drive (owner of the object dictionary). The data, the multiplexor (index and sub-index) of the data set to be downloaded and its size (only optionally for segmented transfer) are indicated to the FD.

The service is confirmed. In case of a failure, the reason is confirmed with an Abort SDO message.

The SDO download consists of at least the Initiate SDO Download service and optional of Download SDO Segment services (for data length bigger than 4 bytes).

SDOs are downloaded as a sequence of zero or more Download SDO Segment services preceded by an Initiate SDO Download service. The sequence is terminated by:

- Initiate SDO Download request/indication with the e-bit set to 1 followed by an Initiate SDO Download response/confirm, indicating the successful completion of an expedited download sequence.
- Download SDO Segment response/confirm with the c-bit set to 1, indicating the successful completion of a normal download sequence.
- Abort SDO Transfer request/indication, indicating the unsuccessful completion of the download sequence.
- Initiate Domain Download request/indication, indicating the unsuccessful completion of the download sequence and the start of a new download sequence.

The initiate SDO Download request from the client is as follows:

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x600 + NodeID	0	> 4	[7 – 5] 1 [4] 0 [3 – 2] n [1] e [0] s	Index		Sub- Index	d			

Tab. 66 - Initiate SDO Download request

DLC shall be at least 5.

Data 0:

- bits [7 – 5] indicate the command specifier, in this case it is 1.
- bit 1 is the transfer type:
  - o 0: segmented,
  - o 1: expedited.
- bit 0 is the size indicator. If it is equal to 1, the data set size is indicated. In case of expedited transfer it is indicated in n, otherwise in a segmented transfer it is indicated in Data [4 – 7].
- bits [3 – 2] only valid if e = 1 and s = 1, otherwise 0. If valid it indicates the number of bytes in d that do not contain data. Bytes [8-n, 7] do not contain data.

Data [1 – 3] Index and sub-index address the data into the object dictionary.

Data [4 – 7] depends upon the type of transfer.

- Expedited: contains the data to be downloaded. 4 – n Bytes if s = 1. Unspecified number if s = 0,
- Segmented: contains the number of bytes to be downloaded (Byte4 LSB, Byte 7 MSB).

In case of successful download FD will answer:

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x580 + NodeID	0	8	0x60	Index		Sub- Index	0			

Tab. 67 – Initiate SDO download answer

The download SDO segment request from client is as follows:

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x600 + NodeID	0	> 1	[7 – 5] 0 [4] t [3 – 1] n [0] c	Segmented data						

Tab. 68 - Download SDO segment request

DLC shall be at least 2.

Data 0:

- bits [7 – 5] indicate the command specifier, in this case it is 0.
- bit 4 is the toggle bit. This bit alternates for each subsequent segment that is downloaded. The first segment have the toggle bit 0. The toggle bit will be equal for the request and the response message.
- bits [3 – 1] n: indicates the number of bytes in segmented data that do not contain data. Bytes [8-n, 7] do not contain segment data. n = 0 if no segment size is indicated.
- bit 0 c: indicates whether there are still more segments to be downloaded:
  - o 0: more segments to be downloaded,
  - o 1: no more segments to be downloaded.

FD answers:

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x580 + NodeID	0	8	[7 – 5] 1 [4] t [3 – 0] 0	0						

Tab. 69 - Download SDO segment answer

Note:

The segments are stored in a buffer before being written into the object dictionary. The buffer is 64 Bytes long. This is the limit of the maximum segmented transfer.

### 11.2.4. Upload SDO

The following services can be used:

- Initiate SDO Upload
- Upload SDO Segment

Through this service the client of a SDO uploads data from the server, i.e. reads from FD. The multiplexor (index and sub-index) of the data set that has to be uploaded is indicated to the server. The SDO upload consists of at least the Initiate SDO Upload service and optional of Upload SDO Segment services (data length > 4 bytes).

The service is confirmed. In case of a failure, an Abort SDO transfer request is executed. In case of success, the server has accepted the segment data and is ready to accept the next segment.

A successful Initiate SDO Download service with segmented transfer type must have been executed.

SDO are uploaded as a sequence of zero or more Upload SDO Segment services preceded by an Initiate SDO Upload service. The sequence is terminated by:

- Initiate SDO Upload response/confirm with the e-bit set to 1, indicating the successful completion of an expedited upload sequence.
- Upload SDO Segment response/confirm with the c-bit set to 1, indicating the successful completion of a normal upload sequence.
- Abort SDO Transfer request/indication, indicating the unsuccessful completion of the upload sequence.
- new Initiate SDO Upload request/indication, indicating the unsuccessful completion of the upload sequence and the start of a new sequence.

The initiate SDO Upload request from the client is as follows:

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x600 + NodeID	0	8	0x40	Index		Sub- Index	0			

Tab. 70 - Initiate SDO upload request

FD will answer:

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x580 + NodeID	0	> 4	[7 – 5] 2 [4] 0 [3 – 2] n [1] e [0] s	Index		Sub- Index	d			

Tab. 71 - Initiate SDO upload answer

DLC shall be at least 5.

Data 0:

- bits [7 – 5] indicate the command specifier, in this case it is 2.
- bit 1 is the transfer type:
  - o 0: segmented,
  - o 1: expedited.
- bit 0 is the size indicator. If it is equal to 1, the data set size is indicated. In case of expedited transfer it is indicated in n, otherwise in a segmented transfer it is indicated in Data [4 – 7].
- bits [3 – 2] only valid if e = 1 and s = 1, otherwise 0. If valid it indicates the number of bytes in d that do not contain data. Bytes [8-n, 7] do not contain data.

Data [1 – 3] Index and sub-index address the data into the object dictionary.

Data [4 – 7] depends upon the type of transfer.

- Expedited: contains the data uploaded. 4 – n Bytes if s = 1. Unspecified number if s = 0,
- Segmented: contains the number of bytes to be uploaded (Byte4 LSB, Byte 7 MSB).

If the FD answer with a segmented transfer, the client shall upload the segment using:

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x600 + NodeID	0	8	[7 – 5] 3 [4] t [3 – 0] 0	Index		Sub- Index	0			

Tab. 72 - Upload SDO segment request

Data [0]:

- t: toggle bit. This bit alternate for each subsequent segment that is uploaded. The first segment have the toggle-bit set to 0.

FD answers:

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x580 + NodeID	0	8	[7 – 5] 0 [4] t [3 – 1] n [0] c	Segmented data						

Tab. 73 - Upload SDO segment answer

Data [0]:

- t: toggle bit. The response message will have the same bit of request.
- n: indicates the number of bytes in seg-data that do not contain segment data. Bytes [8-n, 7] do not contain segment data. n = 0 if no segment size is indicated.
- c: indicates whether there are still more segments to be uploaded:
  - o 0: more segments,
  - o 1: no more segments.

Note:

FD always returns the toggle bit, which is received. It doesn't rise any alarm if it doesn't toggle.

### 11.2.5. Abort SDO

The abort SDO can be sent from the Client to stop a segmented transfer or from the FD in case of error.

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x600 or 0x580 + NodeID	0	8	[7 – 5] 4 [4 – 0] 0	Index		Sub- index	Abort code			

Tab. 74 – Abort SDO

Abort code	Description
0x05040001	Client/server command specifier not valid or unknown.
0x05040002	Invalid block size.
0x05040003	Invalid sequence number.
0x05040004	CRC error.
0x05040005	Out of memory
0x06010002	Attempt to write a read only object.
0x06020000	Object does not exist in the object dictionary.
0x06040041	Object cannot be mapped to the PDO.
0x06040042	The number and length of the objects to be mapped would exceed PDO length.
0x06040043	General parameter incompatibility reason.
0x06040047	General internal incompatibility.
0x06070010	Data type does not match, length of service parameter does not match
0x06090011	Sub-index does not exist.
0x06090030	Value range of parameter exceeded (only for write access).
0x06090031	Value of parameter written too high.
0x06090032	Value of parameter written too low.
0x08000000	General error

Tab. 75 – Abort codes

### 11.2.6. SDO block download

SDO can be transferred as a sequence of blocks where each block is a sequence of up to 127 segments containing a sequence number and the data.

The only objects that can be transferred in this way are the sub-indexes of 0x1F50, Download program data, which are firmware and parameters.

Prior to transferring the blocks there is an initialization phase where client and server prepare themselves for transferring the blocks and negotiating the number of segments in one block. After transferring the blocks there is a finalization phase where client and server can verify the correctness of the previous data transfer by comparing checksums derived from the data set.

After block download the server indicates the client the last successfully received segment of this block transfer by acknowledging this segment sequence number. Doing this the server implicitly acknowledges all segments preceding this segment. The client has to start the following block transfer with the retransmission of all not acknowledged data. Additionally the server has to indicate the number of segments per block for the next block transfer.

The SDO Download Block sequence is terminated by:

- a downloaded segment within a block with the c-bit set to 1, indicating the completion of the block download sequence.
- an 'Abort SDO Transfer' request/indication, indicating the unsuccessful completion of the download sequence.

The whole 'SDO Block Download' service is terminated with the End SDO Block Download service. If client has indicated the ability to generate a CRC during the Initiate SDO Block Download service, the server generates the CRC on the received data. If this CRC differs from the CRC generated by the client, the server indicates this with an 'Abort SDO Transfer' indication.

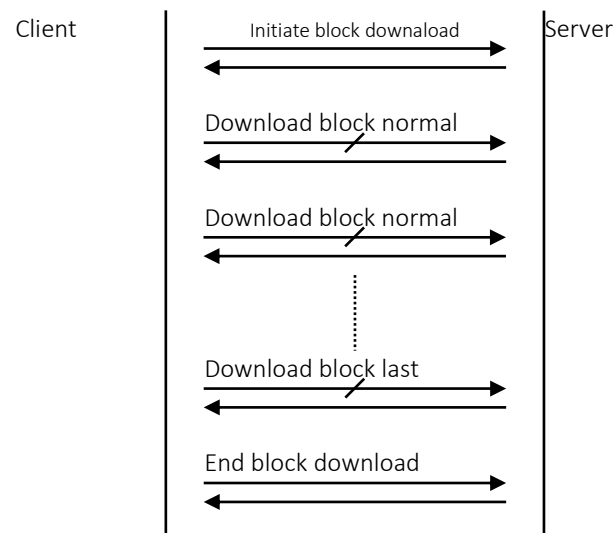


Fig. 18 - SDO block download

When the transmission of firmware and/or parameters has been successfully carried out, the client shall perform a NMT Reset command to jump back to main application.

## Initiate SDO Block Download

Through this service the client requests the server to prepare for downloading data. FD drive will disable motor current and jump to program mode, i.e. the bootloader.

The client as well as the server indicate their ability and/or demand to verify the complete transfer with a checksum that will happen during End SDO Block Download.

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x600 + NodeID	0	> 4	[7 – 5] 6 [4 – 3] 0 [2] cc [1] s [0] 0	Index		Sub- Index	size			

Tab. 76 - SDO init block download request

Data 0:

- bits [7 – 5] indicate the client command specifier, in this case it is 6.
- bit 2 is the client CRC support:
  - o 0: client does not support generating CRC on data,
  - o 1: client supports generating CRC on data.
- bit 1 is the size indicator. If it is equal to 1, the data set size is indicated.

Data [1 – 3] Index and sub-index address the data into the object dictionary. Index can be only 0x1F50, sub index can be:

- sub-index 1: program firmware,
- sub-index 2: program parameters.

Data [4 – 7] download size in Bytes if s = 1.

FD answers:

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x580 + NodeID	0	8	[7 – 5] 5 [4 – 3] 0 [2] sc [1 – 0] 0	Index		Sub- Index	blksize	0		

Tab. 77 - SDO init block download response

Data 0:

- bits [7 – 5] indicate the server command specifier, in this case 5
- bit 2 is the server CRC support = 1, because supported by FD drives.

Data 4: blksize, that is the number of segments per block with  $0 < \text{blksize} < 128$ .

## Download SDO Block

By this service the client supplies the data of the block to the server. The block data is transmitted to the server by a sequence of segments. Each segment consists of the data and a sequence number starting with 1 which is increased for each segment by 1 up to blksize. The parameter blksize is negotiated between server and client in the 'Initiate Block Download' protocol and can be changed by the server with each confirmation for a block transfer. The continue parameter indicates the server whether to stay in the 'Download Block' phase or to change in the 'End Download Block' phase.

The service is confirmed. In case of a success the ackseq parameter indicates the sequence number of the last segment the server has received successfully. If this number does not correspond with the sequence number of the last segment sent by the client during this block transfer the client has to retransmit all segments discarded by the server with the next block transfer. In case of a fatal failure, an Abort SDO Transfer request is executed. In case of success, the server has accepted all acknowledged segment data and is ready to accept the next block. There can be at most one Download SDO Block service outstanding for a SDO transfer.

A successful 'Initiate SDO Block Download' service must have been executed prior to this service.

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x600 + NodeID	0	8	[7] c [6 – 0] s	Segment data						

Tab. 78 - SDO block download segment

Data 0:

- bit 7: c, indicates whether there are still more segments to be downloaded
  - o 0: more segments to be downloaded
  - o 1: no more segments to be downloaded, enter End SDO block download phase
- Bits [6 – 0]: s, sequence number of segment  $0 < s < 128$ .

FD answers:

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x580 + NodeID	0	8	[7–5] 5 [1–0] 2	ackseq	blksize	0				

Tab. 79 - SDO block download segment response

Data 0:

- bits [7-5] indicate the server command specifier, in this case 5
- bits [0-1] server subcommand, in this case 2

Data 1: ackseq: sequence number of last segment that was received successfully during the last block download. If ackseq is set to 0 the server indicates the client that the segment with the sequence number 1 was not received correctly and all segments have to be retransmitted by the client.

Data 2: blksize: Number of segments per block that has to be used by client for the following block download with  $0 < \text{blksize} < 128$ .

### End SDO Block Download

Through this service the SDO Block Download is concluded. The number of bytes not containing valid data in the last transmitted segments is indicated to the server.

If the server as well as the client have indicated their ability and demand to check the complete transfer with a checksum in 'Initiate SDO Block Download' this checksum is indicated to the server by the client. The server also has to generate a checksum which has to be compared with the one generated by the client.

The service is confirmed. The Remote Result parameter will indicate the success of the request (matching checksums between client and server if negotiated) and concludes the download of the data set. In case of a failure, an Abort SDO Transfer request must be executed.

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x600 + NodeID	0	8	[7-5] 6 [4-2] n [0] 1	CRC		0				

Tab. 80 - End SDO block download

Byte 0:

- bits [7-5] is the client command specifier: 6 = block download
- bit 0 is the client subcommand: 1 = end block download request

Byte 1 and 2 are the CRC, Cyclic Redundancy Checksum. The check polynomial has the formula  $x^{16} + x^{12} + x^5 + 1$ . The calculation has to be made with an initial value of 0.

To confirm FD answers:

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x580 + NodeID	0	8	[7-5] 5 [0-1] 1	0						

Tab. 81 - End SDO block download response



### 11.2.7. Synchronization object (SYNC)

The Synchronization Object is broadcasted periodically by the SYNC producer, i.e. the master, and received from all the FD in the network. This SYNC provides the basic network clock. It is unconfirmed, with no data. There can be a time jitter in transmission by the SYNC producer corresponding approximately to the latency due to some other message being transmitted just before the SYNC. In order to guarantee timely access to the CAN bus the SYNC is given a very high priority identifier, expressed in the object 0x1005, COB-ID SYNC.

Default SYNC COB-ID is 0x80.

### 11.2.8. Emergency object (EMCY)

Emergency objects are triggered by the occurrence of an alarm and they are transmitted from FD. An emergency object is transmitted only once per error event. As long as no new error occurs on a device no further emergency objects are transmitted.

COB-ID	RTR	DLC	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x80 + NodeID	0	8	Error code		Error register	0				

Tab. 82 – Emergency objects

Error code	Description
0x0000	No error
0x1000	Generic error
0x2300	Over current, on device output side
0x3210	Over voltage, on V <sub>POW</sub>
0x3220	Under voltage, on V <sub>POW</sub>
0x4310	Over temperature
0x6320	Data error
0x8100	Communication error (it does not generate EMCY)
0x8130	Life guard error
0x8611	Step loss error, following

Tab. 83 - Error codes

After initialization the FD enters the error free status. No error message is sent.

If FD goes in alarm an emergency object with the appropriate error code and error register is transmitted. The error code is also logged in the object 0x1003, pre-defined error field.

When the alarm is reset, an emergency message containing error code 0000 (Error reset) is transmitted.

### 11.2.9. Process Data Objects (PDO)

The real-time data transfer is performed by means of PDO. They are unconfirmed services used to write or read up to 8 Bytes without protocol overhead.

The PDOs correspond to entries in the object dictionary. Data type and mapping of objects into a PDO is determined by a corresponding PDO mapping structure within the dictionary. The mapping of objects into PDOs need to be transmitted during the PRE OPERATIONAL state by applying the SDO services to the PDO mapping objects (only Bytes or multiples can be mapped).

There are two kinds of use for PDOs. The first is data transmission and the second data reception. It is distinguished in Transmit-PDOs (TPDOs) and Receive-PDOs (RPDOs).

FD implements 4 RPDO and 4 TPDO.

The PDO communication parameter describes the communication capabilities of the PDO. The PDO mapping parameter contains information about the contents of the PDOs. The indices of the corresponding Object Dictionary entries are computed by the following formulas:

- RPDO communication parameter index = 1400h + RPDO-number-1
- TPDO communication parameter index = 1800h + TPDO-number-1
- RPDO mapping parameter index = 1600h + RPDO-number-1
- TPDO mapping parameter index = 1A00h + TPDO-number-1

The entries mentioned above are described below in Object Dictionary.

Remotely requested PDO are not implemented, as recommended by:

## CiA 802 – Application note – CAN remote frames: Avoiding of usage

RPDOs default mapping is:

- RPDO 1 = *control word* 0x60400010,
- RPDO 2 = *control word* 0x60400010, *modes of operation* 0x60600008,
- RPDO 3 = *control word* 0x60400010, *target position* 0x607A0020,
- RPDO 4 = *control word* 0x60400010, *target velocity* 0x60FF0020.

TPDOs default mapping is:

- TPDO 1 = *status word* 0x60410010,
- TPDO 2 = *status word* 0x60410010, *modes of operation* 0x60600008,
- TPDO 3 = *status word* 0x60410010, *position actual value* 0x60630020,
- TPDO 4 = *status word* 0x60410010, *velocity actual value* 0x606C0020.

The structure of the mapping parameter, 8 Bytes is as follows:

- Byte 0 = object length (number of bits),
- Byte 1 = sub index,
- Bytes 2 and 3 = index.

For changing the PDO mapping first the PDO has to be deleted, the sub-index 0 must be set to 0 (mapping is deactivated), then the objects can be remapped. After all objects are mapped sub-index 0 has to be set to the valid number of mapped objects.

Finally, the PDO has to be created by setting the bit 32 in the PDO COB\_ID parameter.

As only standard CAN frames are supported (Remotely requested PDO are not implemented), an attempt to set PDO COB\_ID parameter bit 29 to 1 or bit 30 to 0 is responded with an abort message (abort code: 0x06090030).

The transmission type parameter of a PDO specifies the transmission mode as well as the triggering mode.

Synchronous TPDO:

- A transmission type of 0 means that the message shall be transmitted after occurrence of the SYNC but acyclic (not periodically), only if an event occurred before the SYNC.
- A transmission type of n means that the message is transmitted with every n-th SYNC object. ( $0 < n < 242$ ).

Asynchronous TPDO:

- A transmission type of 254 or 255 means that the message is transmitted when the event occurs, without any relation to the SYNC.

Synchronous RPDO:

- A transmission type of n means that the message received after the occurrence of a SYNC is passed to the application with the occurrence of the following SYNC, independent of the transmission rate specified by the transmission type. ( $0 \leq n \leq 240$ ).

Asynchronous RPDO:

- A transmission type of 254 or 255 means that the message is passed directly to the application.

Event based transmission means that the content of the PDO has a variation, i.e. only if there is a change in the content of the TPDO FD transmit it.

## 12.DSP402 – MOTION CONTROL

CANopen and EtherCAT make use of CiA Draft Standard Proposal 402.

The device profile DSP402 defines several modes of operation. FD drives implement profile position mode, homing mode, interpolated position mode, profile velocity mode, cyclic synchronous position mode, cyclic synchronous velocity mode.

The Power Drive System (PDS) Finite State Automation (FSA) defines the drive status and the possible control sequence of the drive. A single state represents a special internal or external behaviour. The state of the PDS also determines which commands are accepted. For example, it is only possible to start a point-to-point move when the drive is in the *operation enabled* state.

The Finite State Automation, illustrated in Fig. 18, describes the device status and the possible control sequences of the drive. States can be changed using the control word and/or internal events. The current state can be read using the status word.

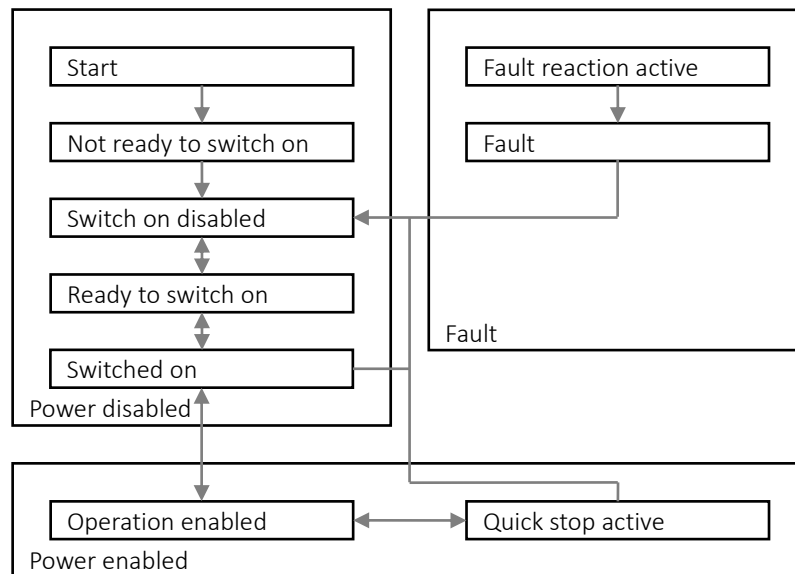


Fig. 19 - FSA finite state automation

- Switch on disabled:
  - *shut down* command sets the state to *ready to switch on*. Motor current and frequency will be disabled.
- Ready to switch on:
  - *quick stop* or *disable voltage* commands set the state to *switch on disabled*. Motor current and frequency will be disabled.
  - *switch on* command sets the state to *switched on*. Motor current and frequency will be disabled.
- Switched on:
  - *shut down* command sets the state in *ready to switch on*. Motor current and frequency will be disabled.
  - *quick stop* or *disable voltage* commands set the state to *switch on disabled*. Motor current and frequency will be disabled.
  - *enable operation* command sets the state in *operation enabled*. Motor current and frequency will be enabled.
- Operation enabled:
  - *shut down* command sets the state in *ready to switch on*. Motor current and frequency will be disabled.
  - *disable voltage* command sets the state to *switch on disabled*. Motor current and frequency will be disabled.
  - *disable operation* command sets the state to in *switched on*. Motor current and frequency will be disabled.
  - *quick stop* command sets the state to *quick stop active*. Motor current and frequency will be enabled.
- Quickstop active:
  - *enable operation* command sets the state to *operation enabled*. Motor current and frequency will be enabled
  - *disable voltage* command sets the state to *switch on disabled*. Motor current and frequency will be disabled.
- Fault:
  - Fault reset command sets the state to switch on disabled. Motor current will be enabled and frequency disabled.

Commands are activated through writes in control word bits as per Tab. 84 – Control word commands.

Commands	Control word bits				
	7	3	2	1	0
	Fault reset	Enable	Quick stop	Enable	Switch on

		operation		voltage	
Shut down	0	X	1	1	0
Switch on	0	X	1	1	1
Disable voltage	0	X	X	0	X
Quick stop	0	X	0	1	X
Disable operation	0	0	1	1	1
Enable operation	0	1	1	1	1
Fault reset	Positive edge	X	X	X	X

Tab. 84 – Control word commands

Note:

Bits marked X are irrelevant.

Operation modes	Control word bits						
	15	12	11	8	6	5	4
Profile position mode	Boost	Sub-mode	Sub-mode	Halt	Sub-mode	Change set immediately	New set point Start move
Profile velocity mode	Boost	-	-	Halt	-	-	-
Homing mode	Boost	-	-	Halt	-	-	Homing start
Position address mode	Boost	-	-	Halt	-	-	-
Interpolated position mode	Boost	-	-	-	-	-	Activate IP mode
Cyclic synchronous position mode	Boost	-	-	-	-	-	-
Cyclic synchronous velocity mode	Boost	-	-	-	-	-	-

Tab. 85 - Control word commands

Boost: When control word bit 15 is active, motor current is boosted at I\_MAX, regardless of measured torque.

Deactivated, motor current control is automatic.

Sub-mode: to start specific profile position movements.

Halt: to stop the movement or to write a sequence of movements and start it upon its removal.

The status word bits return the status of the drive.

Bit number	Status
0	Ready to switch on
1	Switched on
2	Operation enabled
3	Fault
4	Voltage enabled
5	Quick stop
6	Switch on disabled
7	Warning (encoder status)
8	Torque limit
9	Remote (always at 1)
10	Target reached
11	SW Limit switches active
12	Profile position mode: set point acknowledge, Profile velocity mode: speed, Homing mode: homing attained, IP mode: interpolated position mode active, CSP, CSV: drive follows
13	Step loss alarm in profile position, velocity and position address Homing error in homing mode
14	Position uploaded

Tab. 86 - Status word bits

Values (binary)	States
xxxx xxxx x0xx 0000	Not ready to switch on
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 x00x 1111	Fault reaction
xxxx xxxx x0xx 1000	Fault

Tab. 87 - Status

In all the modes the *target reached* bit is set when the deceleration due to Halt command has reached zero speed. In profile position it is also set when the absolute or relative positioning movement have finished to ordered position.

### 12.1.1. Modes of operation

The drive behaviour depends on the activated mode of operation. Since it is not possible to operate the modes in parallel, the user is able to activate the required function by selecting a mode of operation. The master writes to the 0x6060, *modes of operation* object of the drive in order to select the operation mode. The drive provides the 0x6061, *modes of operation display* object to indicate the actual activated operation mode. Controlword, statusword, and set-points are used mode-specific. This implies the responsibility of the control device to avoid inconsistencies and erroneous behaviour.

Values	Mode
0x00	Undefined
0x01	Profile position mode
0x03	Profile velocity mode
0x06	Homing mode
0x07	Interpolated position mode
0x08	Cyclic synchronous position mode
0x09	Cyclic synchronous velocity mode

Tab. 88 - Operating modes

After reset or power on *modes of operation* is set equal to 0x00.

### 12.1.2. Profile position (1)

In this mode the drive internally generates a position trajectory based on input parameters.

From *switched on* status, the master shall set modes of operation to 1, and then write control word 0x0F to enable operation. To start the movement control word bit 4 and status word bit 12 shall commute as per below time graph. The positive edge of control word samples the movement data according to the sub-mode selected:

- *profile velocity* (0x6081),
- *max profile velocity* (0x607F),
- *profile acceleration* (0x6083),
- *profile deceleration* (0x6084),
- *control word* (0x6040) sub-mode bits,
- *target position* (0x607A),
- *delta stop steps* (0x2003).

FD confirms the sampling rising the status word bit 12, set point acknowledge.

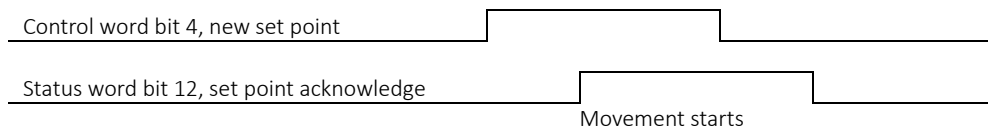


Fig. 20- Profile position mode

After control word bit 4 is lowered, FD signals the readiness of sampling new setpoints with a falling edge of status word bit 12. Always check status word bit 12 level before transmitting new setpoints.

With bit 5 change set immediately at zero, it is possible to create a FIFO buffer of movements giving a new set point while the current movement is running or while the Halt bit is active. As soon as the current movement is finished or the Halt is released the movement defined by the new setpoint starts.

All the movements will be executed in stream, one after the other, till the FIFO is not empty.

FIFO is circular, while the motor is moving it is possible to insert new movements.

Status word bit 12 level at zero shows the status of the FIFO. If the level remains high after bit 4 of control word has been lowered, it means that FIFO is full. The buffer is 31 movements deep, if it is exceeded the set point acknowledge bit will not be released to zero until the current cycle is not completed.

Setting the Halt bit during a movement, the motor stops using profile deceleration and resets the FIFO (same effect as acting on quick stop control word bits). It is recommended not to use the command disable operation while the motor is running, as this command instantly disable the frequency, without deceleration ramp.

The type of trajectory can be absolute positioning, relative positioning, delta stop or time delay cycles. The selection of the type is made by the three bits 12, 11 and 6 *sub-mode of control word*:

Control word bits					Movement
12	11	6	5	4	
0	0	0			Absolute positioning
0	0	1			Relative positioning
0	1	0			Delta stop
1	0	0			Time delay
			1		Change set immediately
				1	New setpoint

Tab. 89 - Sub-modes and commands

The setting of set-points is controlled by the timing of the new set-point bit and the change set immediately bit in the control word as well as the set-point acknowledge bit in the status word.

If the change set immediately bit of the control word is set to 1, the drive will execute the single setpoint. If the change set immediately bit of the control word is set to 0, a set of set-points to be executed in stream can be sent to the drive.

After a set-point is applied to the drive with halt bit at 1, the host starts the motor by a lowering the halt bit in the control word.

When change set immediately bit is at 1, it is possible to continuously transmit to the drive new setpoints, monitoring status word bit 12 to assess that there is enough space into drive's FIFO.

Recommended PDO mapping:

PDO#	B0	B1	B2	B3	B4	B5	B6	B7	Mapping
RPDO1	Control word								1600:0 = 0 1600:1 = 60400010 1600:0 = 1
RPDO2	Profile velocity				Target position				1601:0 = 0 1601:1 = 60810020 1601:2 = 607A0020 1601:0 = 2
RPDO3	Profile acceleration				Profile deceleration				1602:0 = 0 1602:1 = 60830020 1602:2 = 60840020 1602:0 = 2
TPDO1	Status word								1A00:0 = 0 1A00:1 = 60410010 1A00:0 = 1
TPDO2	Velocity actual value				Position actual value				1A01:0 = 0 1A01:1 = 606C0020 1A01:2 = 60640020 1A01:0 = 2

Tab. 90 - PPM recommended PDO mapping

SyncManagers PDO assignment:

SyncManagers	PDOs				Assignment
SM2	RPDO1	RPDO2	RPDO3		1C12:0 = 0 1C12:1 = 1600 1C12:2 = 1601 1C12:3 = 1602 1C12:0 = 3
SM3	TPDO1	TPDO2			1C13:0 = 0 1C13:1 = 1A00 1C13:2 = 1A01 1C13:0 = 2

Tab. 91 - EtherCAT PPM recommended SM assignment

### 12.1.3. Profile velocity (3)

In this mode the drive generates a velocity trajectory. When entering to profile velocity mode the following movement data are sampled:

- profile acceleration (0x6083),
- profile deceleration (0x6084),
- target velocity (0x60FF).

From *switched on* status, the master shall set modes of operation to 3, and then write control word 0x0F to enable operation. As soon as the operation is enabled, the drive will accelerate the motor to target velocity.

Writing on *target velocity* object it is possible to vary the motor speed and direction (this command performs also a new sampling of *profile acceleration* and *profile deceleration* objects). When the sign of *target velocity* is inverted, the motor will decelerate to zero speed and it will accelerate to the opposite direction. When no *invert direction* is configured, positive values correspond to clockwise movements, increasing the position counter value, while negative values correspond to counter-clockwise movements, decreasing the position counter value.

The movement stops setting the control word halt bit or writing target velocity 0.

Once the speed set-point is reached, status word bit 12 *speed* is set.

Velocity actual value is obtained through differentiation from the position encoder, low pass filtered.

Recommended PDO mapping:

PDO#	B0	B1	B2	B3	B4	B5	B6	B7	Mapping
RPDO1	Control word		Target velocity						1600:0 = 0 1600:1 = 60400010 1600:2 = 60FF0020 1600:0 = 2
RPDO2	Profile acceleration				Profile deceleration				1601:0 = 0 1601:1 = 60830020 1601:2 = 60840020 1601:0 = 2
TPDO1	Status word								1A00:0 = 0 1A00:1 = 60410010 1A00:0 = 1
TPDO2	Velocity actual value				Position actual value				1A01:0 = 0 1A01:1 = 606C0020 1A01:2 = 60640020 1A01:0 = 2

Tab. 92 – PVM recommended PDO mapping

SyncManagers PDO assignment:

SyncManagers	PDOs				Assignment
SM2	RPDO1	RPDO2			1C12:0 = 0 1C12:1 = 1600 1C12:2 = 1601 1C12:0 = 2
SM3	TPDO1	TPDO2			1C13:0 = 0 1C13:1 = 1A00 1C13:2 = 1A01 1C13:0 = 2

Tab. 93 - PVM recommended SM assignment



### 12.1.4. Homing (6)

This mode is used to seek the home position. It is possible to specify the speeds, acceleration and the method of homing. A further object, *home offset*, allows to associate a specific value for the home position.

Two *homing speeds* are available: in a typical cycle the faster speed is used to find the home switch and the slower speed is used to release the home switch or to perform the following index cycle.

From *switched on* status, the master shall set modes of operation to 6, write control word 0x0F to enable operation. Following signals are used:

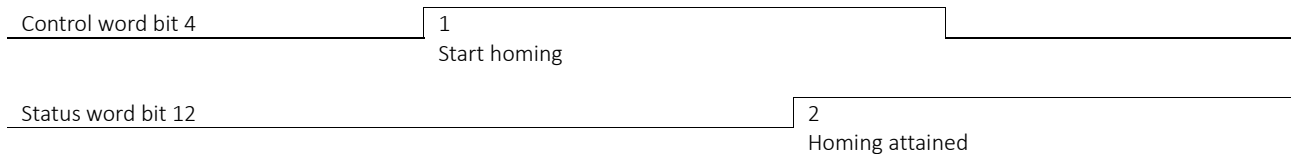


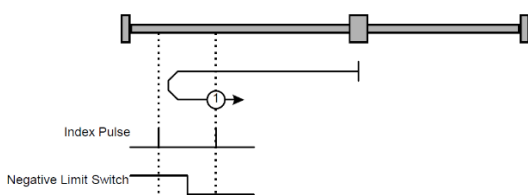
Fig. 21 - homing control

1. The homing movement starts upon a high level of *control word* bit 4, *start homing*. Upon edge detection FD sets also acceleration and deceleration equal to *homing acceleration*. The movement stops in case of low level of the same bit or in case of the *halt* bit.
2. Once the drive completes the homing procedure, status word bit 12, *homing attained*, and bit 10, *target reached* are raised.
3. In case of error during homing status word bit 13, *homing error* is raised.

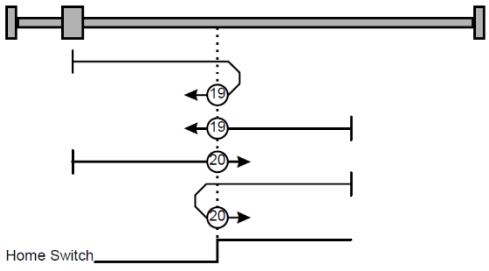
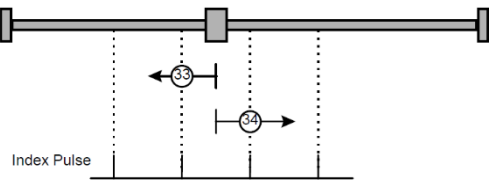
The *homing methods* available are presented in Tab. 70. The method name is composed by up to four abbreviations:

- UP and DW define the motor direction (UP is CW towards increasing positions, DW is CCW towards decreasing positions when no inversion is applied)
- LS and HS define if the axis calibration is performed on a homing switch or limit switch. Multipurpose inputs need to be configured accordingly.
- MK is the index cycle, i.e. a motor rotation to the absolute encoder position destination. An offset to the single turn zero-encoder can be configured by 0x2005:09, position offset.
- MECH is a movement towards mechanical stop.

Homing methods	Values	Description
NEAREST_MK	-33	Motor will rotate in the direction of the single-turn zero encoder following the shortest path.
UP_MECH	-18	Motor will rotate at <i>homing speed research</i> CW towards increasing positions. When the drive detects that the motor shaft is mechanically stopped, calibration is completed.
DW_MECH	-17	Motor will rotate at <i>homing speed research</i> CCW towards decreasing positions. When the drive detects that the motor shaft is mechanically stopped, homing is completed.
UP_MECH_MK	-2	Motor will rotate at <i>homing speed research</i> CW towards increasing positions. When the drive detects that the motor shaft is mechanically stopped, indexer cycle in CCW direction is executed. At the end of indexer cycle, homing is completed.
DW_MECH_MK	-1	Motor will rotate at <i>homing speed research</i> CCW towards decreasing positions. When the drive detects that the motor shaft is mechanically stopped, indexer cycle in CW direction is executed. At the end of indexer cycle, homing is completed.
DW_LS_NO_MK Homing on negative limit switch and index pulse	1	Motor will rotate at <i>homing speed research</i> CCW towards decreasing positions till a positive edge of hardware limit switch down input signal is met (at least one multipurpose input needs to be configured as limit switch down, i.e. input configuration register equals to 7). It will decelerate to zero speed, wait <i>time stop</i> milliseconds and accelerate in opposite direction at <i>homing speed release</i> till the negative edge of the same signal is met and then it continues moving at the same direction and speed till the index cycle is completed. <i>Note:</i> <i>It is important the switch does not commute in the same position of the zero encoder, otherwise there can be uncertainty of one</i>



Homing methods	Values	Description
		<i>revolution.</i>
UP_LS_NO_MK Homing on positive limit switch and index pulse	2	Motor will rotate at <i>homing speed research</i> CW towards increasing positions till a positive edge of hardware limit switch up input signal is met (at least one multipurpose input needs to be configured as limit switch up, i.e. input configuration register equals to 5). It will decelerate to zero speed, wait <i>time stop</i> milliseconds and accelerate in opposite direction at <i>homing speed release</i> till the negative edge of limit switch signal is met and then it continues moving at the same direction and speed till the index cycle is completed. <i>Note:</i> <i>It is important the switch does not commute in the same position of the zero encoder, otherwise there can be uncertainty of one revolution.</i>
UP_HS_NO_MK Homing on positive home switch and index pulse	3	Motor will rotate at <i>homing speed research</i> CW towards increasing positions till a positive edge of homing switch input signal is met (at least one input needs to be configured as homing switch, i.e. input configuration register equals to 2). It decelerates to zero speed, waits <i>time stop</i> milliseconds and accelerates in opposite direction at <i>homing speed release</i> till the negative edge of the same signal is met and then it continues moving at the same direction and speed till the index cycle is completed. <i>Note:</i> <i>It is important the switch does not commute in the same position of the zero encoder, otherwise there can be uncertainty of one revolution.</i>
UP_HS_NC_MK Homing on positive home switch and index pulse	4	Motor will rotate at <i>homing speed research</i> CW towards increasing positions till a negative edge of homing switch input signal is met (at least one input needs to be configured as homing switch, i.e. input configuration register equals to 3). It decelerates to zero speed, waits <i>time stop</i> milliseconds and accelerates in opposite direction at <i>homing speed release</i> till the positive edge of the same signal is met and then it continues moving at the same direction and speed till the index cycle is completed. <i>Note:</i> <i>It is important the switch does not commute in the same position of the zero encoder, otherwise there can be uncertainty of one revolution.</i>
DW_HS_NO_MK	5	Motor will rotate at <i>homing speed research</i> CCW towards decreasing positions till a positive edge of homing switch input signal is met (at least one input needs to be configured as homing switch, i.e. input configuration register equals to 2). It decelerates to zero speed, waits <i>time stop</i> milliseconds and accelerates in opposite direction at <i>homing speed release</i> till the negative edge of the same signal is met and then it continues moving at the same direction and speed till the index cycle is completed.
DW_HS_NC_MK	6	Motor will rotate at <i>homing speed research</i> CCW towards decreasing positions till a negative edge of homing switch input signal is met (at least one input needs to be configured as homing switch, i.e. input

Homing methods	Values	Description
		configuration register equals to 3). It decelerates to zero speed, waits <i>time stop</i> milliseconds and accelerates in opposite direction at <i>homing speed release</i> till the positive edge of the same signal is met and then it continues moving at the same direction and speed till the index cycle is completed.
-	-	
DW_LS_NO	17	Motor will rotate at <i>homing speed research</i> CCW towards decreasing positions till a positive edge of hardware limit switch down input signal is met (at least one multipurpose input needs to be configured as limit switch down, i.e. input configuration register equals to 7). It will decelerate to zero speed, wait <i>time stop</i> milliseconds and accelerate in opposite direction at <i>homing speed release</i> till the negative edge of limit switch signal is met.
UP_LS_NO	18	Motor will rotate at <i>homing speed research</i> CW towards increasing positions till a positive edge of hardware limit switch up input signal is met (at least one multipurpose input needs to be configured as limit switch up, i.e. input configuration register equals to 5). It will decelerate to zero speed, wait <i>time stop</i> milliseconds and accelerate in opposite direction at <i>homing speed release</i> till the negative edge of limit switch signal is met.
UP_HS_NO	19	Motor will rotate at <i>homing speed research</i> CW towards increasing positions till a positive edge of homing switch input signal is met (at least one input needs to be configured as homing switch, i.e. input configuration register equals to 2). It decelerates to zero speed, waits <i>time stop</i> milliseconds and accelerates in opposite direction at <i>homing speed release</i> till the negative edge of the same signal is met.
UP_HS_NC	20	Motor will rotate at <i>homing speed research</i> CW towards increasing positions till a negative edge of homing switch input signal is met (at least one input needs to be configured as homing switch, i.e. input configuration register equals to 3). It decelerates to zero speed, waits <i>time stop</i> milliseconds and accelerates in opposite direction at <i>homing speed release</i> till the positive edge of the same signal is met.
		
DW_HS_NO	21	Motor will rotate at <i>homing speed research</i> CCW towards decreasing positions till a positive edge of homing switch input signal is met (at least one input needs to be configured as homing switch, i.e. input configuration register equals to 2). It decelerates to zero speed, waits <i>time stop</i> milliseconds and accelerates in opposite direction at <i>homing speed release</i> till the negative edge of the same signal is met.
DW_HS_NC	22	Motor will rotate at <i>homing speed research</i> CCW towards decreasing positions till a negative edge of homing switch input signal is met (at least one input needs to be configured as homing switch, i.e. input configuration register equals to 3). It decelerates to zero speed, waits <i>time stop</i> milliseconds and accelerates in opposite direction at <i>homing speed release</i> till the positive edge of the same signal is met.
-	-	
DW_MK	33	Index cycle CCW towards decreasing positions
UP_MK	34	Index cycle CW towards increasing positions
		
CURR_POSITION	35	It sets current position equal to <i>home offset</i> .

Tab. 94 - Homing methods

**Notes:**

The homing attained bit is reset only when the homing movement starts. The information is kept passing to another mode of operation. This means that it is possible to switch back to homing mode and find the homing attained active prior starting the homing movement. As soon as the homing movement is started, homing attained is reset. Homing attained has the same meaning of Modbus status word bit axis calibrated, which can be restored from power on, ref. to 14.4 Homing.

### 12.1.5. Interpolated position (7)

CANopen only. Interpolated position mode is used to control multiple coordinated axis or a single axis with the need for time-synchronization of positions.

The master shall implement a path generation function, to provide the axis positions at every synchronized instant.

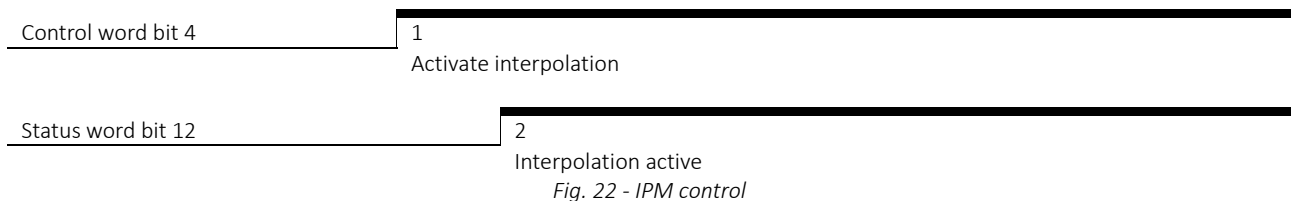
The *interpolation data record* contains the interpolated position set-points, expressed as absolute multi-turn position. This object has the dimensions of a record, because it would eventually be possible to specify a vector, e.g. position, speed and acceleration that the motor shall assume at each sync. FD1 requires just positions, that's why this object is a record with only two registers: number of entries and position. The record size is fixed and defined in the *size of data record* as sub-index of the *interpolation data configuration*.

The interpolated position mode allows a host controller to transmit a stream of interpolation data with an explicit time reference to a drive unit. The period between every sync object is pre-defined by the object *interpolation time period*.

FD drives support an input buffer of 32 positions, the interpolation data may be sent in bursts rather than continuously in real time. The available and the maximum size of the input buffer can be requested by a host using the *interpolation data configuration*. The buffer size is the number of interpolation data records which may be sent to a drive to fill the input buffer and it is not the size in Bytes.

There is no limit function for speed, acceleration and deceleration applied to the interpolation data. If the drive cannot execute the ordered command step accumulation limit alarm arises.

To activate the interpolated position mode, modes of operation shall be set to 7, operation shall be enabled and control word bit 4 shall be high. Slave will answer rising status word bit 12.



When the drive status is operation enable and the interpolated position mode is selected, the drive enters in interpolation inactive. The drive unit will accept input positions and will buffer it for interpolation calculations, but it does not move the motor. Interpolation active state is entered when the device is in operation enable state, the interpolated position mode is selected and activated with control word bit 4. The drive unit will accept input data and it will move the motor unwinding the buffered positions at every synchronized event.

Entering interpolated position mode or operation enabled clears the buffer.

Writing the interpolation position (0x60C1:1), data is loaded into the input buffer, organized as a FIFO, and the pointer of the buffer is incremented to the next buffer position.

Writing 0 to sub-index *buffer clear of interpolation data configuration* clears all positions buffered.

Once the interpolation is activated and sync is transmitted, interpolation position is immediately consumed, without delay, i.e. the drive will move the motor to be at that position at next sync signal. An input buffer for interpolation data records is not mandatory, although it eases the data exchange between a host and a drive unit when transmission of data has jitter in respect to synchronization event. The real-time requirements of the EtherCAT network decrease in this case, because the input buffer decouples the data processing in the drive from the data transmission via the bus line.

In order to follow a two- or more-dimensional curve through the space with a defined speed, a host (an interpolation controller or a PLC) calculates the positions for each set of coordinates which have to be reached at sync instants and transmits them to the axis.

Interpolated position mode can be operated via SyncManager synchronization or via DC synchronization.

Recommended PDO mapping:

PDO#	B0	B1	B2	B3	B4	B5	B6	B7	Mapping
RPDO1	Control word		Interpolated data record, sub-ix 1						1600:0 = 0 1600:1 = 60400010 1600:2 = 60C10120 1600:0 = 2
TPDO1	Status word		Position actual value						1A00:0 = 0 1A00:1 = 60410010 1A00:2 = 60640020 1A00:0 = 2

Tab. 95 - IPM recommended PDO mapping

SyncManagers PDO assignment:

SyncManagers	PDOs				Assignment
SM2	RPDO1				1C12:0 = 0 1C12:1 = 1600 1C12:0 = 1
SM3	TPDO1				1C13:0 = 0 1C13:1 = 1A00 1C13:0 = 1

Tab. 96 – IPM recommended SM assignment

Note:

Before entering IP mode, make sure that the first position that will be transmitted is the same or in proximity of 0x6062, position demand value (valid only if the drive is switched on).

### 12.1.6. Cyclic synchronous position (8)

EtherCAT only. This mode of operation is very similar to interpolated position mode, with some differences. In order control the motor using cyclic synchronization, the master shall still generate a position trajectory. This means that cyclically the master shall calculate for the drive its position at every synchronization period. On the other hand, the drive interpolates between each received target position with a resolution of 50  $\mu$ s and it controls the torque based on the encoder speed and position.

Recommended PDO mapping:

PDO#	B0	B1	B2	B3	B4	B5	B6	B7	Mapping
RPDO1	Control word		Target position						1600:0 = 0 1600:1 = 60400010 1600:2 = 607A0020 1600:0 = 2
TPDO1	Status word		Position actual value						1A00:0 = 0 1A00:1 = 60410010 1A00:2 = 60640020 1A00:0 = 2

Tab. 97 – CSPM recommended PDO mapping

SyncManagers PDO assignment:

SyncManagers	PDOs				Assignment
SM2	RPDO1				1C12:0 = 0 1C12:1 = 1600 1C12:0 = 1
SM3	TPDO1				1C13:0 = 0 1C13:1 = 1A00 1C13:0 = 1

Tab. 98 – CSPM recommended SM assignment

The target position is an absolute value. Feedbacks, such as position actual value and velocity actual value can be used to monitor motor movement, there is no need to close the position loop on master side. The drive itself calculate autonomously the correct amount of speed and torque needed to follow the target position setpoint.

CSP can be operated with SyncManager synchronization or with DC synchronization.

*Note:*

*Compared to interpolated position mode, there is no buffer of setpoints, sync period is not explicit, instead of using a record, position is transmitted directly to an integer object target position.*

### 12.1.7. Cyclic synchronous velocity (9)

EtherCAT only. In this mode of operation, master shall generate a velocity trajectory, hence provide the target velocity to the drive using cyclic synchronization. This means that cyclically the master shall calculate for the drive its velocity at every synchronization period.

Compared to profile velocity mode, where the drive receives just the speed regime setpoint and autonomously accelerates/decelerates using profile acceleration/deceleration parameters, in cyclic synchronous velocity mode the master shall provide the instantaneous velocity during acceleration and deceleration at every synchronization period.

Recommended PDO mapping:

PDO#	B0	B1	B2	B3	B4	B5	B6	B7	Mapping
RPDO1	Control word		Target velocity						1600:0 = 0 1600:1 = 60400010 1600:2 = 60FF0020 1600:0 = 2
TPDO1	Status word		Velocity actual value						1A00:0 = 0 1A00:1 = 60410010 1A00:2 = 606C0020 1A00:0 = 2

Tab. 99 - CSVM recommended PDO mapping

SyncManagers PDO assignment:

SyncManagers	PDOs				Assignment
SM2	RPDO1				1C12:0 = 0 1C12:1 = 1600 1C12:0 = 1
SM3	TPDO1				1C13:0 = 0 1C13:1 = 1A00 1C13:0 = 1

Tab. 100 - CSVM recommended SM assignment

Feedbacks, such as position actual value and velocity actual value shall be used to monitor motor movement, but there is no need to close the velocity, torque loops on master side. The drive itself calculates autonomously the correct amount of speed and torque needed to follow the target velocity setpoint.

CSV can be operated with SyncManager synchronization or with DC synchronization.

## 13. Object dictionary

Auxind FD drives with CANopen/EtherCAT Communications, use the object dictionary as a base for communications.

All objects are assigned four-digit hexadecimal numbers in the areas shown in the following table.

Indexes	Area	Description
0x0000 – 0x0FFF	Data types	Data types definitions
0x1000 – 0x1FFF	CoE communication	Definitions of variables that can be used by all servers for designated communications
0x2000 – 0x2FFF	Manufacturer specific	Variables with common definitions for all Auxind products
0x6000 – 0x9FFF	Device profile specific	CiA402 servo drive profile

Tab. 101 - Object dictionary areas

### 13.1. CoE communication

Index	Sub-index	bit	Type	Access	Object name	Description	EtherCAT	CANopen
0x1000	-	-	U32	RO	Device type	0x40192 = Stepper	✓	✓
0x1001	-	0	U8	RO	Error register	General alarm	✓	✓
		1				Current	✓	✓
		2				Voltage	✓	✓
		3				Temperature	✓	✓
		5				Drive error (device profile specific)	✓	✓
		7				Position error (manufacturer specific)	✓	✓
		7						
0x1003	0		ARR	RW	Pre-defined error	Number of errors	✓	✓
	1 – 8					Error codes	✓	✓
0x1008	-	-	STR	RO	Device name	"Auxind – FD2.1E", "Auxind – FD1.1E", etc.	✓	✓
0x1009	-	-	STR	RO	Hardware version	"2.1E", "1.1E", etc.	✓	✓
0x100A	-	-	STR	RO	Software version	"V6.24"	✓	✓
0x100B	-	-	STR	RO	Bootloader version	"V0.22"	✓	✓
0x100C	-	-	U16	RW	Guard time	Ref. to 11.2.1 Network Management Objects (NMT)		✓
0x100D	-	-	U8	RW	Life time factor	Ref. to 11.2.1 Network Management Objects (NMT)		✓
0x1010	-	-	ARR	RW	Store parameters		✓	✓
	0	-	U8	RO	Number of entries	1	✓	✓
	1	-	U32	RW	Save all parameters	R: 1, Device saves parameters on command W: 0x65766173, signature to save all parameters	✓	✓
0x1011	-	-	ARR	RW	Restore default parameters		✓	
	0	-	U8	RO	Number of entries		✓	
	1	-	U32	RW	Restore all default parameters	R: 1, Device restores parameters W: 0x64616F6C, signature to restore all parameters from flash memory	✓	
0x1016	-	-	ARR	RW	Consumer heartbeat time			✓
	0	-	U8	RO	Number of entries			✓
	1	-	U32	RW	Consumer heartbeat time	Ref. to 11.2.1 Network Management Objects (NMT)		✓
0x1017	-	-	U16	RW	Producer heartbeat time	Ref. to 11.2.1 Network Management Objects (NMT)		✓
0x1018	-	-	REC		Identity object		✓	✓
	0	-	U8	RO	Number of entries	4	✓	✓
	1	-	U32	RO	Vendor ID	0x0F10 (EtherCAT) 0x0531 (CANopen)	✓	✓
	2	-	U32	RO	Product code	FD1.1E: 0x0474 FD1.1EC, FD1.2EC: 0x0488	✓	✓



Index	Sub-index	bit	Type	Access	Object name	Description	Ether CAT	CANopen
						FD2.1E: 0x04D8 FD2.1EC: 0x04EC		
	3	-	U32	RO	Revision number	0	✓	✓
	4	-	U32	RO	Serial number	0	✓	✓
0x1400	-	-	REC	-	Pdo_1_Rx_Param_record			✓
	0	-	U8	RO	Number of entries			✓
	1	-	U32	RW	Pdo_1_Rx_COB_ID	0x200 + Node-ID		✓
	2	-	U8	RW	Pdo_1_Rx_Type	0xFF		✓
0x1401	-	-	REC	-	Pdo_2_Rx_Param_record			✓
	0	-	U8	RO	Number of entries			✓
	1	-	U32	RW	Pdo_2_Rx_COB_ID	0x300 + Node-ID		✓
	2	-	U8	RW	Pdo_2_Rx_Type	0xFF		✓
0x1402	-	-	REC	-	Pdo_3_Rx_Param_record			✓
	0	-	U8	RO	Number of entries			✓
	1	-	U32	RW	Pdo_3_Rx_COB_ID	0x400 + Node-ID		✓
	2	-	U8	RW	Pdo_3_Rx_Type	0xFF		✓
0x1403	-	-	REC	-	Pdo_4_Rx_Param_record			✓
	0	-	U8	RO	Number of entries			✓
	1	-	U32	RW	Pdo_4_Rx_COB_ID	0x500 + Node-ID		✓
	2	-	U8	RW	Pdo_4_Rx_Type	0xFF		✓
0x1600	-	-	REC	-	Pdo_1_Rx_Mapping_record		✓	✓
	0	-	U8	RW	Number of entries	1	✓	✓
	1...7	-	U32	RW	Pdo_1_Rx_mapping_1...7	0x60400010	✓	✓
0x1601	-	-	REC	-	Pdo_2_Rx_Mapping_record		✓	✓
	0	-	U8	RW	Number of entries	2	✓	✓
	1...7	-	U32	RW	Pdo_2_Rx_mapping_1...7	0x60400010; 0x60600008	✓	✓
0x1602	-	-	REC	-	Pdo_3_Rx_Mapping_record		✓	✓
	0	-	U8	RW	Number of entries	2	✓	✓
	1...7	-	U32	RW	Pdo_3_Rx_mapping_1...7	0x60400010; 0x607A0020	✓	✓
0x1603	-	-	REC	-	Pdo_4_Rx_Mapping_record		✓	✓
	0	-	U8	RW	Number of entries	2	✓	✓
	1...7	-	U32	RW	Pdo_4_Rx_mapping_1...7	0x60400010; 0x60FF0020	✓	✓
0x1800	-	-	REC	-	Pdo_1_Tx_Param_record			✓
	0	-	U8	RO	Number of entries			✓
	1	-	U32	RW	Pdo_1_Tx_COB_ID	0x180		✓
	2	-	U8	RW	Pdo_1_Tx_Type	0		✓
	3	-	U16	RW	Pdo_1_Tx_Inhibit_Time			✓
	5	-	U16	RW	Pdo_1_Tx_Event_Timer			✓
0x1801	-	-	REC	-	Pdo_2_Tx_Param_record			✓
	0	-	U8	RO	Number of entries			✓
	1	-	U32	RW	Pdo_2_Tx_COB_ID	0x280		✓
	2	-	U8	RW	Pdo_2_Tx_Type	0		✓
	3	-	U16	RW	Pdo_2_Tx_Inhibit_Time			✓
	5	-	U16	RW	Pdo_2_Tx_Event_Timer			✓
0x1802	-	-	REC	-	Pdo_3_Tx_Param_record			✓
	0	-	U8	RO	Number of entries			✓
	1	-	U32	RW	Pdo_3_Tx_COB_ID	0x380		✓
	2	-	U8	RW	Pdo_3_Tx_Type	0		✓
	3	-	U16	RW	Pdo_3_Tx_Inhibit_Time			✓
	5	-	U16	RW	Pdo_3_Tx_Event_Timer			✓
0x1803	-	-	REC	-	Pdo_4_Tx_Param_record			✓
	0	-	U8	RO	Number of entries			✓
	1	-	U32	RW	Pdo_4_Tx_COB_ID	0x480		✓

Index	Sub-index	bit	Type	Access	Object name	Description	Ether CAT	CANopen
	2	-	U8	RW	Pdo_4_Tx_Type	0		✓
	3	-	U16	RW	Pdo_4_Tx_Inhibit_Time			✓
	5	-	U16	RW	Pdo_4_Tx_Event_Timer			✓
0x1A00	-	-	REC	-	Pdo_1_Tx_Mapping_record		✓	✓
	0	-	U8	RW	Number of entries	1	✓	✓
	1...7	-	U32	RW	Pdo_1_Tx_mapping_1...7	0x60410010	✓	✓
0x1A01	-	-	REC	-	Pdo_2_Tx_Mapping_record		✓	✓
	0	-	U8	RW	Number of entries	2	✓	✓
	1...7	-	U32	RW	Pdo_2_Tx_mapping_1...7	0x60410010; 0x60610008	✓	✓
0x1A02	-	-	REC	-	Pdo_3_Tx_Mapping_record		✓	✓
	0	-	U8	RW	Number of entries	2	✓	✓
	1...7	-	U32	RW	Pdo_3_Tx_mapping_1...7	0x60410010; 0x60630020	✓	✓
0x1A03	-	-	REC	-	Pdo_4_Tx_Mapping_record		✓	✓
	0	-	U8	RW	Number of entries	2	✓	✓
	1...7	-	U32	RW	Pdo_4_Tx_mapping_1...7	0x60410010; 0x606C0020	✓	✓
0x1C00	-	-	REC	-	SyncManagers comm. type		✓	
	0	-	U8	RO	Number of entries	4	✓	
	1	-	U8	RO	SM0 communication type	1: Mailbox Out	✓	
	2	-	U8	RO	SM1 communication type	2: Mailbox In	✓	
	3	-	U8	RO	SM2 communication type	3: Buffer Out	✓	
	4	-	U8	RO	SM3 communication type	4: Buffer In	✓	
0x1C12	-	-	REC	-	SM2 PDO assignment		✓	
	0	-	U8	RW	Number of entries	2	✓	
	1...4	-	U16	RW	SM2 PDO assignment_1...4	0x1600; ...	✓	
0x1C13	-	-	REC	-	SM3 PDO assignment		✓	
	0	-	U8	RW	Number of entries	2	✓	
	1...4	-	U16	RW	SM3 PDO assignment_1...4	0x1A00; ...	✓	
0x1C32	-	-	REC	-	Sync manager 2 parameter		✓	
	0	-	U8	RO	Number of entries		✓	
	1		U16	RW	SM2 synchronization type	0x00: free run 0x01: synchronous with SM 0x02: DC SYNC0 0x03: DC SYNC1	✓	
	2		U32	RW	SM2 cycle time	Expressed in ns. Free run: not used Synchronous with SM2: time between two SM2 events DC SYNC0 and DC SYNC1: time between two SYNC0.	✓	
	4	0	U16	RO	SM2 sync types supported	1: Free run supported	✓	
		1				1: Synchronous with SM2 supported	✓	
		2				1: Synchronous with DC SYNC0 supported	✓	
		3				1: Synchronous with DC SYNC1 supported	✓	
		5				Shift settings	✓	
		6				0: no output shift supported	✓	
	5	-	U32	RO	SM2 minimum cycle time	1'000'000 ns	✓	
	6				Calc and copy time		✓	
	11				SM-event missed		✓	
	12				Cycle time too small	This error counter is incremented when the cycle time is too small so that the local cycle cannot be completed and input data cannot be provided before the next SM event.	✓	

Index	Sub-index	bit	Type	Access	Object name	Description	Ether CAT	CANopen
	20				Sync error		✓	
0x1C33	-	-	REC	-	Sync manager 3 parameter		✓	
	0	-	U8	RO	Number of entries		✓	
	1		U16	RW	SM3 synchronization type	0x00: free run 0x01: synchronous with SM 0x02: DC SYNC0 0x03: DC SYNC1	✓	
	2		U32	RW	SM3 cycle time	Expressed in ns. Free run: not used Synchronous with SM2: time between two SM2 events DC SYNC0 and DC SYNC1: time between two SYNC0.	✓	
	4	0	U16	RO	SM3 sync types supported	1: Free run supported	✓	
		1				1: Synchronous with SM supported	✓	
		2				1: Synchronous with DC SYNC0 supported	✓	
		3				1: Synchronous with DC SYNC1 supported	✓	
		5				Shift settings	✓	
		6				0: no output shift supported	✓	
	5	-	U32	RO	SM3 minimum cycle time	1'000'000 ns	✓	
	6				Calc and copy time		✓	
	11				SM-event missed		✓	
	12				Cycle time too small	This error counter is incremented when the cycle time is too small so that the local cycle cannot be completed and input data cannot be provided before the next SM event.	✓	
	20				Sync error		✓	
0x1F50	-	-	ARR	-	Download program data	Ref. to 11.2.6 SDO block download		✓
	0	-	DOM	WO	Program firmware			✓
	1	-	DOM	WO	Program parameters			✓
0x1F52	0	-	ARR	-	Verify application software			✓
	1	-	U32	RO	Application software date	Contains the number of days since January 1, 1984.	✓	✓
	2	-	U32	RO	Application software time	Contains the number of milliseconds after midnight	✓	✓

Tab. 102 - Communication objects

### 13.1.1. 0x1000, Device type

Read only unsigned 32, it is composed of lower 16-bit field which describes the device profile that is used (DSP 402) and higher 16 bits which describes the device type, which for stepper motor is equal to 0x4. Hence 0x40192.

### 13.1.2. 0x1001, Error register

FD drives map internal errors in this byte. It is part of an emergency object. Only the fault reset command is able to reset this register.

Bit number	Description
0	Generic error
1	Current error
2	Voltage error
3	Temperature error
4	Communication error
5	Device profile specific
6	Always 0
7	Manufacturer specific

Tab. 103 - Error register bits

### 13.1.3. 0x1003, Pre-defined error

The object at index 0x1003 holds the alarms that have occurred and have been signaled transmitting CANopen EMCY object, plus all the communication errors. In doing so, it provides an error history.

The entry at sub-index 0 contains the number of actual errors that are recorded in the array starting at sub-index 1. It is RW and ranges from 0 to 8. Writing a 0 to sub-index 0 deletes the entire error history (empties the array). Values higher than 0 are not allowed to write. This leads to an abort message (error code: 0x06090030).

Every new error is stored at sub-index 1, the older ones move down the list, made of maximum 8 elements.

The error numbers are of type U32. The error code is contained in the lower 2 bytes (LSB) and the additional information is included in the upper 2 bytes (MSB).

### 13.1.4. 0x100C / 0x100D, Guard time and Life time factor

CANopen only. The objects at index 100Ch and 100Dh include the guard time in milliseconds and the life time factor.

The life time factor multiplied with the guard time gives the life time for the Life Guarding Protocol. It is 0 if not used. Life guard gets active from the first NMT received since Guard time and Life time factor differs from zero.

In case of life guarding event (i.e. master didn't transmit NMT for the requested time), the slave stops all the movements and reset the object dictionary. Error code 0x8130 can be read in 0x1003, pre-defined error field.

### 13.1.5. 0x1010, Store parameters

This object supports the saving of parameters in flash. By read access it provides information about its saving capabilities.

In order to avoid storage of parameters by mistake, storage is only executed when a specific signature is written to the appropriate Sub-Index. The signature is 0x65766173.

On reception of the correct signature in sub-index 1, the drive stores the current parameters in flash, then confirms the SDO transmission (initiate download response). If the storing failed, the device responds with an Abort SDO Transfer (abort code: 0x06060000). If a wrong signature is written, the device refuses to store and responds with Abort SDO Transfer (abort code: 0x0800002x).

### 13.1.6. 0x1011, Restore default parameters

EtherCAT only. With this object all the default values of parameters are restored. By read access the device provides information about its capabilities to restore these values.

In order to avoid the restoring of default parameters by mistake, restoring is only executed when a specific signature is written to the appropriate sub-index. The signature is 0x 64 61 6F 6C.

On reception of the correct signature in the appropriate sub-index the device restores the default parameters and then confirms the SDO transmission (initiate download response). If the restoring failed, the device responds with an Abort

SDO Transfer (abort code: 0x06060000). If a wrong signature is written, the device refuses to restore the defaults and responds with an Abort SDO Transfer (abort code: 0x0800002x).

### 13.1.7. 0x1016, Consumer heartbeat time

CANopen only. The drive monitors a heartbeat producer. In case of missed

The consumer heartbeat time object indicates the expected heartbeat cycle time. The drive monitors a heartbeat producer. The monitoring start after the reception of the first heartbeat.

The heartbeat time shall be given in multiples of 1 ms.

In case the heartbeat time elapses without any reception of the heartbeat, the drive stops all the movements and reset the object dictionary. Error code 0x8130 can be read in 0x1003, pre-defined error field.

Bit number	Value	Description
31-24	0	Reserved
23-16	Node-ID	Node-ID of the heartbeat to be monitored
15-0	Heartbeat time	Maximum elapsed time

Tab. 104 - Consumer heartbeat time

### 13.1.8. 0x1017, Producer heartbeat time

CANopen only. The producer heartbeat time defines the cycle time of the heartbeat, produced by the drive and monitored by a consumer, that generally is the NMT master, but it can also be another slave drive. The producer heartbeat time is 0 if it not used. The time has to be a multiple of 1ms.

### 13.1.9. 0x140x / 0x180x, RPDOx / TPDOx parameters

CANopen only. The object at index 0x140x defines the COB-ID and type of RPDOs 1, 2, 3 and 4.

The object at index 0x180x defines the COB-ID and type of TPDOs 1, 2, 3 and 4.

Sub-index 0 contains the number of entries, equal to 2, type U8.

Sub-index 1 contains the COB-ID of the PDO and its status, type U32:

Bit number	Value	Description
31	Activation	0: PDO is activated 1: PDO is not active
30	1: RTR not allowed 0: RTR allowed	RPDO: RTR is not allowed TPDO: RTR is allowed
29 – 11	Always 0	Standard ID
10 – 0	COB-ID	Default configuration: RPDO: 0x200 + (0x100 * RPDO number) + Node-ID TPDO: 0x180 + (0x100 * TPDO number) + Node-ID

Tab. 105 - PDO parameters

The PDO active/not active allows to select which PDOs are used in the operational state. There can be PDOs fully configured (e.g. by default) but not used, and therefore set to not active.

FD drives support the standard CAN frame type only and do not support Remote Frames, an attempt to set bit 29 to 1 or bit 30 to 0 is responded with an abort message (abort code: 0609 0030h).

It is not allowed to change bit 0-29 while the PDO is active (Bit 31 = 0).

Sub-index 2 contains the transmission type. Any attempt to set type higher than 240 or lower than 254 is responded with an abort message (abort code: 0609 0030h).

Synchronous RPDO:

- A transmission type of n means that the message received after the occurrence of a SYNC is passed to the application with the occurrence of the following SYNC, independent of the transmission rate specified by the transmission type. (0 ≤ n ≤ 240).

Asynchronous RPDO:

- A transmission type of 254 or 255 means that the message is passed directly to the application as soon as it is received.

Synchronous TPDO:

- A transmission type of 0 means that the drive transmits the frame after occurrence of the SYNC but acyclic (not periodically),

i.e. only if an event occurred before the SYNC (when the content of the TPDO changes).

- A transmission type of  $n$  means that the message is transmitted with every  $n$ -th SYNC object. ( $0 < n < 242$ ).

Synchronous TPDO RTR-only:

- A transmission type of 252 means that the drive transmits the frame only if it receives the corresponding RTR, after the occurrence of the following SYNC.

Asynchronous TPDO:

- A transmission type of 254 or 255 means that the message is transmitted when the event occurs, without any relation to the SYNC.

Asynchronous TPDO RTR-only:

- A transmission type of 253 means that the drive transmits the frame after the reception of the corresponding RTR.

Sub-index 3 of TPDO parameters contains the inhibit time. Minimum time between asynchronous TPDO transmission.

Sub-index 5 of TPDO parameters contains the event timer. Maximum time between asynchronous TPDO transmission.

### 13.1.10. 0x160x / 0x1A0x, RPDOx / TPDOx mapping records

The objects at indexes 0x1600, 0x1601, 0x1602, 0x1603 define the mapping of RPDO 1, 2, 3 and 4.

The objects at indexes 0x1A00, 0x1A01, 0x1A02, 0x1A03 define the mapping of TPDO 1, 2, 3 and 4.

To create the PDO object mapping, first the PDO has to be deleted, the sub-index 0, number of objects mapped, type U8, must be set to 0 (PDO mapping is deactivated).

Then the sub-indexes 1-8, type U32, can be remapped writing object index, sub-index and number of bits of the object to be mapped into the PDO (number of bits can be only a multiple of 8, i.e. only Bytes can be mapped).

After all objects are mapped sub index 0 shall be set to the valid number of mapped objects.

Byte 3	Byte 2	Byte 1	Byte 0
Index_H	Index_L	Sub-index	Number of bits

Tab. 106 - PDO mapping records

Ref. to default mapping for having an example of the format.

If the change of the PDO mapping cannot be executed (e.g. the PDO length is exceeded or the SDO client attempts to map an object that cannot be mapped) FD responds with an Abort SDO Transfer Service.

PDO mapping can be performed only in pre-operational state.

### 13.1.11. 0x1C12, 0x1C13, SM2 and SM3 PDO assignment records

EtherCAT only. After mapping the objects into the PDOs, the PDOs have to be assigned to the SyncManagers, more precisely RPDOs shall be assigned to SM2, TPDOs to SM3.

The assignment is made only for those PDOs that are used in buffered mode, SyncManagers 0 and 1 are used in mailbox mode and for this reason they don't have PDO assigned.

Since there are 4 RPDOs and 4 TPDOs, objects 0x1C12 and 0x1C13 have a maximum of 4 entries each.

In a similar way of PDO mapping, SM assignment shall be performed by first setting to zero the sub-index 0, writing the PDOs to sub-indexes, then finally writing to sub-index 0 the number of PDOs assigned.

### 13.1.12. 0x1C32, 0x1C33, SM2 and SM3 parameters

EtherCAT only. There are mainly three types of synchronizations: Free run (not synchronized), synchronization with SM2 and synchronization with DC.

By default, the drive works synchronized with SM2. In case of need, the master can configure the ESC registers 0x0981, SYNC activation and 0x09A0, SYNC0 cycle time to activate the DC mode in the passage between pre-operational and safe operational. Reading from the ESC these registers, the drive microcontroller is able to recognize the type and period of synchronization to be used.

The received configuration can also be monitored and configured by the master via SDOs through records 0x1C32, 0x1C33 for SM2 and SM3.

EtherCAT offers many different types of synchronization, only a part of them relevant for stepper motor application has been implemented:

- Synchronization type

0 = free run,

1 = SM2 synchronous

2 = DC SYNC0 synchronous

3 = DC SYNC1 synchronous

- Cycle time

Period of time between two events in ns.

- Minimum cycle time

Minimum period of time between two events, i.e. 1'000'000 ms.

### **13.1.13. 0x1F50, Download program data record**

CANopen only. At its sub-indexes is possible to perform a SDO download block transfer of the firmware and the parameters. In particular:

Sub-ix 1: firmware

Sub-ix 2: parameters

Refer to CANopen SDO download block transfer for additional information.

### **13.1.14. 0x1F52, Verify application software**

To support verification of the version of the firmware.

Sub-ix 1: application software date contains the number of days since January 1, 1984.

Sub-ix 2: application software time contains the number of milliseconds after midnight (00:00).

## 13.2. Manufacturer specific

Index	Sub-index	bit	Type	Access	Object name	Description	Ether CAT	CANopen
0x2003	-	-	U32	RW	Delta-stop steps	Initialized to Cycle 0 delta stop steps	✓	✓
0x2005	0-255	-	ARR	RW	Modbus registers array 1		✓	✓
0x2006	0-...	-	ARR	RW	Modbus registers array 2		✓	✓
0x200C	-	-	ARR	-	Encapsulated SDO		✓	
	0	-	U8	RO	Number of entries	2	✓	
	1	-	U64	RW	SDO Request		✓	
	2	-	U64	RO	SDO Response		✓	
0x2010			ARR		Encoder position latch	It is used to measure the position of a sensor activation/deactivation	✓	
	0	-	U8	RO	Number of entries	2	✓	
	1		S32	RW	Position at rising edge		✓	
	2		S32	RW	Position at falling edge		✓	

Tab. 107 - Manufacturer specific objects

### 13.2.1. 0x2003, Delta-stop steps

Delta-stop steps are used on delta-stop cycles. This type of movement allows very accurate and fast positioning based on the activation of a sensor, detected on interrupt, to avoid time consuming homing cycles.

### 13.2.2. 0x2005 / 0x2006, Modbus register arrays

All the Modbus registers have been mapped into two arrays (they are two, since the maximum size of an array is 255 elements). Hence object 0x2005, sub 1 contains the Modbus register START, 0x2005, sub 2 contains STOP and so on. Sub-index 0 of both arrays 0x2005 and 0x2006 is the number of entries.

All the objects are 32 bits long, the same length of Modbus registers.

If Modbus register address divided by 2 is less than 255, it is mapped in 0x2005.

The sub-index is calculated as:  $(\text{Modbus register address} / 2) + 1$ .

If Modbus register address divided by 2 is higher or equal to 255, it is mapped in 0x2006.

The sub-index is calculated as:  $(\text{Modbus register address} / 2) - 254$ .

### 13.2.3. 0x200C, SDO encapsulated

EtherCAT only. SDO request and SDO response are two sub-indexes of the record SDO encapsulated. They are used to access the complete dictionary using PDO's, instead of normal SDO's. The main advantage is that PDOs are faster and they can be used to monitor essential information such as motor current, voltage, temperature, etc.

Their structure is very similar to SDOs, they are made of 8 Bytes:

E.g., in order to transmit the read command of 0x2006:05 Motor current, the content of the PDO shall be as follows:

B0	B1	B2	B3	B4	B5	B6	B7
0x40	0x06	0x20	0x05	0x00	0x00	0x00	0x00
Upload request	Index		Sub-index	Don't care			

The response of the slave for a current of 5'000 mA will be:

B0	B1	B2	B3	B4	B5	B6	B7
0x43	0x06	0x20	0x05	0x88	0x13	0x00	0x00
Upload response	Index		Sub-index	0x1388 = 5000			

E.g., in order to transmit the write command of 0x6083:00, profile acceleration = 2000 Kstep/s<sup>2</sup>, the content of the PDO shall be as



follows:

B0	B1	B2	B3	B4	B5	B6	B7
0x33	0x83	0x60	0x00	0xD0	0x07	0x00	0x00
Download request + Toggle	Index		Sub-index	0x07D0 = 2000			

The slave's response is:

B0	B1	B2	B3	B4	B5	B6	B7
0x70	0x83	0x60	0x00	0x00	0x00	0x00	0x00
Download response + Toggle	Index		Sub-index				

With the first write the toggle verification is positive, whether 0 or 1.

In case of abort, the slave's response is:

B0	B1	B2	B3	B4	B5	B6	B7
0x80	0x83	0x60	0x00	0x00	0x00	0x00	0x00
Abort	Index		Sub-index	Abort code.			

### 13.2.4. 0x2010, Encoder position latch

EtherCAT only. In order to use this functionality in CANopen you need to use objects 0x2005:249, Encoder latch rising, 0x2006:10 Encoder latch falling.

Rising and falling edges of the configured input trigger a rapid interrupt, which captures and latches the multi-turn encoder position. The latched positions can be accessed via the ENC\_LATCH\_RIS and ENC\_LATCH\_FAL Modbus registers, addressable in EtherCAT and CANopen as 0x2005:249, 0x2006:10, in EtherCAT also available via 0x2010:1, 0x2010:2.

### 13.3. DSP402 servo drive profile

Index	Sub-index	bit	Type	Access	Object name	Description	EtherCAT	CANopen
0x603F	-	-	U16	RO	Error code	Refer to detailed description	✓	✓
0x6040	-	0	U16	RW	Control word	Switch on	✓	✓
		1				Enable Voltage	✓	✓
		2				Quick Stop	✓	✓
		3				Enable Operation	✓	✓
		4				Operation mode specific	✓	✓
		5				Operation mode specific	✓	✓
		6				Operation mode specific	✓	✓
		7				Fault Reset	✓	✓
		8				Halt	✓	✓
		9				Reserved	✓	✓
		10				Reserved	✓	✓
		11				Operation mode specific	✓	✓
		12				Operation mode specific	✓	✓
		13				reserved	✓	✓
		14				Reserved	✓	✓
		15				Motor current boost control	✓	✓
0x6041	-	0	U16	RO	Status word	Ready to switch on	✓	✓
		1				Switched on	✓	✓
		2				Operation enabled	✓	✓
		3				Fault	✓	✓
		4				Voltage enabled	✓	✓
		5				Quick stop	✓	✓
		6				Switch on disabled	✓	✓
		7				Warning	✓	✓
		8				Torque limit	✓	✓
		9				Remote	✓	✓
		10				Target reached	✓	✓
		11				SW Limit Switches	✓	✓
		12				Operation mode specific	✓	✓
		13				Operation mode specific	✓	✓
		14				Position Upload	✓	✓
0x6060	-	-	S8	RW	Modes of operation	1: profile position mode 3: profile velocity mode 6: homing mode 7: interpolated position mode 8: cyclic synchronous position mode 9: cyclic synchronous velocity mode	✓	✓
0x6061	-	-	S8	RO	Modes of operation display	Actual modes of operation	✓	✓
0x6062	-	-	S32	RO	Position demand value	Position of the motor current phase setpoint	✓	✓
0x6063	-	-	S32	RO	Position actual internal value	Position of the encoder feedback	✓	✓
0x6064	-	-	S32	RO	Position actual value	Position of the encoder feedback	✓	✓
0x6065	-	-	U32	RW	Following error window	Maximum allowed absolute value of the difference between position demand value and position actual value	✓	✓
0x606B	-	-	S32	RO	Velocity demand value	Velocity of the motor current phase setpoint	✓	✓
0x606C	-	-	S32	RO	Velocity actual value	Velocity of the encoder feedback	✓	✓
0x6073	-	-	U16	RW	Max current	Maximum motor current	✓	✓
0x6078	-	-	U16	RO	Current actual value	Actual motor current	✓	✓
0x6079	-	-	U32	RO	DC link circuit voltage	Voltage at drive's power input	✓	✓
0x607A	-	-	S32	RW	Target position	Refer to profile position mode and cyclic synchronous position mode	✓	✓
0x607C	-	-	S32	RW	Home offset	Refer to homing mode	✓	✓

Index	Sub-index	bit	Type	Access	Object name	Description	EtherCAT	CANopen
0x607D			REC		Software position limit	Configured maximum and minimum software position limits	✓	✓
	0		U8	RO	Number of entries		✓	✓
	1		S32	RW	Min position limit		✓	✓
	2		S32	RW	Max position limit		✓	✓
0x607F	-	-	U32	RW	Max profile velocity	Upper limited to 300'000 [step/s]	✓	✓
0x6081	-	-	U32	RW	Profile velocity	Refer to profile position mode	✓	✓
0x6083	-	-	U32	RW	Profile acceleration	Refer to profile position mode, profile velocity mode [kstep/s <sup>2</sup> ]	✓	✓
0x6084	-	-	U32	RW	Profile deceleration	Refer to profile position mode, profile velocity mode [kstep/s <sup>2</sup> ]	✓	✓
0x6086	-	-	S16	RW	Motion profile type	0: Linear, 1: Parabolic, 2: s-curve.	✓	✓
0x6098	-	-	S8	RW	Homing method	Refer to homing mode	✓	✓
0x6099	-	-	REC	-	Homing speeds	Configured speeds used during homing procedure	✓	✓
	0	-	U8	RO	Number of entries		✓	✓
	1		U32	RW	Homing speed research		✓	✓
	2		U32	RW	Homing speed release		✓	✓
0x609A	-	-	U32	RW	Homing acceleration	It sets both acceleration and deceleration. Refer to homing mode. Range [1- 20'000] in [kstep/s <sup>2</sup> ]	✓	✓
0x60C0	-	-	S16	RW	Interpolation sub mode select	Period ≤ 2.5 ms 0: Linear interpolation Period > 2.5 1: Parabolic interpolation		✓
0x60C1			REC		Interpolation data record			✓
		0	U8	RO	Number of entries	1		✓
		1	S32	RW	Interpolated position			✓
0x60C2	-	-	REC		Interpolation time period	Expressed as: units x 10 <sup>index</sup>		✓
		0	U8	RO	Number of entries	2		✓
		1	U8	RW	Interpolation time units	Number of milliseconds		✓
		2	S8	RW	Interpolation time index	Power of ten: -3 represents milliseconds		✓
0x60C4			REC		Interpolation data configuration			✓
		0	U8	RO	Number of entries			✓
		1	U32	RO	Maximum buffer size	32		✓
		2	U32	RO	Actual buffer size	0		✓
		3	U8	RO	Buffer organization	0		✓
		4	U16	RO	Buffer position	0		✓
		5	U8	RO	Size of data record	1		✓
		6	U8	RW	Buffer clear	0		✓
0x60F4			S32	RO	Following error actual value	Difference between demand position and feedback position	✓	✓
0x60FD	-	0	U32	RO	Digital inputs	Negative limit switch	✓	✓
	-	1				Positive limit switch	✓	✓
	-	2				Home switch	✓	✓
	-	16				IN1	✓	✓
	-	17				IN2	✓	✓
	-	18				IN3	✓	✓
	-	19				IN4	✓	✓
	-	20				IN5	✓	✓
	-	21				IN6	✓	✓
	-	22				IN7	✓	✓
	-	24				OUT1	✓	✓

Index	Sub-index	bit	Type	Access	Object name	Description	EtherCAT	CANopen
	-	25				OUT2	✓	✓
	-	26				OUT3	✓	✓
0x60FE	-	-	REC		Digital outputs		✓	✓
	0	-	U8		Number of entries	2	✓	✓
	1	16	U32	RW	OUT1 physical		✓	✓
		17			OUT2 physical		✓	✓
		18			OUT3 physical		✓	✓
	2	16	U32	RW	OUT1 mask		✓	✓
		17			OUT2 mask		✓	✓
		18			OUT3 mask		✓	✓
0x60FF	-	-	S32	RW	Target velocity	Used in profile velocity mode and cyclic synchronous velocity [step/s].	✓	✓
0x6402	-	-	U16	RO	Motor type	9: micro-stepper motor	✓	✓
0x6502		0	U32	RO	Supported drive modes	Profile position = 1	✓	✓
		1				Velocity = 0	✓	✓
		2				Profile velocity = 1	✓	✓
		3				Profile torque = 0	✓	✓
		4				Reserved = 0	✓	✓
		5				Homing = 1	✓	✓
		6				Interpolated position = 1	✓	✓
		7				Cyclic synchronous position = 1	✓	✓
		8				Cyclic synchronous velocity = 1	✓	✓

Tab. 108 - DSP 402 objects

### 13.3.1. 0x603F, Error code

It is an unsigned 16 bits, read only objects, which contains the actual error of the device. It is zero if no error is present. Once a new error take place, an emergency message is transmitted and the code is logged into 0x1003, predefined error field.

Value	Description
0x0000	Drive ok
0x1000	Generic error
0x2300	Motor over-current
0x3210	DC link over-voltage
0x3220	DC link under-voltage
0x3230	Open motor cable
0x4310	Over-temperature
0x5943	Torque off error
0x6010	Watchdog
0x6320	Parameter error
0x7122	Motor error
0x7320	Encoder error
0x7321	Encoder fault
0x8100	Communication error
0x8130	NMT error
0x8611	Following error (step accumulation limit)

Tab. 109 - Error code values

### 13.3.2. 0x6040 / 0x6041, Control word and status word

0x6040, control word and 0x6041, status word are unsigned 16 bits objects to command motor movements and check the status. Ref. to 12- DSP402 – MOTION CONTROL for details.

### 13.3.3. 0x6060 / 0x6061, Modes of operation

0x6060, Mode of operation is a signed 8 bit object read write register, which can be used to change from one mode to another. 0x6061, Mode of operation display is a signed 8 bit read only object, which shows the actual mode in use. Ref. to 12.1.1 Modes of operation for details.

### 13.3.4. 0x6062 / 0x6063 / 0x6064 / 0x6065, Positions

0x6062, position demand value, 0x6063, position actual internal value and 0x6064 position actual value are a signed 32 bits read only objects. 0x6062 represents the ordered position, 0x6063 and 0x6064 are the position feedback from the encoder. All three values share the same resolution (step/rev).

0x6065, following error window is an unsigned 32 bits read write object. It can be configured to define the maximum allowable deviation between the ordered position and the feedback position. Similar to the Modbus step accumulation limit, its upper limit is constrained to 10 motor revolutions.

0x60F4, following error actual value, measures the difference between the ordered position and the feedback position. When the motor is stationary, this value should not exceed 1.8°, i.e. 1 / 200 revolution. In this a state, the drive supplies maximum current to the motor, allowing it to deliver maximum torque to the load. This condition persists until the following error window is exceeded or if the position deviation decreases.

During the motor movement, the position deviation corresponding to the maximum motor torque may vary based on factors such as speed, load inertia, and motor winding parameters.

#### Notes:

*In Modbus the current position is read/write register. In CANopen/EtherCAT it is read only. In order to force position demand value to a new value, it is necessary to use homing mode: by writing on 0x6060, modes of operation = 6 and 0x607C, homing mode = 35. In this way it is possible to choose the new position value writing on 0x607C, home offset and commanding a start homing acting on 0x6040, control word. Position demand value will be set equal to home offset and position actual value will maintain the small difference, without any movement of the motor. Alternatively, you can access Modbus registers via CAN, and write directly on 0x2005: 0x06, Modbus current position*

*When controlling FD in interpolated position (IP) or cyclic synchronous position (CSP) modes the master shall initialize its position counter with the object 0x6062, position demand value.*

### 13.3.5. 0x606B / 0x606C, Velocities

0x606B and 0x606C are signed 32 bits read only objects.

0x606B, velocity demand value represents the actual speed set-point; during acceleration and deceleration its value increases and decreases following the selected ramp profile.

0x606C is the velocity actual value, which is calculated from the encoder using a 20 ms filter (at low speeds the measurement can be noisy).

#### Note:

*Since 0x606C, velocity actual value is measured directly from the encoder, it can be used to observe the maximum allowed speed of a specific movement.*

### 13.3.6. 0x6073 / 0x6078, Motor current

0x6073, Maximum current is an unsigned 16 bits object, which represents the maximum peak current that the drive supplies to the motor. Normally the rated current on the datasheet of a stepper motor is expressed as bi-polar, both phases on, i.e. the current supplied by full-step drives, where both phases are energized with the same current and by inverting one at a time their sign, the motor spins. Such rated current is established in the conditions of motor not running and both phases energized at rated current, so that after a certain period of time the motor will not exceed the maximum temperature.

FD stepper drives, instead, work in sinusoidal micro-step, hence the maximum current parameter represents the peak of the sine wave of the motor current. In order to be compared with the motor rated current, its r.m.s. ( $\sqrt{2}$  of the peak) shall be used.

When the motor is not running, the main power dissipation is given by the Joule effect,  $RI^2$ . When rotating at high speeds fast changing

of magnetic fields generates Foucault's currents, which sum up to increase the thermal power dissipation. In order to avoid motor over-heating, it is also important to evaluate the duty cycle of the motor (continuous, intermittent, etc.).

Since FD drives exploits the integrated magnetic encoder to detect the torque applied to the motor, they are able to decrease the current when not needed, in order to let it cool down. The current setting ranges from 0x6073, Maximum current to 0x2005: 0xF6, Minimum current, depending on the torque applied.

0x0678, current actual value represents a filtered measurement of the milli-amp that flows into the motor.

### 13.3.7. 0x6079, DC link circuit voltage

0x6079, DC link circuit voltage is a unsigned 32 bit object that represents the voltage on the DC power supply. It is represented as  $10^{-2}$  V.

### 13.3.8. 0x607A, Target position

0x607A, target position is a signed 32 bits read write object. Its default value is initialized to Cycle 0 position.

It is used in profile position mode with the meaning of:

- position destination in absolute sub-mode,
- movement distance in relative sub-mode,
- maximum movement distance in delta-stop sub-mode,
- position delay and time delay in delay sub-modes.

In cyclic synchronous position mode, it represents the actual position set-point that is interpolated by the drive.

### 13.3.9. 0x607C, Home offset

0x607C, home offset is a signed 32 bits read write object, initialized to homing position register. It represents the value of position counter at the end of homing mode movement.

### 13.3.10. 0x607D, Software position limits

*Software position limit* contains the sub-indexes *min position limit* and *max position limit*. These parameters define the absolute limits for demand position counter.

Writing one sub-index cause its activation. They are not active in homing, interpolated position cyclic synchronous position, cyclic synchronous velocity modes and when the drive is used in step/dir or quadrature step modes.

### 13.3.11. 0x607F, Maximum profile velocity

It is an unsigned 32 bits read write object, which upper limit the register 0x6081, profile velocity. Its default value is equal to 300'000 step / s and it cannot be exceeded.

### 13.3.12. 0x6081, Profile velocity

It is an unsigned 32 bits read write object, initialized with cycle 0 speed. It represents the speed set-point of profile position modes.

### 13.3.13. 0x6083 / 0x6084 / 0x6086, Profile acceleration / deceleration / type

0x6083, profile acceleration and 0x6084, profile deceleration are unsigned 32 bits read write objects, and initialized with acceleration and deceleration Modbus registers. They represent the maximum speed increment / decrement every millisecond. Their value is upper limited to 20'000 kstep/s<sup>2</sup>.

0x6086, motion profile type is a signed 16 bits read write object, initialized to bits 7 and 8 of configuration register. Its value determines the speed ramp:

- 0: linear,

- 1: parabolic,
- 2: s-curve.

### 13.3.14. 0x6098 / 0x6099 / 0x609A, Homing method / speeds / acceleration

0x6098, homing method is a signed 32 bits read write object.

0x6099, homing speeds is an array of 3 elements:

- Sub-index 0: unsigned 8 bits read only, number of entries = 2
- Sub-Index 1: unsigned 32 bits read write, research speed
- Sub-index 2: unsigned 32 bits read write, release speed

0x609A, homing acceleration is an unsigned 32 bits read write object, initialized with acceleration register. Its value is upper limited to 20'000 kstep/s<sup>2</sup> and it used only during homing movements.

For a detailed description refer to ch. 0.

### 13.3.15. 0x60C0 / 0x60C1 / 0x60C2 / 0x60C4, Interpolation

0x60C0, interpolation sub-mode select is a signed 16 bits read only object, currently equal to 0: linear interpolation.

0x60C1, interpolation data record is a record of just two objects:

- Sub-index 0: unsigned 8 bits read only, number of entries = 1,
- Sub-index 1: signed 32 bits read write object is the interpolated position, which constitutes the entry point of the interpolated positions buffer.

0x60C2, interpolation time period is used to define the time distance between two synchronized interpolated positions.

- Sub-index 0: unsigned 8 bits read only, number of entries = 2,
- Sub-index 1: unsigned 8 bits read write, interpolation time units, initialized to 1,
- Sub-index 2: signed 8 bits read write, interpolation time index, initialized to -3.

The period is calculated as *interpolation time unit* · 10<sup>interpolation time index</sup> seconds.

0x60C4, interpolation data record is used to define the buffer of interpolated points:

- Sub-index 0: unsigned 8 bits read only, number of entries = 6,
- Sub-index 1: unsigned 32 bits read only, maximum buffer size = 32,
- Sub-index 2: unsigned 32 bits read only, actual buffer size,
- Sub-index 3: unsigned 8 bits read only, buffer organization = 0,
- Sub-index 4: unsigned 16 bits read only, buffer position = 0,
- Sub-index 5: unsigned 8 bits read only, size of data record = 1,
- Sub-index 6: unsigned 8 bits write only, write 0 to clear the buffer.

#### Notes:

*The object 0x60C1 is a record because CANopen standard offers the possibility to define the interpolated motion with a set of multiple parameters for each interpolation point, e.g. position, speed, motor current, etc. FD drives use instead just a single parameter, which is the position, but keep the record format defined from the standard.*

*EtherCAT simplifies this approach using cyclic synchronous position mode, where only the 0x607A, target position is transmitted.*

*Interpolation time period generally varies in the range 1 to 10 milliseconds.*

### 13.3.16. 0x60FD, Digital inputs

It is unsigned 32 bits read only object, whose bits assume the meaning of below table:

Bit number	FD1E	FD1.xEC	FD2E	FD2.xC
0	Hardware limit switch down active			
1	Hardware limit switch up active			
2	Homing sensor active			
...	...			
16	IN1	IN1	IN1	IN1
17	IN2	IN2	IN2	IN2
18		IN3	IN3	IN3
19		IN4	IN4	IN4
20			IN5	
21			IN6	
22			IN7	
23				
24	OUT1	OUT1	OUT1	OUT1
25		OUT2	OUT2	OUT2
26			OUT3	

Tab. 110 - 0x60FD, Digital inputs

### 13.3.17. 0x60FE, Digital outputs

Digital outputs is a record used to force the outputs or to read their status.

- Sub-index 0: unsigned 8 bits read only, number of entries = 2,
- Sub-index 1: unsigned 32 bits read write, physical outputs. When read it represents the status of the output, when written, if the corresponding bit on the mask is 1, it forces open or close the output.
- Sub-index 2: unsigned 32 bits read write, bit mask. When its bits are written, it enables (1) or disables (0) the forcing of physical outputs.

### 13.3.18. 0x60FF, Target velocity

0x60FF, target velocity is signed 32 bits read write object, initialized to cycle 0 speed, which is the profile velocity and cyclic synchronous velocity modes speed set-point. Positive values create movements towards increasing position, negative values towards decreasing positions. It is upper limited to 300'000 step/s and lower limited to -300'000 step/s.

Writing the object during profile velocity mode determines speed ramp up or down to new speed set-point using profile acceleration and deceleration.

Writing the object during cyclic synchronous velocity mode immediately takes the speed setpoint, hence the master shall gradually increase the value during acceleration and decrease it during decelerations.



## 14. CYCLES AND SEQUENCES

FD drives can be programmed with up to 32 user defined cycles. Every cycle is identified by five parameters: type, speed, position, direction and delta-stop  $\mu$ steps. Acceleration and deceleration are common for all the cycles. Types are:

- 1: Jog,
- 2: Positioning relative,
- 3: Homing,
- 4: Delta-stop,
- 5: Positioning absolute,
- 7: Time delay.

Cycles can be combined in sequences and executed in stream, one after the other. Each sequence is described by 20 parameters, each of them identifies a cycle number or a command (Stop or Loop). A total of 10 sequences made of 20 parameters each are available.

Cycles and sequences can be selected, started and stopped using dedicated Modbus registers, digital inputs or EtherCAT objects.

When a sequence is selected, the start command launches the cycle identified by the first sequence parameter. As soon as this cycle finishes, the motor performs the cycle identified by the second parameter and so on.

When a Stop parameter (254) is met in the sequence, the motor decelerates till stopping; when it is a Loop parameter (255) the sequence restarts from the first cycle.

When neither Loop nor Stop are present, as the drive finishes one sequence of 20 cycles, it starts to perform the next sequence. This means a maximum loop sequence of 200 cycles or, if the last parameter in the last sequence is a Stop, a single sequence made of 199 cycles.

Every sequence is interruptible via stop command. In this case the sequence pointer is reset and a future start runs the first cycle.

Cycles and sequences parameters can be modified in RAM.

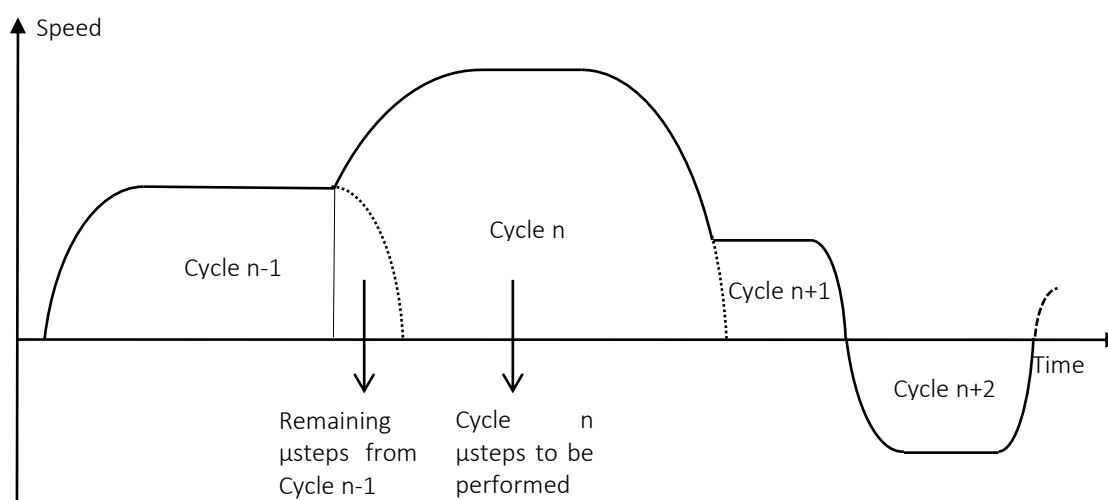


Fig. 23 - Sequence of cycles with parabolic ramps

Note:

The single cycle is considered finished when the remaining  $\mu$ steps are equal to the deceleration ramp area, in this condition the next cycle starts, performing its own  $\mu$ steps and the  $\mu$ steps left from the previous cycle, as shown in Fig. 6 motor speed ramps up or down to the new speed set-point.

## 14.1. Cycle and sequence selection

When no input is used as cycle or sequence selection, cycles and sequences can be selected using SEL\_CYC\_SEQ register:

**Select Cycle [0 – 31] → SEL\_CYC\_SEQ from 0 to 31**  
**Select Sequence [0 – 9] → SEL\_CYC\_SEQ from 32 to 41**

Otherwise, when the selection is made using digital inputs, the meaning associated to inputs combination depends on CONFIG register bit 13, 32\_CYCLES.

CONFIG register bit 13 = 0 → DI selects 10 Sequence and 22 Cycles

CONFIG register bit 13 = 1 → DI selects 32 Cycles

IN2, IN3, IN4, IN5, IN6 can be used for cycle or sequence selection. Inputs don't have a pre-assigned bit weight, the weight is ordered with the input number (weight zero is assigned to the lowest input number), e.g. if only IN3 and IN5 are used for sequence selection (CONFIG register bit 13 equal to zero), only sequences 0, 1, 2, 3 can be selected. The same sequences can be selected using only IN2 and IN4.

Using all the four inputs and CONFIG register bit 13 equal to zero, it is possible to select sequences from 0 to 9 and the remaining bits configurations from 10 to 15 are used to select single cycles from 10 to 15.

Note:

When the cycle or sequence selection is made using DI, writing SEL\_CYC\_SEQ register has no effect, as the selected value will be immediately overwritten by DI configuration.

To configure an input as cycle or sequence selection the register IN\_x\_CNF shall be set to 11. To do so it is possible to program such configuration in flash using IAP or write it via fieldbus in RAM.

## 14.2. Jog

When the jog cycle is started the motor accelerates at the selected speed and direction.

During jog cycle the STATUS\_WORD register bit 3 MODE\_JOG is set.

For all the other types of cycles the start command samples the cycle or sequence selection and the selection is not anymore read till the end of the movement. For Jog cycles, instead, it is possible to switch between them just by changing the selection, without passing through motor stop. An example using FD2E follows:

CONFIG register, bit 13      1: Select 32 cycles  
 IN1 configuration      1: Start NO,  
 IN2 configuration      1: Stop NO,  
 IN3 configuration      11: Cycle or sequence selection,  
 IN4 configuration      11: Cycle or sequence selection,

Cycle 0, Type              0: Jog,  
 Cycle 0, Speed            1'000  $\mu$ step/s  
 Cycle 0, Direction        0: CW, increasing positions

Cycle 2, Type              0: Jog,  
 Cycle 2, Speed            3'000  $\mu$ step/s  
 Cycle 2, Direction        0: CW, increasing positions

Cycle 1, Type              0: Jog,  
 Cycle 1, Speed            1'000  $\mu$ step/s  
 Cycle 1, Direction        1: CCW, decreasing positions

Cycle 3, Type              0: Jog,  
 Cycle 3, Speed            3'000  $\mu$ step/s  
 Cycle 3, Direction        1: CCW, decreasing positions

In this example the selection addresses cycles and IN3 has weight bit 0 and IN4 has weight bit 1. Jog cycles 0, 1, 2 and 3 have been configured in order to have IN4 direction, IN5 low and high speed:

IN3 = 0              CW, increasing positions  
 IN3 = 1              CCW, decreasing positions  
 IN4 = 0              Low speed 1'000  $\mu$ step/s  
 IN4 = 1              High 3'000  $\mu$ step/s

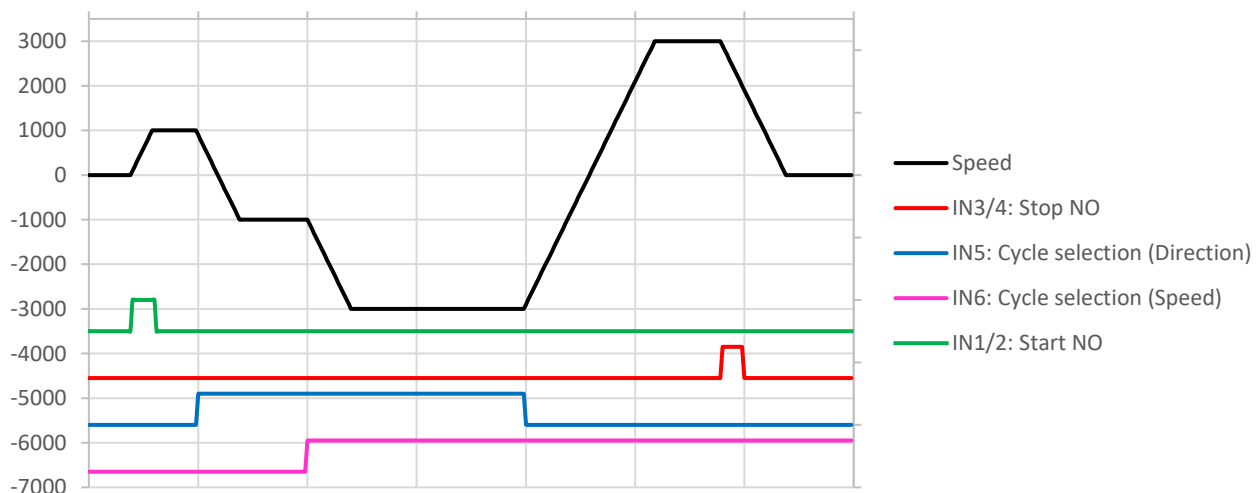


Fig. 24 - Jog

Note:

When a cycle selection involves a change of direction, the motor speed ramps down to zero and then ramp up to the new speed set-point in the opposite direction. If the selection changes again during the deceleration to another Jog at the original direction, anyhow the motor will decelerate to zero speed and then it reaccelerates at the last selected direction.

### 14.3. Positioning

Positioning cycles are motor movements to a precise position destination. The destination can be relative, expressed as an increment or decrement of the current position, or absolute, expressed as the final position of the movement.

Positioning relative cycles are defined by speed, position and direction. Position is an unsigned 32 bits register and direction can be:

- 0: CW towards increasing positions,
- 1: CCW towards decreasing positions.

Positioning absolute cycles are defined by speed and position. Position is a signed 32 bits register.

When the movement is completed, the target position is compared with the current position and, if the comparison is positive (target position reached), bit 6 TARG\_POS of the STATUS\_WORD is set.

Note:

When the single input is configured as Start NO / Stop NC for positioning relative movements, make sure that the signal does not bounce to avoid improper motor movements during signal bouncing. In any case it is recommended to split Start and Stop signals in two different inputs or to use positioning absolute cycles.

Sequences made of several positioning cycles can be used to configure custom made speed profiles.

In the transition between two positioning cycles of a sequence configured in the same direction, the speed increases or decreases to reach the new speed set-point, without passing through zero speed. If the zero-speed crossing is wanted, just interpose a third cycle in between at the opposite direction with zero  $\mu$ steps.

## 14.4. Homing

FD drives support several methods of homing cycles. Methods are selected via HOMING register. They can be:

- 0: Index
- 1: Homing simple
- 2: Homing, time stop, inversion
- 3: Homing, time stop, no inversion
- 4: Homing, time stop, inversion, index
- 5: Homing, time stop, no inversion, index
- 6: Mechanical stop
- 7: Mechanical stop, index
- 8: Current position

HOME\_POSITION will be the new position loaded in the CURR\_POSITION register at the end of homing cycle.

During homing cycles the STATUS\_WORD bit 8 HOMING is set and bit 11 AX\_CALIBRATED will be reset.

If the homing succeeds, at the end of the cycle bit 11 AX\_CALIBRATED will be set. AX\_CALIBRATED will remain set until no alarm arises or no homing cycle is launched again.

IN3 and IN4 can be set as homing inputs or hardware limit switch inputs. Homing is possible on both configurations.

FD drives can restore the multi-turn axis calibration during power-on using the integrated absolute encoder.

### 14.4.1. Index

When the HOMING register is equal to zero, which corresponds to an index cycle, the axis's calibration consists in a motor rotation to the absolute encoder position destination. The motor will rotate at V\_RELEASE speed.

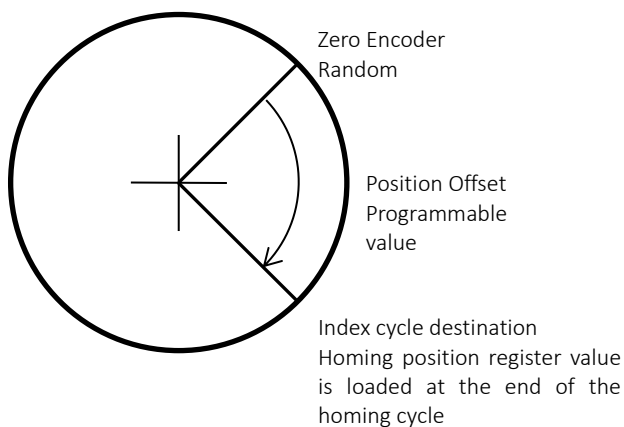


Fig. 25 - Index cycle

The cycle direction can be:

- 0: CW,
- 1: CCW,
- 2: Shortest distance.

POSITION\_OFFSET register modifies the absolute encoder position destination. When POSITION\_OFFSET is a zero value, the index cycle will stop at the zero-encoder position. The zero-encoder position is a random position, which depends upon the encoder mounting.

Customer can request to have zero encoder programmed in a particular position. In this case all the drives will be delivered with such precise encoder mounting.

### 14.4.2. Homing simple

When the HOMING register is equal to one, which corresponds to homing simple cycle, the homing consists in a motor rotation at V\_RESEARCH speed.

The cycle direction can be:

- 0: CW,
- 1: CCW.

As soon as the homing or the hardware limit switches inputs get active the motor starts to decelerate and the homing is concluded.

### 14.4.3. Homing, time stop and inversion / no-inversion

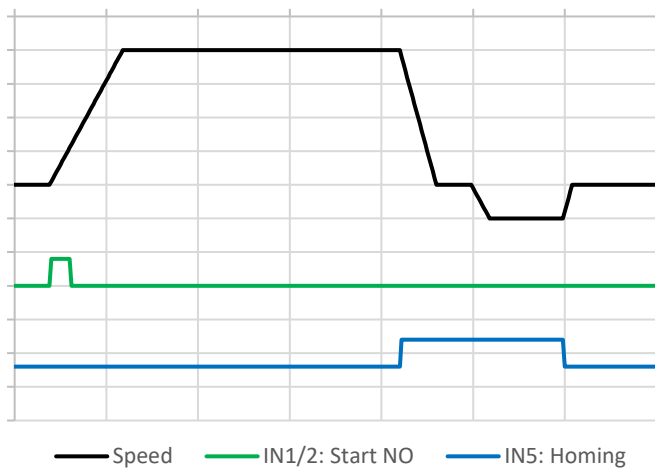


Fig. 26 - Homing invert

When the HOMING register is equal to two, which corresponds to homing, time stop and inversion cycle, the homing consists in a motor rotation at V\_RESEARCH speed.

The cycle direction can be:

0: CW,  
1: CCW.

As soon as the homing or the hardware limit switches inputs get active the motor starts to decelerates, it waits TIME STOP milliseconds and then it inverts the direction running at V\_RELEASE speed. The motor starts the deceleration to zero speed when the switch input gets inactive.

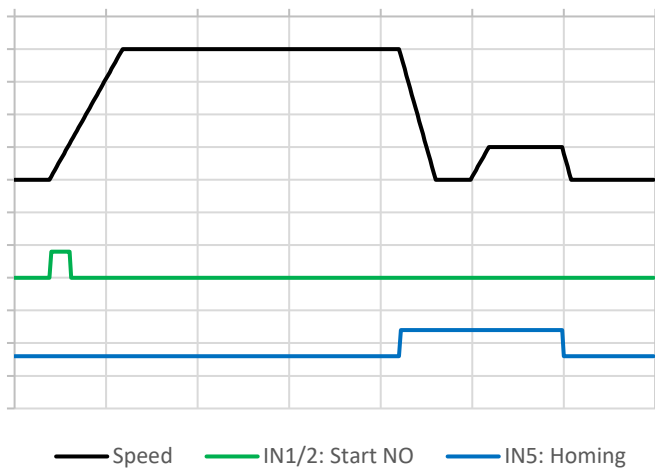


Fig. 27 - Homing no invert

If homing, time stop and no-inversion is selected (HOMING register equal to three) the motor, after waiting time stop, will accelerate to the same direction till the switch gets inactive.

### 14.4.4. Homing, time stop, inversion / no-inversion and index

When the HOMING register is equal to four, which corresponds to homing, time stop, inversion and index cycle, the homing consists in a motor rotation at V\_RESEARCH speed.

The cycle direction can be:

0: CW,  
1: CCW.

As soon as the homing or the hardware limit switches inputs get active the motor will start the deceleration, it will wait TIME STOP milliseconds and then it will invert the direction running at V\_RELEASE speed.

When the switch input gets inactive the motor will perform a index cycle to the absolute encoder position destination defined in POSITION\_OFFSET register.

Homing, time stop, no-inversion and index is selected with HOMING register equal to five.

*Note:*

*Absolute encoder position destination needs to be set approximately half motor revolution far from the deactivation of the switch to avoid the possible error of one revolution.*

#### 14.4.5. Mechanical stop

When the HOMING register is equal to six, which corresponds to mechanical stop cycle, the homing consists in a motor rotation at V\_RESEARCH speed.

When the position deviation is higher than 1920  $\mu$ steps, it means that the motor has found an obstacle and the homing cycle is concluded.

When the HOMING register is equal to seven, which corresponds to mechanical stop cycle + index, the mechanical stop cycle is followed by an index cycle in opposite direction.

#### 14.4.6. Current position

When the HOMING register is equal to eight, which corresponds to home at current position, no movement takes place, but the current position counter is updated at HOME\_POSITION value.

### 14.5. Delta stop

Delta stop is a particular positioning relative cycle, used when it is needed to stop the motor in a very accurate position after the activation of a sensor. A normal stop would decelerate the motor till zero speed without controlling the destination position, which would depend upon regime speed and deceleration. Delta stop cycles, instead, executes a programmed number of steps after delta stop input activation. IN3 and IN4 can be configured as delta stop inputs.

It is defined by speed, position, direction and delta stop steps.

When delta stop input gets active the motor executes exactly the steps identified by delta stop parameter and STATUS\_WORD bit 20 DELTA\_STOP\_OK is set. Delta stop input triggers a dedicated microprocessor interrupt routine to ensure fast position sampling.

If delta stop input does not get active during the movement or it gets active only during deceleration, the cycle terminates after the positioning relative steps configured in position parameter have been made.

To detect this event, it is possible to:

- configure multipurpose output (FD1E OUT1, FD2E OUT2) as RUNNING + /DSTOP, which would maintain the output in high state if no sensor activation is detected (timeout logic to be implemented on master side);
- evaluate cycle counter CNT\_CYC and delta stop counter CNT\_DSTOP registers to detect if their increment at the end of the cycle, i.e. when TARGET\_POS, bit 6 of STATUS\_WORD register, is set.
- Read status word delta stop ok bit.

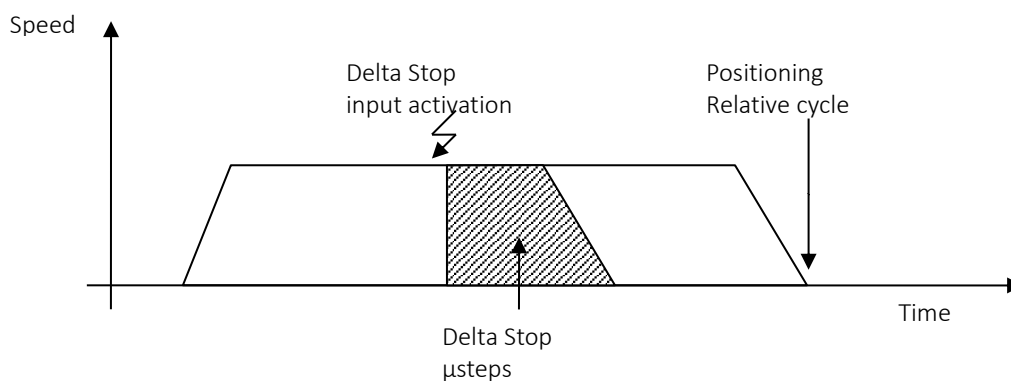


Fig. 28 - Delta Stop Cycles

Note:

The position of sensor activation shall be anticipated in respect to the wanted destination position of the delta stop steps.

To avoid too steep deceleration, make sure to set Delta Stop steps parameter higher than the deceleration steps at the maximum speed used.

Being a fast input, it is needed to use shielded cables and all the relevant measures to avoid disturbances.

Delta stop cycles can also be used inside the sequences instead of relative positioning cycles in order to jump from the current cycle to the next one activating Delta Stop input.

## 14.6. Time delay

When it is needed to configure a simple time delay in a sequence between two cycles, it is possible to use time delay cycles.

Position parameter will express the milliseconds of motor stop between previous and next cycle.

Note:

During delay, even if motor speed is zero, output running will remain active.



## 15. COMMANDS

To reduce the overhead of messages transmitted, the start command and cycle selection has been put together in a single write command on EXE\_FUN register. This command selects Cycle 0, modifies its parameters accordingly and then starts the cycle or performs below detailed functions. Only single cycles, not sequences can be started.

### 15.1. Rotate Right (ROR), Rotate Left (ROL)

EXE\_FUN = 1, 2  
or EXE\_FUN = 31, 32 with no reset

The rotate right command launches a jog movement in clockwise direction (DIR = "0") at the selected speed, increasing the position counter value. When the direction is inverted (bit 6 of CONFIG register) the motor will rotate in counterclockwise direction always increasing the position counter value.

The rotate left command launches a JOG movement in counter-clockwise direction (DIR = "1") at the selected speed, decreasing the position counter value. When the direction is inverted (bit 6 of CONFIG register) the motor will rotate in clockwise direction decreasing the position counter value.

When the motor is running it is possible to modify speed and direction giving ROL/ROR commands. Giving ROL after a previous ROR command and vice versa produce a motor deceleration and re-acceleration to the opposite direction.

When the drive is in alarm, ROR and ROL commands perform a system reset. As a consequence of this type of reset, the drive will be re-initialized at the flash programmed parameters (FD1 short circuit alarm can be reset only by powering off).

In case this type of reset is not wanted, ROR\_NO\_RST (EXE\_FUN = 31) and ROL\_NO\_RST (EXE\_FUN = 32) can be used instead.

During jog movements the bit 3 MODE\_JOG of the STATUS\_WORD register is set.

### 15.2. Motor Stop (MST)

EXE\_FUN = 3

The stop command launches a motor deceleration to zero speed.

If a homing cycle is stopped the bit 11 AX\_CALIBRATED of the STATUS\_WORD is reset.

### 15.3. Move to Position (MVP)

EXE\_FUN = 10, 11

This command launches a positioning relative cycle. When the DwLoader ABS check box is selected an absolute movement is performed and the POSITION parameter corresponds to destination position. When deselected, the motor will perform a relative movement of the total amount of µsteps defined in POSITION parameter.

In Modbus protocol two commands are used:

- MVP\_ABS: Exe\_Fun = 10 for absolute positioning cycles,
- MVP\_REL: Exe\_Fun = 11 for relative positioning cycles.

Starting at the position 10'000, a command "MVP\_REL,-1'000" produces a motor rotation till position 9'000.

Starting at the position 10'000, a command "MVP\_ABS,-1'000" produces a motor rotation till position -1'000.

After an MVP\_ABS command Cycle 0 position and direction are overwritten according to a relative positioning cycle.

This command can be executed only when the motor is stopped. The cycle uses acceleration and deceleration configured in general data, speed and delta position configured in Cycle 0 data.

When the movement is completed, the target position is compared with the current position and, if the comparison is positive (target position reached), bit 6 TARG\_POS of the STATUS\_WORD is set.

### 15.4. Reference Search (RFS)

EXE\_FUN = 13

This command launches the homing cycle. Type of homing shall be selected using HOMING register.

### 15.5. Reset (RST)

EXE\_FUN = 14

This command performs a System Reset. As a consequence of the reset all the RAM data will return to their Flash programmed values and the alarms will be reset.

When no alarm is present the current position, the bit 11 AX\_CALIBRATED and the bit 6 TARG\_POS will be reloaded after reset.

### 15.6. Alarm Reset (ALR)

EXE\_FUN = 15

Alarm Reset command performs the software alarm reset.

When the Step Accumulation alarm is reset, the drive is able to restore the multi-turn position. If the axis was calibrated before that alarm occurred, the STATUS\_WORD bit 11 AX\_CALIBRATED will be restored. Ref. to 7 ALARMS.

Note:

FD1 short circuit alarm cannot be reset via software, nor with RST, nor with ALR.

### 15.7. Disable/Enable Motor Current (DMC/EMC)

EXE\_FUN = 16, 17

Disable/Enable Motor Current commands can be used to free the motor and move it to a desired position. During current disable the STATUS WORD bit 16 DIS\_CURR is set.

When the current is enabled the system is able to calculate the correct position and currents configuration taking into account the movement performed during disable. STATUS\_WORD bit 11 AX\_CALIBRATED will be restored.

If an alarm is present the commands DMC followed by EMC perform an alarm reset.

If an input is configured as 0: Disable Current, its status will prevail over EXE\_FUN commands.

### 15.8. Disable/Enable Frequency (DFR/EFR)

EXE\_FUN = 18, 19

Disable Frequency command can be used to deselect certain drives when using a broadcast start command or when distributing the same Step Frequency Input to more drives. Motor movements are inhibited, but the current and hence the holding torque are still present.

During frequency disable the STATUS WORD bit 17 DIS\_FREQ is set.

If an input is configured as 1: Disable Frequency, its status will prevail over EXE\_FUN commands.

## 15.9. Enter Programming Mode (PRG)

EXE\_FUN = 20

Enter Programming Mode command is used when it is necessary to re-program the firmware of the drive. FD will automatically reset. The pointer of the main program will jump to a dedicated flash memory block, where firmware re-programming code is installed.

In programming mode bit 18 of status word register is set. All the movements and motor current are disabled. Using Write File Record (21) it is possible to reprogram the firmware and the user data.

Two files numbers are supported:

- FD Firmware = 1,
- User Data = 2.

It is not necessary to set the drive in programming mode when programming file 2, User Data.

The first Write File Record request must begin with Record Number equal to zero. The next requests must have a sequential record numbers (1, 2, 3, ...).

## 16. MODBUS

### 16.1. Registers addresses

Modbus protocol addresses word registers (16-bits). Data is 32-bits type and it is then organized into two 16 bits registers (low and high words).

The same data are also mapped inside CANopen/EtherCAT objects at indexes 0x2005 and 0x2006 and sub-indexes as per below table.

Register name	Modbus register address <sup>1</sup>		CANopen/EtherCAT Index: subindex [hex]
	H	L	
START	0	1	0x2005: 0x01
STOP	2	3	0x2005: 0x02
ACCELERATION	4	5	0x2005: 0x03
DECELERATION	6	7	0x2005: 0x04
CURR_SPEED	8	9	0x2005: 0x05
CURR_POSITION	10	11	0x2005: 0x06
CURR_CYCLE	12	13	0x2005: 0x07
HOME_POSITION	14	15	0x2005: 0x08
POSITION_OFFSET	16	17	0x2005: 0x09
SEL_CYC_SEQ	18	19	0x2005: 0x0A
TARGET_VERSION	20	21	0x2005: 0x0B
IO_BITS	22	23	0x2005: 0x0C
CONFIG	24	25	0x2005: 0x0D
MODBUS_ADDRESS	26	27	0x2005: 0x0E
EXE_FUN	28	29	0x2005: 0x0F
I_MAX	30	31	0x2005: 0x10
DATA_LINK_STATUS	32	33	0x2005: 0x11
ERR_FAT	34	35	0x2005: 0x12
MODBUS_BAUD_RATE	36	37	0x2005: 0x13
STATUS_WORD	38	39	0x2005: 0x14
CYC_0_TYPE	40	41	0x2005: 0x15
CYC_0_SPEED	42	43	0x2005: 0x16
CYC_0_DELTA_POS	44	45	0x2005: 0x17
CYC_0_DIRECTION	46	47	0x2005: 0x18
CYC_0_DELTA_STOP	48	49	0x2005: 0x19
CYC_n_TYPE	$2 \times (20 + 5 \times n)$	$2 \times (20 + 5 \times n) + 1$	0x2005: (0x1A + 5 × n)
CYC_n_SPEED	$2 \times (21 + 5 \times n)$	$2 \times (21 + 5 \times n) + 1$	0x2005: (0x1B + 5 × n)
CYC_n_DELTA_POS	$2 \times (22 + 5 \times n)$	$2 \times (22 + 5 \times n) + 1$	0x2005: (0x1C + 5 × n)
CYC_n_DIRECTION	$2 \times (23 + 5 \times n)$	$2 \times (23 + 5 \times n) + 1$	0x2005: (0x1D + 5 × n)
CYC_n_DELTA_STOP	$2 \times (24 + 5 \times n)$	$2 \times (24 + 5 \times n) + 1$	0x2005: (0x1E + 5 × n)
SEQ_0_DW_0	360	361	0x2005: 0xB5
SEQ_0_DW_1	362	363	0x2005: 0xB6
SEQ_0_DW_2	364	365	0x2005: 0xB7
SEQ_0_DW_3	366	367	0x2005: 0xB8
SEQ_0_DW_4	368	369	0x2005: 0xB9
SEQ_n_DW_0	$2 \times (180 + 5 \times n)$	$2 \times (180 + 5 \times n) + 1$	0x2005: (0xBA + 5 × n)
SEQ_n_DW_1	$2 \times (181 + 5 \times n)$	$2 \times (181 + 5 \times n) + 1$	0x2005: (0xBB + 5 × n)
SEQ_n_DW_2	$2 \times (182 + 5 \times n)$	$2 \times (182 + 5 \times n) + 1$	0x2005: (0xBC + 5 × n)
SEQ_n_DW_3	$2 \times (183 + 5 \times n)$	$2 \times (183 + 5 \times n) + 1$	0x2005: (0xBD + 5 × n)
SEQ_n_DW_4	$2 \times (184 + 5 \times n)$	$2 \times (184 + 5 \times n) + 1$	0x2005: (0xBE + 5 × n)
IN_3_CNF	460	461	0x2005: 0xE7
IN_4_CNF	462	463	0x2005: 0xE8
HOMING	464	465	0x2005: 0xE9
TS	466	467	0x2005: 0xEA
V_RESEARCH	468	469	0x2005: 0xEB
V_RELEASE	470	471	0x2005: 0xEC
POS_SW_LS_UP	472	473	0x2005: 0xED

<sup>1</sup> Register addresses of the Modbus holding registers are calculated as per following example:

006B hex = 107 , + 40001 offset = input #40108

Register name	Modbus register address <sup>1</sup>		CANopen/EtherCAT Index: subindex [hex]
	H	L	
POS_SW_LS_DW	474	475	0x2005: 0xEE
I_LIM	476	477	0x2005: 0xEF
KNUM_NOM	478	479	0x2005: 0xF0
TEMPERATURE	480	481	0x2005: 0xF1
ACCUMULATION_LIMIT	484	485	0x2005: 0xF3
ENC_REV	486	487	0x2005: 0xF4
ACCUMULATED_STEPS	488	489	0x2005: 0xF5
I_MIN	490	491	0x2005: 0xF6
ENC_POS	492	493	0x2005: 0xF7
ENC_SPEED	494	495	0x2005: 0xF8
ENC_LATCH_RIS	496	497	0x2005: 0xF9
PON_CALIB_LIM	498	499	0x2005: 0xFA
TIME_CONST	500	501	0x2005: 0xFB
START_STOP_FREQ	502	503	0x2005: 0xFC
TEMP_OFF	504	505	0x2005: 0xFD
IN_1_CNF	506	507	0x2005: 0xFE
IN_2_CNF	508	509	0x2005: 0xFF
OUT_1_CNF	510	511	0x2006: 0x01
OUT_2_CNF	512	513	0x2006: 0x02
RESOLUTION_NUM	514	515	0x2006: 0x03
RESOLUTION_DEN	516	517	0x2006: 0x04
I_MIS	518	519	0x2006: 0x05
STATION_ADDRESS	520	521	0x2006: 0x06
CAN_BAUD_RATE	522	523	0x2006: 0x07
CNT_CYC	524	525	0x2006: 0x08
CNT_DSTOP	526	527	0x2006: 0x09
ENC_LATCH_FAL	528	529	0x2006: 0x0A
ERR_LOG	530	531	0x2006: 0x0B
FRAME_PERIOD_AV	532	533	0x2006: 0x0C
FRAME_PERIOD_VAR	534	535	0x2006: 0x0D
IN_5_CNF	536	537	0x2006: 0x0E
...			
ENC_STATUS	540	541	0x2006: 0x10
IN_6_CNF	542	543	0x2006: 0x11
IN_VAL_A1/IN_7_CNF	544	545	0x2006: 0x12
OUT_3_CNF	546	547	0x2006: 0x13
MAIN_PERIOD	548	549	0x2006: 0x14
CAN_ERR_STATUS	550	551	0x2006: 0x15
V_DC	552	553	0x2006: 0x16
SM0_0	554	555	0x2006: 0x17
SM0_1	556	557	0x2006: 0x18
SM1_0	558	559	0x2006: 0x19
SM1_1	560	561	0x2006: 0x1A
SM2_0	562	563	0x2006: 0x1B
SM2_1	564	565	0x2006: 0x1C
SM3_0	566	567	0x2006: 0x1D
SM3_1	568	569	0x2006: 0x1E
AL_STATUS	570	571	0x2006: 0x1F
P0_P1_RX_ERROR_CNT	572	573	0x2006: 0x20
P2_P3_RX_ERROR_CNT	574	575	0x2006: 0x21
ESCREG_CMD_ADD	576	577	0x2006: 0x22
ESCREG_DATA_0	578	579	0x2006: 0x23
ESCREG_DATA_1	580	581	0x2006: 0x24
APPLICATION_VERSION	582	583	0x2006: 0x25
RPDO1_PARAM	584	585	0x2006: 0x26
RPDO1_MAP0	586	587	0x2006: 0x27
RPDO1_MAP1	588	589	0x2006: 0x28

Register name	Modbus register address <sup>1</sup>		CANopen/EtherCAT Index: subindex [hex]
	H	L	
RPDO1_MAP2	590	591	0x2006: 0x29
RPDO1_MAP3	592	593	0x2006: 0x2A
RPDO1_MAP4	594	595	0x2006: 0x2B
RPDO2_PARAM	596	597	0x2006: 0x2C
RPDO2_MAP0	598	599	0x2006: 0x2D
RPDO2_MAP1	600	601	0x2006: 0x2E
RPDO2_MAP2	602	603	0x2006: 0x2F
RPDO2_MAP3	604	605	0x2006: 0x30
RPDO2_MAP4	606	607	0x2006: 0x31
RPDO3_PARAM	608	609	0x2006: 0x32
RPDO3_MAP0	610	611	0x2006: 0x33
RPDO3_MAP1	612	613	0x2006: 0x34
RPDO3_MAP2	614	615	0x2006: 0x35
RPDO3_MAP3	616	617	0x2006: 0x36
RPDO3_MAP4	618	619	0x2006: 0x37
RPDO4_PARAM	620	621	0x2006: 0x38
RPDO4_MAP0	622	623	0x2006: 0x39
RPDO4_MAP1	624	625	0x2006: 0x3A
RPDO4_MAP2	626	627	0x2006: 0x3B
RPDO4_MAP3	628	629	0x2006: 0x3C
RPDO4_MAP4	630	631	0x2006: 0x3D
TPDO1_PARAM	632	633	0x2006: 0x3E
TPDO1_MAP0	634	635	0x2006: 0x3F
TPDO1_MAP1	636	637	0x2006: 0x40
TPDO1_MAP2	638	639	0x2006: 0x41
TPDO1_MAP3	640	641	0x2006: 0x42
TPDO1_MAP4	642	643	0x2006: 0x43
TPDO2_PARAM	644	645	0x2006: 0x44
TPDO2_MAP0	646	647	0x2006: 0x45
TPDO2_MAP1	648	649	0x2006: 0x46
TPDO2_MAP2	650	651	0x2006: 0x47
TPDO2_MAP3	652	653	0x2006: 0x48
TPDO2_MAP4	654	655	0x2006: 0x49
TPDO3_PARAM	656	657	0x2006: 0x4A
TPDO3_MAP0	658	659	0x2006: 0x4B
TPDO3_MAP1	660	661	0x2006: 0x4C
TPDO3_MAP2	662	663	0x2006: 0x4D
TPDO3_MAP3	664	665	0x2006: 0x4E
TPDO3_MAP4	666	667	0x2006: 0x4F
TPDO4_PARAM	668	669	0x2006: 0x50
TPDO4_MAP0	670	671	0x2006: 0x51
TPDO4_MAP1	672	673	0x2006: 0x52
TPDO4_MAP2	674	675	0x2006: 0x53
TPDO4_MAP3	676	677	0x2006: 0x54
TPDO4_MAP4	678	679	0x2006: 0x55
TPDO1_TIME	680	681	0x2006: 0x56
TPDO2_TIME	682	683	0x2006: 0x57
TPDO3_TIME	684	685	0x2006: 0x58
TPDO4_TIME	686	687	0x2006: 0x59
...	...	...	
CAN_LIFETIME	700	701	0x2006: 0x60
CAN_CONS_HEARTBEAT	702	703	0x2006: 0x61
CAN_PROD_HEARTBEAT	704	705	0x2006: 0x62
SHIFT_SM2_DC	706	707	0x2006: 0x63
SERIAL_NUMBER	708	709	0x2006: 0x64

Tab. 111 - Registers addresses

## 16.2. Registers Description

### 16.2.1. Start, Stop

START and STOP are two Read/Write registers normally equal to zero. Setting the START register, the motor will run using selected Cycle data. Setting the STOP register, the motor will decelerate till stopping.

Once START and STOP registers are processed, they are automatically reset by software.

If the START is set again during the running cycle, at the end of the movement the selected cycle starts. This is valid only for single cycles.

### 16.2.2. Acceleration, Deceleration

ACCELERATION and DECELERATION are two Read/Write registers. These values are expressed as thousands of  $\mu$ -steps per second square.

Calculation example using linear ramp:

Requested speed:  $v_{rpm} = 300$  [rpm]

Requested acceleration time:  $\Delta t = 50$  [ms]

$$v_{\mu\text{step/s}} = 300 \cdot \frac{12'800}{60} = 64'000 \left[ \frac{\text{step}}{\text{s}} \right],$$

$$a_{\mu\text{step/s}^2} = \frac{v_{\mu\text{step/s}}}{\Delta t} = \frac{64'000}{0.05} = 1'280'000 \left[ \frac{\text{step}}{\text{s}^2} \right],$$

$$\text{ACCELERATION} = \frac{a_{\mu\text{step/s}^2}}{1'000} = 1'280 \left[ \frac{1'000 \cdot \text{step}}{\text{s}^2} \right].$$

Calculation example using parabolic ramp:

Requested speed:  $v_{rpm} = 300$  [rpm]

Requested acceleration time:  $\Delta t = 50$  [ms]

$$v_{\mu\text{step/s}} = 300 \cdot \frac{12'800}{60} = 64'000 \left[ \frac{\text{step}}{\text{s}} \right],$$

$$a_{\mu\text{step/s}^2} = 2 \cdot \frac{v_{\mu\text{step/s}}}{\Delta t} = \frac{128'000}{0.05} = 2'560'000 \left[ \frac{\text{step}}{\text{s}^2} \right],$$

$$\text{ACCELERATION} = \frac{a_{\mu\text{step/s}^2}}{1'000} = 2'560 \left[ \frac{1'000 \cdot \text{step}}{\text{s}^2} \right].$$

### 16.2.3. Current Speed, Position, Cycle

CURR\_SPEED and CURR\_CYCLE are Read Only registers, while CURR\_POSITION is a Read/Write register. These registers are updated every millisecond.

During sequences CURR\_CYCLE indicates the cycle number that is currently running in the sequence.

### 16.2.4. Homing Position, Position Offset

HOME\_POSITION and POSITION\_OFFSET are two Read/Write registers used in homing cycles.

### 16.2.5. Select Cycle Sequence

SEL\_CYC\_SEQ is a Read/Write register. It has to be set from 0 to 31 to select the single cycle 0-31 and from 32 to 41 to select the sequence 0-9. Ref. to 0



Cycle and sequence selection.

### 16.2.6. Target Version

TARGET\_VERSION is a Read Only register, which indicates in the low word the Vx.xx firmware version and in the high word the hardware FDx.xx version.

### 16.2.7. I/O Bits

IO\_BITS is a Read Only register used to read the status of the I/O port.

Bit number	I/O	Description
0	IN1	Status of the input/output pins
1	IN2	
2	IN3	
3	IN4	
4	IN5	
5	IN6	
6	IN7	
7	IN8	
8	OUT1	
9	OUT2	
10	OUT3	

Tab. 112 - I/O bits register

### 16.2.8. Configuration

CONFIG is a Read / Write register used to configure driver features.

Bit number	Name	Description
1	AUTO_FULL_STEP	Enabled, it increases motor current rms at medium speeds.
2	ENCODER_ENABLE	It enables encoder functionalities.
4	SW_LS_UP_ENABLE	Software limit switch at increasing positions.
5	SW_LS_DW_ENABLE	Software limit switch at decreasing positions.
6	INVERT_DIR	0: normal direction: e.g. ROL = counter-clockwise towards decreasing positions, 1: inverted direction: e.g. ROL = clockwise towards decreasing positions.
7-8	RAMP	0: linear, 1: parabolic, 2: s-curve.
9	START_STOP_FREQ	0: automatic, 1: start stop frequency enabled.
10	POS_UPLOAD	Ref. to 9 POWER-ON POSITION RESTORE (QUASI-ABSOLUTE MULTI-TURN).
11	EN_BOOST	It increases the motor current during movement starts
12	DIS_TORQUE_CONTROL	0: Torque control and step accumulation enabled 1: Torque control and step accumulation disabled. Just step loss detection and current reduction are enabled.
13	SELECT_32_CYC	0: Inputs select 10 sequences 1: Inputs select 32 cycles

Tab. 113 - Configuration register

### 16.2.9. Modbus Address

MODBUS\_ADDRESS is a Read/Write register used to set the slave address into Modbus network, when all DIP switches are off.

FD drives address is selected via DIP switches. When they are all off, since the zero address is used for broadcast messages, the address is taken from MODBUS\_ADDRESS parameter.

### 16.2.10. Execute Function

EXE\_FUN is a Read / Write register to launch cycle 0 commands. Write following values to activate the specific function.

Value	Function	Description
1	ROR	Rotate towards increasing positions
2	ROL	Rotate towards decreasing positions
3	MST	Motor stop
10	MVP_ABS	Move versus position absolute
11	MVP_REL	Move versus position relative
13	RFS	Reference search (homing)
14	RST	Reset
15	ALR	Alarm reset
16	DMC	Disable motor current
17	EMC	Enable motor current
18	DFR	Disable frequency
19	EFR	Enable frequency
20	PRG	Programming mode
31	ROR_NO_RST	Rotate towards increasing positions – w/o system reset
32	ROL_NO_RST	Rotate towards decreasing positions – w/o system reset

Tab. 114 - Execute function register

### 16.2.11. Maximum, Minimum and Limit Currents

I\_MAX is a Read/Write register used to set the maximum sine wave peak current per phase expressed in milliamps. A protection has been implemented in order to prevent damages to the drive and motor in case of improper settings. I\_MAX value is upper limited to I\_LIM, a Read Only register, which cannot be modified.

I\_MIN is a Read/Write register used to set the minimum sine wave peak current per phase. I\_MIN represents the amount of current that pass through the windings if no torque is applied. A low I\_MIN value reduces power dissipation and temperature.

I\_MIS is a Read Only register, which express the actual motor current sine wave peak in mA.

### 16.2.12. Data Link status

DATA\_LINK\_STATUS is a Read Only register to monitor EtherCAT slave controller data link status.

Bit number	Name	Description
0	PDI/EEPROM	1: EEPROM loaded correctly, PDI operational
1	PDI watchdog status	1: PDI watchdog reloaded
2	Enhanced Link Detection	1: Activated for at least one port
3	reserved	
4	Physical Link on Port 0	1: Link detected
5	Physical Link on Port 1	1: Link detected
6	Physical Link on Port 2	1: Link detected
7	reserved	
8	Loop Port 0	1: Closed.
9	Communication on Port 0	1: Communication established
10	Loop Port 1	1: Closed.
11	Communication on Port 1	1: Communication established
12	Loop Port 2	1: Closed.
13	Communication on Port 2	1: Communication established

Tab. 115 – Data link status

### 16.2.13. Fatal Error, Error Log

ERR\_FAT is a Read Only register used to read the status of the actual active alarm. Ref. to 7 ALARMS.

ERR\_LOG is a Read/Write register used to read the previous 4 alarm. Byte 0 has the information of the most recent alarm, Byte 4 has the information of the oldest alarm.

### 16.2.14. Modbus Baud Rate

MODBUS\_BAUD\_RATE is a Read/Write register used to configure the RS-232 and RS-485 baud rates. It ranges from 4'800 up to 115'200 bps for RS-232 and from 4'800 up to 921'600 bps for RS-485.

### 16.2.15. Status Word

STATUS\_WORD is a Read Only register which shows the driver operating conditions.

Bit number	Name	Description
0	DRIVER_READY_0	Drive initialization succeeded.
1	DRIVER_READY_1	Current and frequency are enabled.
2	ALARM_ACTIVE	Ref. to 7 ALARMS
3	MODE_JOG	The drive is executing a jog cycle.
4	TORQUE_LIMIT	The drive is following the position actual value.
...		
6	TARGET_POS	The motor is arrived at the correct destination position.
7	CYCLE_RUNNING	The motor is running a cycle, including delay cycles
8	HOMING	The motor is performing a homing cycle.
...		
10	POS_UPLOADED	Exact power-off position have been successfully uploaded during power-on.
11	AX_CALIBRATED	The axis is calibrated. This bit is set after a homing cycle.
12	SW_LS_UP_ACTIVE	Software Limit Switch Up is reached. The motor is not able to execute increasing position movements.
13	SW_LS_DW_ACTIVE	Software Limit Switch Down is reached. The motor is not able to execute decreasing position movements.
14	HW_LS_UP_ACTIVE	Hardware Limit Switch Up is reached. The motor is not able to execute increasing position movement.
15	HW_LS_DW_ACTIVE	Hardware Limit Switch Down is reached. The motor is not able to execute decreasing position movement.
16	DIS_CURR	Current is disabled.
17	DIS_FREQ	Frequency is disabled.
18	PRG_MODE	Drive is in programming mode.
19	POWER_ON_POS_OK	The deviation of the power on absolute position is inside the POWER_ON_CALIBRATION_LIMIT register. The multi-turn position, AX_CALIBRATED and the current configuration is restored.
20	DELTA_STOP_OK	Delta stop gets active during DS cycle.
21	IX_EARLY	Marker cycle ended before 1/8 of revolution.
22	IX_LATE	Marker cycle ended after 7/8 of revolution

Tab. 116 - Status word register

### 16.2.16. Cycles Data

Cycle registers are Read/Write registers which are used to define the motion parameters to be selected and started/stopped using registers or digital inputs.

Values	Registers	Description
[1 – 7]	CYC_n_TYPE	1: Jog, 2: Positioning relative, 3: Homing, 4: Delta stop, 5: Positioning absolute, 7: Time delay
[1 – 300 000]	CYC_n_SPEED	Speed expressed in $\mu$ step per second
[0 – $2^{32}-1$ ] or [- $2^{31}$ – $2^{31}-1$ ]	CYC_n_DELTA_POS	Jog: not used Positioning relative: steps to be executed (unsigned 32 bits) Homing: not used Delta stop: maximum steps to be executed (unsigned 32 bits). Positioning absolute: destination position (signed 32 bits). Time delay: milliseconds delayed.
[0, 1]	CYC_n_DIRECTION	When bit 6 INVERT_DIR of CONFIG register is zero: 0: CW rotation towards increasing positions 1: CCW rotation towards decreasing positions
[0 – $2^{32}-1$ ]	CYC_n_DELTA_STOP	Delta stop: steps executed after delta stop input activation (unsigned 32 bits)

Tab. 117 - Cycle n register

### 16.2.17. Sequence

Sequence registers are Read/Write registers which can be used to define 10 sequences made of up to 20 cycles or commands (Stop / Loop) each. Each sequence is described by 5 Double Words (32 bits) sequence parameters. The ordered sequence of cycles and commands are addressed in Bytes inside the 5 DW's. The lowest addressable Byte into the sequence identifies the first cycle to be executed.

Values	Name	Description
[0-255][0-255][0-255][0-255]	SEQ_n_DW_0	Sequence parameter 0, 1, 2, 3
[0-255][0-255][0-255][0-255]	SEQ_n_DW_1	Sequence parameter 4, 5, 6, 7
[0-255][0-255][0-255][0-255]	SEQ_n_DW_2	Sequence parameter 8, 9, 10, 11
[0-255][0-255][0-255][0-255]	SEQ_n_DW_3	Sequence parameter 12, 13, 14, 15
[0-255][0-255][0-255][0-255]	SEQ_n_DW_4	Sequence parameter 16, 17, 18, 19

Tab. 118 - Seq n register

### 16.2.18. Start, Stop and Input Frequency Configuration

Start and step inputs configuration registers:

- IN\_1\_CNF

Stop and direction inputs configuration registers:

- IN\_2\_CNF

are Read / Write registers, which have different configurations depending on drive model.

Since FD1.1E is equipped with only two inputs, all the functions have been made available for this drive as per following tables.

Value	Function	Description
0	CURRENT_DISABLE	This input disables/enables the motor current, freeing the shaft. In this situation if the shaft is manually moved beyond synchronous mode limit ( $\pm 1.8^\circ$ ) the red led will be steady on. If further moved beyond the step accumulation limit the related alarm arises. The re-activation of the current will reset the alarms.
1	FREQUENCY_DISABLE	It disables all the movement commands. The motor remains in torque, i.e. current keeps on flowing in the motor windings, but it does not move and the position counter does not change. This input can be used as a selection input, when the same frequency signal is multiplexed to several drive.
2	HOMING_NO	Homing Switch normally open / close.
3	HOMING_NC	
4	not used	
5	HW_LIMIT_SWITCH_UP_NO	Hardware Limit Switch up / down normal open / close. When it is active all the movements towards increasing or decreasing positions are inhibited.
6	HW_LIMIT_SWITCH_UP_NC	
7	HW_LIMIT_SWITCH_DW_NO	
8	HW_LIMIT_SWITCH_DW_NC	
9	ENC_LATCH	It latches the multi-turn encoder position via interrupt. Latched value is saved into ENC_LATCH_RIS and ENC_LATCH_FAL registers, depending on the input signal edge.
10	STEP	Every pulse produces one $\mu$ step
11	SEL_CYC_SEQ	Ref. to 0 Cycle and sequence selection
12	DELTA_STOP_NO	Ref. to 0 Delta stop
13	DELTA_STOP_NC	
14	QUAD_STEP	It is used in quadrature input frequency mode. Every edge produces one $\mu$ step.
15	START_NO	To start and stop the selected cycle or sequence.
16	START_NC	
17	STOP_NO	
18	STOP_NC	
19	ENABLE_CURRENT	This input enables/disables the motor current, freeing the shaft. The activation of the current will reset the alarms.
20	START_NO_STOP_NC	To start the selected cycle or sequence and to stop the running one.
21	START_NC_STOP_NO	
22	INTERLOCK_CURRENT	When this input is active, motor current can be managed via fieldbus. When the input is inactive, motor current will be disabled.

Tab. 119 - FD1.1E IN1 configuration register

Value	Function	Description
0	CURRENT_DISABLE	This input disables/enables the motor current, freeing the shaft. In this situation if the shaft is manually moved beyond synchronous mode limit ( $\pm 1.8^\circ$ ) the red led will be steady on. If further moved beyond the step accumulation limit the related alarm arises. The re-activation of the current will reset the alarms.
1	FREQUENCY_DISABLE	It disables all the movement commands. The motor remains in torque, i.e. current keeps on flowing in the motor windings, but it does not move and the position counter does not change. This input can be used as a selection input, when the same frequency signal is multiplexed to several drive.
2	HOMING_NO	Homing Switch normally open / close.
3	HOMING_NC	
4	not used	
5	HW_LIMIT_SWITCH_UP_NO	Hardware Limit Switch up / down normal open / close. When it is active all the movements towards increasing or decreasing positions are inhibited.
6	HW_LIMIT_SWITCH_UP_NC	
7	HW_LIMIT_SWITCH_DW_NO	
8	HW_LIMIT_SWITCH_DW_NC	
9	ENC_LATCH	It latches the multi-turn encoder position via interrupt. Latched value is saved into ENC_LATCH_RIS and ENC_LATCH_FAL registers, depending on the input signal edge.
10	DIR	It is used in STEP / DIR mode. 0: CW towards increasing positions, 1: CCW towards decreasing positions.
11	SEL_CYC_SEQ	Ref. to 0 Cycle and sequence selection
12	DELTA_STOP_NO	Ref. to 0 Delta stop
13	DELTA_STOP_NC	
14	QUAD_STEP	It is used in quadrature input frequency mode. Every edge produces one $\mu$ step.
15	START NO	To start and stop the selected cycle or sequence.
16	START NC	
17	STOP NO	
18	STOP NC	
19	ENABLE_CURRENT	This input enables/disables the motor current, freeing the shaft. The activation of the current will reset the alarms.
20	START_NO_STOP_NC	To start the selected cycle or sequence and to stop the running one.
21	START_NC_STOP_NO	
22	INTERLOCK_CURRENT	When this input is active, motor current can be managed via fieldbus. When the input is inactive, motor current will be disabled.

Tab. 120 - FD1.1E IN2 configuration register

On FD1.xEC, FD2.1E, FD2.1EC start and step frequency inputs on IN1 and IN2 as per following tables.

Value	Function	Description
0	STEP	Every pulse produces one $\mu$ step
1	START NO	To stop the running cycle or sequence.
2	START NC	
4	not used	
6	QUAD_STEP_A	It is used in quadrature input frequency mode. Every edge produces one $\mu$ step.
7	START_NO_STOP_NC	To start the selected cycle or sequence and to stop the running one.
8	START_NC_STOP_NO	
30	START_NO_STOP_NC_CYC1	To increase +1 the selected cycle or sequence and then it starts the selected or it stops the running. It can be used to have separate inputs to start different cycles. E.g. one push-button can start one cycle, a second push-button, connected to the input configured as _CYC2, starts the second cycle.
31	START_NC_STOP_NO_CYC1	
32	START_NO_STOP_NC_CYC2	To increase +2 the selected cycle or sequence and then it starts the selected or it stops the running.
33	START_NC_STOP_NO_CYC2	

Tab. 121 - FD1.xEC, FD2.1E, FD2.1EC IN1 configuration register

Value	Function	Description
0	DIR	
1	STOP NO	To stop the running cycle or sequence.
2	STOP NC	
4	not used	
6	QUAD_STEP_B	It is used in quadrature input frequency mode. Every edge produces one $\mu$ step.
7	START_NO_STOP_NC	To start the selected cycle or sequence and to stop the running one.
8	START_NC_STOP_NO	
11	SEL_CYC_SEQ	Ref. to 0 Cycle and sequence selection
30	START_NO_STOP_NC_CYC1	To increase +1 the selected cycle or sequence and then it starts the selected or it stops the running. It can be used to have separate inputs to start different cycles. E.g. one push-button can start one cycle, a second push-button, connected to the input configured as _CYC2, starts the second cycle.
31	START_NC_STOP_NO_CYC1	
32	START_NO_STOP_NC_CYC2	To increase +2 the selected cycle or sequence and then it starts the selected or it stops the running.
33	START_NC_STOP_NO_CYC2	

Tab. 122 - FD1.xEC, FD2.1E, FD2.1EC IN2 configuration register

### 16.2.19. Multipurpose Inputs Configuration

IN3 and IN4 are multipurpose inputs configuration registers.

FD1.xEC, FD2.1E, FD2.1xC are equipped with these inputs. On FD2.1xC IN4 can be used as digital or analog input.

IN\_3\_CNF and IN\_4\_CNF shall be configured as per following tables:

Value	Function	Description
0	CURRENT_DISABLE	This input disables/enables the motor current, freeing the shaft. In this situation if the shaft is manually moved beyond synchronous mode limit ( $\pm 1.8^\circ$ ) the red led will be steady on. If further moved beyond the step accumulation limit the related alarm arises. The re-activation of the current will reset the alarms.
1	FREQUENCY_DISABLE	It disables all the movement commands. The motor remains in torque, i.e. current keeps on flowing in the motor windings, but it does not move and the position counter does not change. This input can be used as a selection input, when the same frequency signal is multiplexed to several drive.
2	HOMING_NO	Homing switch normally open / close.
3	HOMING_NC	
4	not used	
5	HW_LIMIT_SWITCH_UP_NO	Hardware limit switch up / down normal open / close. When it is active all the movements towards increasing (up) or decreasing (down) positions are inhibited.
6	HW_LIMIT_SWITCH_UP_NC	
7	HW_LIMIT_SWITCH_DW_NO	
8	HW_LIMIT_SWITCH_DW_NC	
9	ENC_LATCH	It latches the multi-turn encoder position via interrupt. Latched value is saved into ENC_LATCH_RIS and ENC_LATCH_FAL registers, depending on the input signal edge.
10	DIR	It is used in STEP / DIR mode. 0: CW towards increasing positions, 1: CCW towards decreasing positions.
11	SEL_CYC_SEQ	Ref. to 0  Cycle and sequence selection
12	DELTA_STOP_NO	Ref. to 0  Delta stop
13	DELTA_STOP_NC	
15	START_NO	To start and stop the selected cycle or sequence.
16	START_NC	
17	STOP_NO	
18	STOP_NC	
19	ENABLE_CURRENT	This input enables/disables the motor current, freeing the shaft. The activation of the current will reset the alarms.
22	INTERLOCK_CURRENT	When this input is active, motor current can be managed via fieldbus. When the input is inactive, motor current will be disabled.
26	ENC_LATCH_RIS	It latches the multi-turn encoder position via interrupt on rising edges only and saves the latched value in ENC_LATCH_RIS register.
27	ENC_LATCH_FALL	It latches the multi-turn encoder position via interrupt on rising edges only and saves the latched value in ENC_LATCH_FAL register.
30	AIN_SPEED_SCALING	FD2.1xC only. To scale the speed of the cycles based on analog input. Cycle speed data is the speed when analog input is at full scale

Tab. 123 - IN3 configuration register



### 16.2.20. Aux Inputs

FD2.1E is equipped with three additional inputs: IN\_CNF\_5, IN\_CNF\_7, IN\_CNF\_8. They are Read / Write registers, which can be configured as follows.

Value	Function	Description
11	SEL_CYC_SEQ	Ref. to 0 Cycle and sequence selection

Tab. 124 - FD2.1E aux inputs configuration

### 16.2.21. Analog input value

Register IN\_VAL\_A1 is a Read Only register to be used to read the value of analog input of FD2.1xC. The range is 0-4095.

### 16.2.22. Output

Output configuration register OUT\_1\_CNF is a Read / Write registers, which have different configuration depending on drives. On FD1.1E it can be configured as per following table:

Value	Function	Description
0	CYCLE_RUNNING	Active on running cycles or sequences.
1	ENC_INDEX_50	It produces one pulse of 50ms at POSITION_OFFSET angular position.
2	ENC_INDEX_Π	It is high at half revolution starting from POSITION_OFFSET angular position.
3	POS_UPLOADED	Exact power-off position has been restored.
4	AX_CALIBRATED	Axis is calibrated.
5	POWER_ON_POS_OK	Position is restored compensated with the power-off position deviation
6	TORQUE_LIM	Drive is following the ordered position, out of synchronism.
7	CYCLE_RUN_NDSTOP	Active on running cycles or sequences with timeout logic for delta stop signal
8	FORCED_L	Output is forced open
9	FORCED_H	Output is forced high
10	DRIVE_OK	The output drive ok is normally closed and it opens in case of alarm (recommended)
11	ALARM	The output alarm is normally opened and it closes in case of alarm

Tab. 125 - FD1E OUT1 configuration register

On FD1.xEC, FD2.1E, FD2.1xC OUT\_1\_CNF can be configured as:

Value	Function	Description
0	DRIVE_OK	The output drive ok is normally closed and it opens in case of alarm (recommended)
1	ALARM	The output alarm is normally opened and it closes in case of alarm
8	FORCED_L	Output is forced open
9	FORCED_H	Output is forced high

Tab. 126 - FD2E OUT1 configuration register

On FD1.xEC, FD2.1E, FD2.1xC OUT\_2\_CNF can be configured as:

Value	Function	Description
0	CYCLE_RUNNING	Active on running cycles or sequences.
1	ENC_INDEX_50	It produces one pulse of 50 ms at POSITION_OFFSET angular position.
2	ENC_INDEX_Π	It is high at half revolution starting from POSITION_OFFSET angular position.
3	POS_UPLOADED	Exact power-off position has been restored.
4	AX_CALIBRATED	Axis is calibrated.
5	POWER_ON_POS_OK	Position is restored compensated with the power-off position deviation
6	TORQUE_LIM	Drive is following the ordered position, out of synchronism.
7	CYCLE_RUN_NDSTOP	Active on running cycles or sequences with timeout logic for delta stop signal
8	FORCED_L	Output is forced open
9	FORCED_H	Output is forced high

Tab. 127 - FD1.xEC, FD2.1E, FD2.1xC OUT2 configuration register

Output configuration register FD2.1E OUT\_3\_CNF is a Read / Write registers, which can be configured as follows:

Value	Function	Description
8	FORCED_L	Output is forced open
9	FORCED_H	Output is forced high

Tab. 128 - FD2.1E OUT3 configuration register

### 16.2.23. Homing, Time Stop, Release and Research Speeds

HOMING, TS, V\_RESEARCH, V\_RELEASE are Read/Write registers. These registers are used to setup the homing movement to be launched with a homing cycle.

TS (Time Stop) indicates the waiting time between V\_RESEARCH and V\_RELEASE speeds (it is expressed in milliseconds).

HOME\_POSITION is loaded into the position register and STATUS\_WORD bit 11 AX\_CALIBRATED is set at the successful conclusion of all the homing cycles.

Value	Function	Description
0	INDEX	Ref. to 14.4 Homing
1	HOME_SIMPLE	
2	HOME_TS_INV	
3	HOME_TS_NO_INV	
4	HOME_TS_INV_MRK	
5	HOME_TS_NO_INV_MRK	
6	MECH_STOP	
7	MECH_STOP_MRK	
8	CURR_POS	

Tab. 129 - Homing register

### 16.2.24. Position Software Limit Switch Up/Down

POS\_SW\_LS\_UP and POS\_SW\_LS\_DW are Read/Write registers that can be used to setup the software limit switches. They need to be enabled by setting CONFIG register bits 4, 5.

### 16.2.25. Temperature

TEMP is a Read Only register that shows the microcontroller temperature in Celsius degrees. When the temperature rises above 100 °C, the temperature alarm is given.

### 16.2.26. Temperature Offset

TEMP\_OFF is a Read Only register used to calibrate the microcontroller temperature sensor. It cannot be modified.

**16.2.27. K**

KNUM\_NOM and KV are Read Only registers used for setting up the current loop control. They cannot be modified.

**16.2.28. Steps Accumulation Limit**

ACCUMULATION\_LIMIT is a Read/Write register used for setting the step accumulation alarm limit. This register is upper limited to 128'000. If the accumulated steps exceed the ACCUMULATION\_LIMIT value the motor stops, the ERR\_FAT will be set to 1 and the red LED will be steady on.

Real time value of accumulated steps can be monitored reading ACCUMULATED\_STEPS Read Only register.

**16.2.29. Encoder Position, Speed, Latch and Revolution**

ENC\_POS and ENC\_SPEED are two Read Only registers that show the encoder position in  $\mu$ steps and the encoder speed in  $\mu$ steps per second.

ENC\_LATCH\_RIS and ENC\_LATCH\_FAL are a Read Write registers used in combination with the multipurpose inputs encoder latch settings. When one of this input is configured as encoder latch and it gets active, the multi-turn encoder position is saved into ENC\_LATCH\_RIS register, while when the input gets inactive it is saved into ENC\_LATCH\_FAL register.

ENC\_REV is a Read Only register that shows the encoder position inside the revolution (it ranges from 0 to 4095).

**16.2.30. Power On Calibration Limit**

PON\_CALIB\_LIM is a Read/Write register used for power on position restore. Ref. to 9 POWER-ON POSITION RESTORE (QUASI-ABSOLUTE MULTI-TURN)

**16.2.31. Time Constant**

TIME\_CONST is a Read/Wirte register that shall not be modified.

**16.2.32. Start Stop Frequency**

START\_STOP\_FREQ is a Read/Write register used for setting the start stop frequency. It is expressed in  $\mu$ steps per second.

**16.2.33. Resolution**

RESOLUTION\_NUM and RESOLUTION\_DEN are two Read-Only registers, which can be modified only during programming. They are unsigned 16 bits each. Their ratio shall be in the range [8 – 4 096], allowing a resolution in the range [400 – 204 800]  $\mu$ step per revolution. Following formula to be used:

$$\text{Resolution} = 50 \cdot \frac{\text{RESOLUTION}_{\text{NUM}}}{\text{RESOLUTION}_{\text{DEN}}},$$

expressed in  $\mu$ step per revolution.

### 16.2.34. CANopen Baud Rate, Address and Status

CAN\_BAUDRATE and STATION\_ADDRESS are two Read/Write registers.

CAN\_BAUDRATE need to be set using integer numbers from 0 to 8 that correspond to the following frequencies:

0: 1000 KHz, 1: 800 KHz, 2: 500 KHz, 3: 250 KHz, 4: 125 KHz, 5: 100 KHz, 6: 50 KHz, 7: 20 KHz, 8: 10 KHz.

When DIP switches selection is zero (all off), STATION\_ADDRESS can be used to program the drive CANopen address.

When DIP switches selection differs from zero, STATION\_ADDRESS will reflect DIP selection.

After modifying the address, it is necessary to give a NMT command: Reset node or Reset communication to make the modification effective.

CAN\_ERR\_STATUS is a Read-Only register, which is used to monitor the CAN peripheral status.

Bit number	Name	Description
[0 – 7]	TEC	Transmit error counter. The implementing part of the fault confinement mechanism of the CAN protocol.
[8 – 15]	REC	Receive error counter. In case of an error during reception, this counter is incremented by 1 or by 8 depending on the error condition as defined by the CAN standard. After every successful reception the counter is decremented by 1 or reset to 120 if its value was higher than 128. When the counter value exceeds 127, the CAN controller enters the error passive state.
16	BUS_OFF	Bus off, i.e. TEC greater than 255. Once this condition is reached, the bus off state is left automatically by hardware once 128 occurrences of 11 recessive bits have been monitored.
[17 – 19]	LAST_ERROR	0: No Error 1: Stuff Error 2: Form Error 3: Acknowledgment Error 4: Bit recessive Error 5: Bit dominant Error 6: CRC Error
20	ACK_INIT	CAN hardware cannot synchronize (monitor a sequence of 11 consecutive recessive bits on the CAN RX signal)
21	RX_SIGNAL	0: recessive 1: dominant
22	TX_MODE	CAN hardware is in transmission
23	RX_MODE	CAN hardware is in reception
24	ARBITRATION_LOST	Previous TX failed due to an arbitration lost
25	TX_ERROR	Previous TX failed due to an error
26	RX_FIFO_0_OVERRUN	This bit is set by hardware when a new message cannot be received because the FIFO 0 or FIFO 1 were full.
27	RX_FIFO_1_OVERRUN	

Tab. 130 - CANopen error and status

### 16.2.35. Cycles Counters

CNT\_CYC and CNT\_DSTOP are two Read/Write registers used to check the correct execution of ordered command.

Every time a positioning movement is completed (relative, absolute or delta stop) V6 compares the arrival position with the wanted destination to make sure the target is reached. If they are identical CNT\_CYC register is incremented. If the cycle is a delta stop cycle and the delta stop input activation has been the reason of motor stopping, also CNT\_DSTOP register is incremented.

### 16.2.36. Encoder Status

ENC\_STATUS is a Read register used to monitor the encoder status. If all bits are zero, no problem is present.

FD1.1E, FD2.1E and FD2.xC encoder has following status:

Bit number	Name	Description
0	Parity Error	0: Ok, 1: parity error in the reading of absolute position.
1	Magnetic Decrement	0: Ok, 1: decrement of the magnetic field of the encoder sensor.
2	Magnetic Increment	0: Ok, 1: increment of the magnetic field of the encoder sensor.
3	Linearity Error	0: Ok, 1: encoder reading linearity error.
4	CORDIC Overflow	0: Ok, 1: Overflow of the Coordinate Rotation Digital Computer (CORDIC) that calculates the angle and the intensity of the rotating field.
5	Offset Compensation	0: Ok, 1: Offset compensation not finished.
7	Current Disabled	0: Ok, 1: MOSFET gate drivers are disabled.

Tab. 131 - FD1.1E, FD2.1E and FD2.xC encoder status

Bits 1, 2 and 3 represent the correct magnetic field of the encoder as per following table:

Status bits			Description
Mag Inc	Mag Dec	Lin	
0	0	0	Magnetic input field OK (GREEN range, ~45mT to 75mT)
1	1	0	YELLOW range: magnetic field is ~ 25mT to 45mT or ~75mT to 135mT. The drive could still be operated in this range, but torque control might be affected.
1	1	1	RED range: magnetic field is ~<25mT or >~135mT. It is not recommended to use the drive in this condition.

Tab. 132 - FD1.1E, FD2.1E and FD2.xC encoder magnetic status

FD2.1xC mount a different encoder, whose status is as follows:

Bit number	Name	Description
0	Agc-warning	Agc-warning=1. The flag sets to 1 in case the AGC Value reaches 0LSB or 255LSB. The detailed information which level is reached can be found in the diagnostic register.
1	MagHalf	This flag sets to 1 in case the AGC Value reaches 255 LSB and the magnitude value is the half of the of the regulated magnitude value (between AGC = 0LSB and AGC = 255LSB) which is typical 4800LSB.
2	P2ram_warning	ECC is correcting one bit of P2RAM in customer area
3	P2ram_error	ECC has detected 2 uncorrectable errors in P2RAM in customer area
4	Framing_error	Framing if SPI communication wrong
5	Command_error	SPI invalid command received
6	CRC_error	CRC error during SPI communication
7	WDTST	Watchdog information. In case the flag sets to 1, the internal oscillator or the watchdog is not working correctly
8	BRKHALL	Broken Hall element information
9	OffCompNotFinished	In case the flag is 1 the internal offset compensation is not finished
10	CORDIC Overflow	Reading the Overflow bit of the CORDIC
...		

16	Vdd_mode	0: VDD 3.0 Mode 1: VDD 5.0 Mode
17	LoopsFinished	All magneto core loops finished
18	Cordic_overflow	Error flag CORDIC overflow
19	Comp_l	Warning flag AGC low
20	Comp_h	Warning flag AGC high
21	MagHalf_flag	Error flag magnitude is below half of target value
22	CosOff_fin	Cosine offset compensation finished
23	SinOff_fin	Sine offset compensation finished
24	Off comp finished	Error flag offset compensation finished
25	AGC_finished	Initial AGC settling finished
26	Fusa_error	Error flag broken Hall element
27:28	SPI_cnt	SPI frame counter

Tab. 133 - FD2.1xC encoder status

### 16.2.37. EtherCAT SyncManager2 Period Average and Variance

EtherCAT only. EtherCAT synchronous network can be setup as SM sync or DC sync. When used in SM sync, the drive is synchronized on received setpoints on SM2. Via FRAME\_PERIOD\_AV and FRAME\_PERIOD\_VAR it is possible to monitor the average and the variance of the received frames period.

SHIFT\_SM2\_DC represent the microseconds between the reception of the frame in SM2 and the reception of Distributed Clock.

### 16.2.38. EtherCAT Sync managers properties

EtherCAT only. SM0\_0 and SM0\_1 contain the mailbox input configurations.

SM1\_0 and SM2\_1 contain the mailbox output configurations.

SM2\_0 and SM2\_1 contain the PDO input configurations.

SM3\_0 and SM3\_1 contain the PDO output configurations.

Bit number	Name	Description
0-15	SyncManager x Physical Start Address Register	Specifies the first byte that will be handled by SyncManager x.
16-31	SyncManager x Length Register	Number of bytes assigned to SyncManager x. (This field shall be greater than 1, otherwise the SyncManager is not activated. If set to 1, only Watchdog Trigger is generated, if configured.)

Tab. 134 - SMx\_0

Bit number	Name	Description
0-1	Operation Mode	00: Buffered (3 buffer mode) 10: Mailbox (single buffer mode)
3:2	Direction	00: Read: ECAT read access, PDI write access 01: Write: ECAT write access, PDI read access
4	Interrupt in ECAT Event Request Register	1: Enabled
5	Interrupt in PDI Event Request Register	1: Enabled
6	Watchdog Trigger Enable	1: Enabled
...		
8	Interrupt Write	0: Interrupt cleared after first byte of buffer was read 1: Interrupt after buffer was completely and successfully written
9	Interrupt Read	0: Interrupt cleared after first byte of buffer was written 1: Interrupt after buffer was completely and successfully read
11	Mailbox Status	Mailbox Mode: 0: Mailbox Empty 1: Mailbox Full
12	Buffer Status (Last Written Buffer)	Buffered Mode: 00: 1. buffer 01: 2. buffer 10: 3. buffer 11: No buffer written
...		
16	SyncManager Enable/Disable	0: Disable: Access to memory without SyncManager control 1: Enable: SyncManager is active and controls memory area set in configuration.
17	Repeat Request	A toggle of Repeat Request indicates that a mailbox retry is needed (primarily used in conjunction with ECAT Read Mailbox)
...		
22	Latch Event ECAT	0: No 1: Generate latch event if EtherCAT master issues a buffer exchange.
23	Latch Event PDI	0: No 1: Generate latch events if PDI issues a buffer exchange or if PDI accesses buffer start address.
24	Deactivate SyncManager x	Read: 0: Normal operation, SyncManager x activated 1: SyncManager x deactivated and reset SyncManager x locks access to memory area Write: 0: Activate SyncManager 1: Request SyncManager Deactivation
25	Repeat Ack	If this is set to the same value as Repeat Request, the PDI acknowledges the execution of a previous set repeat request.

Tab. 135 - SMx\_1

### 16.2.39. Main period

MAIN\_PERIOD is a Read Only register that represents in  $\mu$ sec the running period of main function.

### 16.2.40. Supply voltage

V\_DC is a Read-Only register, which shows the DC bus voltage applied to the drive, expressed with a resolution of 10 mV.

### 16.2.41. EtherCAT AL Status register

EtherCAT only. Represents EtherCAT slave controller registers 0x130 AL status register and 0x134 AL status code register.

Bit number	Name	Description
0-3	Actual State of the Device State Machine	Actual State of the Device State Machine 1h: Init State 2h: Pre-Operational State 3h: Bootstrap State 4h: Safe-Operational State 8h: Operational State
4	Error Ind	0: Device is in state as requested or Flag cleared by command 1: Device has not entered requested state or changed state as a result of a local action
...		
16...31	AL Status Code	

Tab. 136 - AL status register

### 16.2.42. Ports error counters

EtherCAT only.

Bit number	Name	Description
0-7	Port 0 Invalid Frame Counter	Counting is stopped when 0xFF is reached.
8-15	Port 0 RX Error Counter	Counting is stopped when 0xFF is reached. This is coupled directly to RX ERR of the MII/EBUS interfaces.
16-23	Port 1 Invalid Frame Counter	Counting is stopped when 0xFF is reached.
24-31	Port 1 RX Error Counter	Counting is stopped when 0xFF is reached. This is coupled directly to RX ERR of the MII/EBUS interfaces.

Tab. 137 - P0\_P1\_RX\_ERROR\_CNT

Bit number	Name	Description
0-7	Port 2 Invalid Frame Counter	Counting is stopped when 0xFF is reached.
8-15	Port 2 RX Error Counter	Counting is stopped when 0xFF is reached. This is coupled directly to RX ERR of the MII/EBUS interfaces.
16-23	Port 3 Invalid Frame Counter	Counting is stopped when 0xFF is reached.
24-31	Port 3 RX Error Counter	Counting is stopped when 0xFF is reached. This is coupled directly to RX ERR of the MII/EBUS interfaces.

Tab. 138 - P2\_P3\_RX\_ERROR\_CNT

### 16.2.43. ESC registers read/write command

EtherCAT only. EtherCAT slave controller registers can be read or written via ESCREG\_CMD\_ADD, data to be written or read can be managed via ESCREG\_DATA\_0, ESCREG\_DATA\_1.

Bit number	Name	Description
0-7	Command	1: read 2: write
8-23	Address	Address of the ESC register
24-31	Size	Number of Bytes to be read/written

Tab. 139 - ESCREG\_CMD\_ADD



### 16.2.44. PDO Param and mapping

CANopen only. PDO parameters are used to configure the CANopen from flash, saving setup time.

Bit number	Name	Description
0-15	COB-ID	COB-ID of the PDO
16-23	Type	Type of the PDO
24-31	Status	0: disabled 1: enabled

*Tab. 140 - PDO Param*

Bit number	Name	Description
0-7	Length	Number of bits of the object
8-15	Sub-Index	Sub-index of the object
16-31	Index	Index of the object

*Tab. 141 - PDO mapping from flash*

### 16.2.45. CANopen Lifetime, Consumer and Producer Heartbeat

CANopen only.

Bit number	Name	Description
0-15	Guard time	Ref. to 11.2.1 Network Management Objects (NMT)
16-31	Life time factor	Ref. to 11.2.1 Network Management Objects (NMT)

*Tab. 142 - CAN\_LIFETIME*

Bit number	Name	Description
0-15	Consumer heartbeat time	Ref. to 11.2.1 Network Management Objects (NMT)
16-23	Consumer heartbeat ID	Ref. to 11.2.1 Network Management Objects (NMT)

*Tab. 143 - CAN\_CONS\_HEARTBEAT*

Bit number	Name	Description
0-31	Producer heartbeat time	Ref. to 11.2.1 Network Management Objects (NMT)

*Tab. 144 - CAN\_PROD\_HEARTBEAT*

### 16.2.46. Serial number

SERIAL\_NUMBER is a Read Only register to read the S/N of the device.

## 17. MODBUS PROTOCOL

Modbus protocol defines the format and the modality of communication between a master and one or more slaves that answer to master's requests.

V8 implements Modbus protocol as a slave with:

- RTU mode,
- 8 data bits,
- 1 stop bit,
- no parity bit.

The Modbus message is composed of:

- Device address (from 1 up to 247)
- Function code: V8 implements Read Holding Registers (03), Write Single Holding Register (06), Write Multiple Register (16), Write File Record (21).
- Data
- Error analysis (CRC16 algorithm)

If an error is present (format error or CRC16 error), the message is considered not valid and discarded and the slave will produce a 5 Bytes error message. In case of CRC error, no answer will take place.

Use Address 0 to send broadcast messages (no drive will answer).

## 17.1. Read Holding Register (03)

This function allows reading the values of 16 bits registers. V8 implements up to 125 registers reading at a time.

### 17.1.1. Master Read Request

In order to read the current position, speed and cycle of the driver number 13 the following read command have to be performed.

Slave address	Function code	Address		Quantity of words		CRC16	
1 Byte	1 Byte	2 Bytes		2 Bytes		2 Bytes	
13	03	0	8	0	6	68	198

Tab. 145 - CURR\_POS, CURR\_SPEED and CURR\_CYCLE read request

### 17.1.2. Slave Read Answer

For example if:

- current speed is 30'000 (corresponding to words: 0 H and 30'000 L)
- current position is 230'113 steps (corresponding to words: 3 H and 33'505 L),
- current cycle is 29

the slave answers would be:

Slave Add	Function Code	Quantity of Bytes	Current Position [32 bits]				Current Speed [32 bits]				Current Cycle [32 bits]				CRC 16	
1 B	1 B	1 B	4 Bytes				4 Bytes				4 Bytes				2 Bytes	
13	3	12	0	0	117	48	0	3	130	225	0	0	0	29	136	114

Tab. 146 - CURR\_SPEED, CURR\_POS, CURR\_CYCLE read answer

## 17.2. Write Single Holding Register (06)

This function allows to write the value of one 16 bits register.

### 17.2.1. Master Write Request

In order to write 100'000 to Cycle 0 Speed register (32 bits) (corresponding to words: 1 H and 34'464 L) of the drive number 13 the two following message (High and Low parts of the double word CYC\_0\_SPEED) have to be generated.

Slave address	Function code	Address		Data		CRC16	
1 Byte	1 Byte	2 Bytes		2 Bytes		2 Bytes	
13	06	0	42	0	1	105	14

*Tab. 147 - CYC\_0\_SPEED\_H write request*

Slave address	Function code	Address		Data		CRC16	
1 Byte	1 Byte	2 Bytes		2 Bytes		2 Bytes	
13	06	0	43	134	160	155	22

*Tab. 148 - CYC\_0\_SPEED\_L write request*

### 17.2.2. Slave Write Answer

The slave write answer consists of the re-transmission of the received message.

Slave address	Function code	Address		Data		CRC16	
1 Byte	1 Byte	2 Bytes		2 Bytes		2 Bytes	
13	06	0	42	0	1	105	14

*Tab. 149 - CYC\_0\_SPEED\_H write answer*

Slave address	Function code	Address		Data		CRC16	
1 Byte	1 Byte	2 Bytes		2 Bytes		2 Bytes	
13	06	0	43	134	160	155	22

*Tab. 150 - CYC\_0\_SPEED\_L write answer*

### 17.3. Write Multiple Holding Register (16)

This function allows to write the values of multiple 16 bits registers in the same message. V8 implements up to 123 registers writing at a time.

#### 17.3.1. Master Write Request

In order to write the Cycle 0 Type, Speed, Position, Direction and Delta Stop registers:

- Type equal to positioning relative (corresponding to words: 0 H and 2 L)
- Speed equal to 100'000 (corresponding to words: 1 H and 34'464 L)
- Position equal to 270'000 (corresponding to words: 4 H and 7'856 L)
- Direction equal to 1 (corresponding to words: 0 H and 1 L)
- Delta Stop equal to 1'000 (corresponding to words: 0 H and 1'000 L)

of the driver number 13 the following message have to be generated.

Slave Add	Function	Register Address	Words Count	Byte Count	Type [32 bits]	Speed [32 bits]	Position [32 bits]	Direction [32 bits]	Delta Stop [32 bits]	CRC16
1B	1B	2 B	2 B	1B	4 Bytes	4 Bytes	4 Bytes	4 Bytes	4 Bytes	2 B
13	16	0 40	0 10	20	0 0 0 2	0 1 134 160	0 4 30 176	0 0 0 1	0 0 3 232	21 205

Tab. 151 - CYC\_0\_DATA write request

#### 17.3.2. Slave Write Answer

Slave Add	Function	Register Address	Word Count	CRC 16
1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes
13	16	0 40	0 10	192 202

Tab. 152 - CYC\_0\_DATA write answer

## 17.4. Write File Record (21)

This function code is used to perform a file record write.

Request Length is limited to 251, as the Modbus message cannot be longer than 256 Bytes.

Request Type is always equal to 6.

Two files numbers are supported:

- FD Firmware = 1,
- User Data = 2.

Before re-programming the FD Firmware, it is necessary to set the drive in programming mode.

This is performed by setting the EXE\_FUN register equal to 20. Consequently, bit 18 of status word register will be set. It is not necessary to set the drive in programming mode when programming file 2, User Data.

A file is an organization of records. The first Write File Record request must begin with Record Number equal to zero. The next requests must have a sequential record numbers (1, 2, 3, ...).

All Request Lengths are provided in terms of number of Bytes and all Record Lengths are provided in terms of the number of 16-bit words.

At the end of the file, when the programming is complete, it is necessary to write the RST command into EXE\_FUN register.

### 17.4.1. Master Write File Request

Slave Add	Function	Req Length	Req Type	File Number	Record Number	Record Length	Record Data	CRC16
1B	1B	1B	1B	2 B	2B	2B	12B	2 B
13	21	19	6	0 1	0 0	0 6	35 96 32 0 33 69 8 0 115 237 8 0	119 209

Tab. 153 – Master write file request

### 17.4.2. Slave Write File Answer

The normal response is an echo of the request.

Slave Add	Function	Req Length	Req Type	File Number	Record Number	Record Length	Record Data	CRC16
1B	1B	1B	1B	2 B	2B	2B	12B	2 B
13	21	19	6	0 1	0 0	0 6	35 96 32 0 33 69 8 0 115 237 8 0	119 209

Tab. 154 - Slave write file answer

## 17.5. Checksum Calculation

```
#define CRC_POLY_16_MODBUS    0xA001
#define CRC_START_MODBUS     0xFFFF
#define MODBUS_BUFF_SIZE     256

static void init_CRC16_tab_Modbus(void)
{
    u16 i, j, crc, c;

    for (i = 0; i < 256; i++)
    {
        crc = 0;
        c = i;

        for (j = 0; j < 8; j++)
        {
            if ((crc ^ c) & 0x0001)
                crc = (crc >> 1) ^ CRC_POLY_16_MODBUS;
            else
                crc = crc >> 1;
            c = c >> 1;
        }
        crc_tab16_Modbus[i] = crc;
    }
}

static u16 CRC16_Modbus(u8 *chkbufBase, u8 lIxStart, u16 len)
{
    u16 crc, tmp, short_c, a, lIx;
    u8 *ptr;

    crc = CRC_START_MODBUS;

    lIx = lIxStart;

    for (a = 0; a < len; a++)
    {
        ptr = &chkbufBase[lIx];
        short_c = 0x00ff & (u16)*ptr;
        tmp = crc ^ short_c;
        crc = (crc >> 8) ^ crc_tab16_Modbus[tmp & 0xff];
        lIx++;
        lIx %= MODBUS_BUFF_SIZE;
    }
    return ((crc & 0xFF) << 8) + (crc >> 8);
}
```