

# TG Motion

## version 4

### SERVO type group

### operation manual

**Revision History**

<b>date</b>	<b>version</b>	<b>revision</b>
31 July 2017	1.0	Initial release

**Contents**

1. SERVO group.....	4
1.1 SERVO group description .....	4
1.2 SERVO group constituent parts .....	4
2. Servo drive control and diagnostics .....	5
2.1 Servo drive control principle .....	5
2.2 Important registers.....	6
2.3 Register principle description and use .....	6
3. Profile Generator of desired position (PG) .....	9
3.1 Important registers.....	9
3.2 Description of the PG structure .....	9
3.3 Ramp generator.....	9
3.4 Using profile generator .....	10
4. GEAR Generator of desired position.....	13
4.1 Description of the structure GEAR .....	13
4.2 Important registers.....	13
4.3 Cancelling cam .....	15
4.4 Incremental cam .....	15
4.5 External data table.....	16
4.6 Use of GEAR generator.....	17
5. COMMAND functional interface .....	19
5.1 Important registers.....	19
5.2 Description of the structure Command.....	19
5.3 Use of Command functional interface .....	21
6. Interface for SDO communication .....	23
6.1 Description of the structure SDO.....	23
6.2 Important registers.....	23
6.3 Registers: description and how to work with them .....	23
6.4 Use of the SDO structure .....	23
7. Capture interface .....	25
7.1 Description of the structure Capture.....	25
7.2 Important registers.....	25
7.3 Description of the structure Capture.....	25
7.4 Use of Capture interface.....	25
8. Appendix .....	26
List and description of Servo group registers .....	26

# 1. SERVO group

## 1.1 SERVO group description

The Servo group unifies the control interface for different servo drive types. From the PLC code user's and programmer's point of view, all servo drives behave in the same way, they have identical register to control them or change their setting. Therefore, servo drives may be changed operatively, without any need for rewriting the Virtual PLC code. The same PLC code may be applied to a multitude of different servo drives.

The Servo group makes an interface for servo drive control and diagnostics. It allows the user to check general registers and easily control the servo drives through virtual generators of desired position, such as the profile generator (PG) for absolute and relative position control of point-to-point type and speed control.

It further offers the GEAR generator for linear synchronization and simulation of cancelling and incremental cams. For simpler PG and GEAR generator-based tasks, **COMMAND** functional interface is available. Application of this interface simplifies the program code written by the user and speeds up the Virtual PLC operation.

Furthermore, the Servo interface comprises a mechanism for communicating with the servo drives through **SDO** objects. This communication method allows the user to read or write easily the parameters of a particular servo drive.

The interface includes also a **CAPTURE** mechanism for capturing the positions of different servo drives and external incremental sensors with a higher accuracy than that offered by **Cycle\_Time**. The interface may be employed for index mark synchronizing, higher accuracy at elevated speeds, etc.

## 1.2 SERVO group constituent parts

### General registers

- servo drive general settings and parameters
- servo drive status and operating mode detection and setting
- position acquiring and setting
- error messages

### Position profile generator (PG)

- absolute position control from position to position
- relative position control
- speed control

### GEAR – servo drive synchronization generator

- linear synchronization – linear gearbox simulation
- non-linear synchronization – cancelling or incremental cam simulation

### Command

- functional interface for operating PG and GEAR generators from user programs

### SDO

- group of registers allowing reading and writing of Service Data Objects into particular servo drives

### Capture

- mechanism designed to capture the position of particular servo drives

## 2. Servo drive control and diagnostics

### 2.1 Servo drive control principle

The position of individual servo drives is calculated by means of **PG** and **GEAR** within a single cycle of **Cycle\_Time**. Its size is defined in the file **TGMotion4xx.ini** (250 μs, 500 μs, 1 ms, 2 ms, etc.). **TG Motion** sends the desired positions to the servo drives in every **Cycle\_Time**. The servo amplifier interpolates between two desired positions (i.e., inside a single interval of **Cycle\_Time**). The interpolation process and frequency depend on the particular servo amplifier (refer to the particular servo amplifier manual).

The internal control structure of the servo drive strives to control its current position for the following equation to hold: **Servo.Position** = **Servo.WritePosition** (actual position = desired position). The difference between these values defines the regulatory deviation, which the servo amplifier regulator strives to reduce.

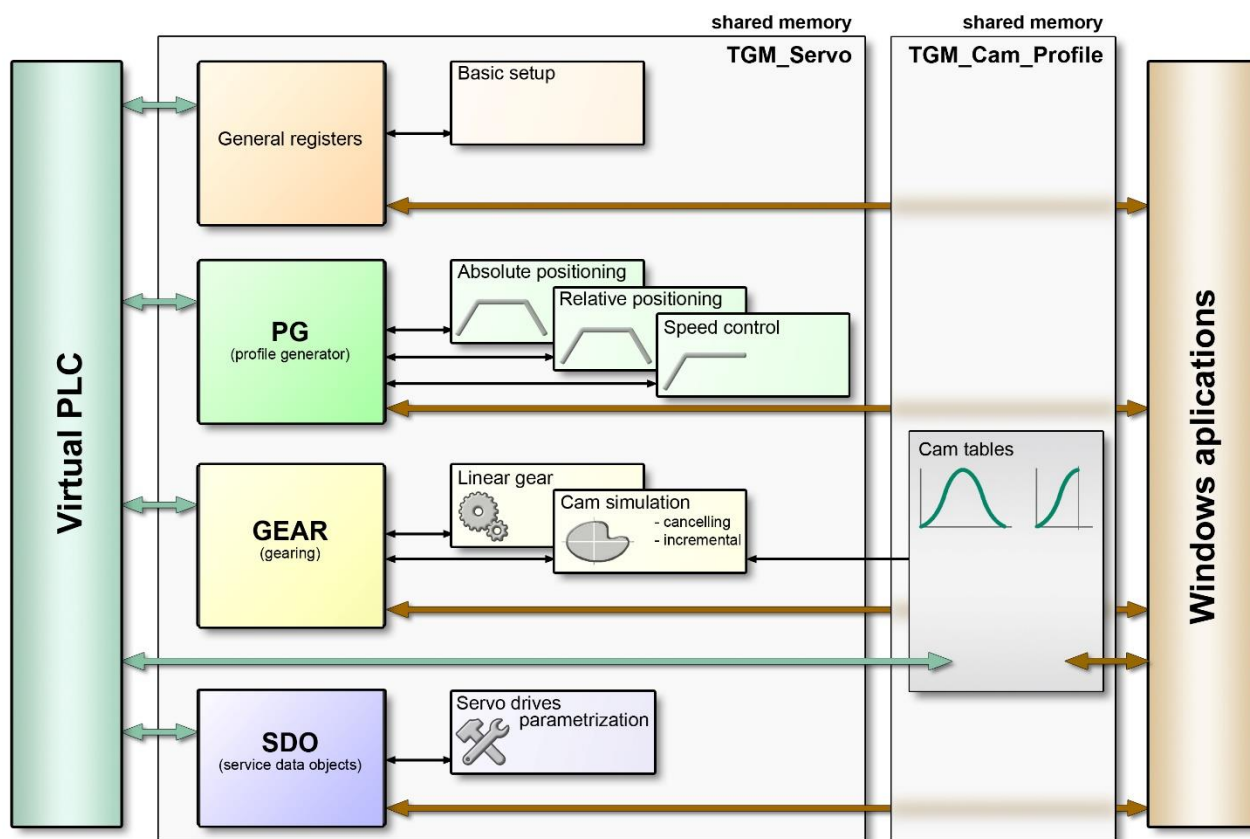


Fig. PLC interface – Servo drive – Windows

## 2.2 Important registers

**Servo.Status** – servo drive current status.

**Servo.Position** – current (actual) position of the servo drive in increments.

**Servo.RefPosition** – current (actual) position of the servo drive shifted by Offset in increments.

**Servo.Offset** – servo drive position shift in increments.

**Servo.Mode** – desired position source:

**0** – the desired position is not modified by **TG Motion** system; the current value of Servo.WritePosition is being sent to the servo drive.

**1** – PG makes the source of the desired value.

**3** – GEAR generator makes the source of the desired value.

**4** – PG and GEAR make the source of the desired value.

**6** – Interpolator makes the source of the desired value.

**7** – PG and Interpolator make the source of the desired value.

**8** – GEAR and Interpolator make the source of the desired value.

**Servo.Error** – error messages – they depend on a particular servo drive.

**Servo.Offset** – servo drive desired position in increments.

**Servo.Correction** – servo drive position correction.

For a complete list of all Servo group registers, including their description, refer to Appendix.

## 2.3 Register principle description and use

Any servo drive can be controlled by means of **Servo.Mode** register, which makes it possible to connect the servo drive to any desired position generator (PG, GEAR, Interpolator) or to a couple of desired position generators, which can be used to create superimposed movements. Furthermore, electric current limitation for the respective servo amplifier can be adjusted by means of **Servo.LimitCurrent**.

To diagnose the drives **Servo.Status** and **Servo.Error** variables are used, which determine the drive current status, and, as the case may be, the error code. The faults can be reset by setting the **Servo.Control** register (for the values see the listing below). Furthermore, servo drive position, speed or current can be monitored.

The **Servo.Correction** register serves as a user correction of the position, which has been calculated by means of PG, GEAR, Interpolator, or their superposition. **TG Motion** does not write any values into this register, but uses it into the calculation of the servo drive desired position. The register may therefore be used to superimpose another movement or a user movement. The calculation of **Servo.Correction** register is to be carried out in **Program\_04**.

### Servo.Mode – value meaning

#### Servo.Mode = 0 (not connected)

The **Servo.WritePosition** register is disconnected from the desired value generator. The user may generate the position desired value in **Program\_04** PLC program.

#### Servo.Mode = 1 (profile generator)

Only the position calculated by **PG** position generator is being sent to the servo drive. In this mode, the **Servo.WritePosition** is calculated using the following formula:

$$\text{Servo.WritePosition} = \text{Servo.Pg.APos} + \text{Servo.Offset} + \text{Servo.Correction}$$

#### Servo.Mode = 3 (GEAR generator)

Only the position calculated by **GEAR** position generator is being sent to the servo drive. In this mode, the **Servo.WritePosition** is calculated using the following formula:

If Servo.GEAR.Mode = 1 (gear mode):

$$\text{Servo.WritePosition} = \text{Servo.GEAR.Position} + \text{Servo.GEAR.Offset} + \text{Servo.Offset} + \text{Servo.Correction}$$

If Servo.GEAR.Mode = 2 (cam mode):

$$\text{Servo.WritePosition} = \text{Servo.GEAR.CamPosition} + \text{Servo.GEAR.Offset} + \text{Servo.Offset} + \text{Servo.Correction}$$

#### **Servo.Mode = 4 (profile generator + GEAR generator)**

A sum of positions calculated by the position **PG** generator and the position **GEAR** generator is sent the servo drive. This mode is used to generate superimposed movements. In this mode, the **Servo.WritePosition** is calculated using the following formula:

If Servo.GEAR.Mode = 1 (gear mode):

$$\text{Servo.WritePosition} = \text{Servo.Pg.APos} + \text{Servo.GEAR.Position} + \text{Servo.GEAR.Offset} + \text{Servo.Offset} + \text{Servo.Correction}$$

If Servo.GEAR.Mode = 2 (cam mode):

$$\text{Servo.WritePosition} = \text{Servo.Pg.APos} + \text{Servo.GEAR.CamPosition} + \text{Servo.GEAR.Offset} + \text{Servo.Offset} + \text{Servo.Correction}$$

#### **Servo.Mode = 6 (Interpolator)**

Only the position calculated by the **Interpolator** position generator is sent to the servo drive. Here, the **Servo.Cnc.Number** register specifies the number of the Interpolator, to which the servo drive is connected, and the **Servo.Cnc.NumberAxes** register determines the Interpolator axis, whose calculated position is being written into **Servo.WritePosition** register. In this mode, the **Servo.WritePosition** is calculated using the following formula:

$$\begin{aligned} \text{Intr} &= \text{Interpolator}[\text{Servo.Cnc.Number}]; \\ \text{AxisIdx} &= \text{Servo.Cnc.NumberAxes}; \end{aligned}$$
$$\text{Servo.WritePosition} = \text{Intr.PositionInc}[\text{AxisIdx}] + \text{Intr.Backlash}[\text{AxisIdx}] + \text{Intr.Offset}[\text{AxisIdx}] + \text{Servo.Offset} + \text{Servo.Correction}$$

#### **Servo.Mode = 7 (Interpolator + profile generator)**

A sum of positions calculated by the **Interpolator** position generator and the position **PG** generator is sent the servo drive. This mode is used to generate superimposed movements. In this mode, the **Servo.WritePosition** is calculated using the following formula:

$$\begin{aligned} \text{Intr} &= \text{Interpolator}[\text{Servo.Cnc.Number}]; \\ \text{AxisIdx} &= \text{Servo.Cnc.NumberAxes}; \end{aligned}$$
$$\text{Servo.WritePosition} = \text{Intr.PositionInc}[\text{AxisIdx}] + \text{Intr.Backlash}[\text{AxisIdx}] + \text{Intr.Offset}[\text{AxisIdx}] + \text{Servo.Pg.APos} + \text{Servo.Offset} + \text{Servo.Correction}$$

#### **Servo.Mode = 8 (Interpolator + GEAR generator)**

A sum of positions calculated by the **Interpolator** position generator and the **GEAR** position generator is sent the servo drive. This mode is used to generate superimposed movements. In this mode, the **Servo.WritePosition** is calculated from the following formula:

If Servo.GEAR.Mode = 1 (gear mode):

$$\begin{aligned} \text{Intr} &= \text{Interpolator}[\text{Servo.Cnc.Number}]; \\ \text{AxisIdx} &= \text{Servo.Cnc.NumberAxes}; \end{aligned}$$
$$\begin{aligned} \text{Servo.WritePosition} &= \text{Intr.PositionInc}[\text{AxisIdx}] + \text{Intr.Backlash}[\text{AxisIdx}] + \\ &\text{Intr.Offset}[\text{AxisIdx}] + \\ &\text{Servo.GEAR.Position} + \text{Servo.GEAR.Offset} + \text{Servo.Offset} + \text{Servo.Correction} \end{aligned}$$

If Servo.GEAR.Mode = 2 (cam mode):

```
Intr = Interpolator[Servo.Cnc.Number];  
AxisIdx = Servo.Cnc.NumberAxes;
```

```
Servo.WritePosition = Intr.PositionInc[AxisIdx] + Intr.Backlash[AxisIdx] +  
Intr.Offset[AxisIdx] +  
Servo.GEAR.CamPosition + Servo.GEAR.Offset + Servo.Offset + Servo.Correction
```

### Servo.Control – value meaning

The register is approached bit-wise. Following items can be controlled by bit settings:

**bit 0** = fault reset.

**bit 1** = servo drive is not under torque.

**bit 2** = EtherCAT communication reset.

**bit 3** = automatic calculation of **Servo.TorqueFeedForward** register is permitted.



*When resetting a fault, the respective bit must be set during the period of one **Cycle\_Time** at least. Bits are not cleared automatically, this step must be ensured by the programmer.*

## 3. Profile Generator of desired position (PG)

### 3.1 Important registers

- PG.APos** – current desired position value.
- PG.ASpeed** – current desired speed value.
- PG.Acc**, **PG.Dec** – drive acceleration, deceleration.
- PG.PosSpeed** – maximum positioning speed.
- PG.DPos** – target position.
- PG.Speed** – desired speed value.
- PG.Mode** – generator mode:
  - 0** – speed control.
  - 1** – position control.
- PG.Rdy** – PG status:
  - 1** – target position has been reached.

For a complete list of all Servo group registers, including their description, refer to Appendix.

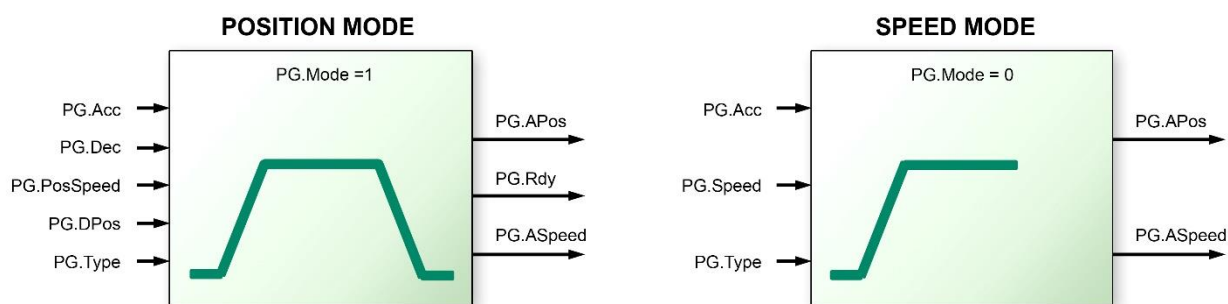


Fig. Position and speed control by PG generator

### 3.2 Description of the PG structure

**PG** (Profile Generator) allows easy positioning from one position to another and the servo drive speed control. At the generator output there are the **PG.APos** variable, which specifies the current desired position value, and the **PG.ASpeed** variable, which specifies the current desired speed value. The generator parameters are **PG.Acc** and **PG.Dec**, which specify the maximum acceleration or deceleration of the drive, and **PG.PosSpeed**, which specifies the maximum speed during positioning, and **PG.DPos**, specifying the target position. The **PG.Speed** parameter specifies the desired speed value in the case of speed control (**PG.Mode = 0**). Control of the generator is implemented by means of **PG.Mode** and **PG.Rdy** variables.

The generator operating mode: during every period of the servo loop and on the basis of **PG.Acc**, **PG.Decel**, **PG.PosSpeed**, **PG.DPos** registers and **PG.Mode** and **PG.Speed** variables, it calculates the **PG.APos** current desired position value and **PG.ASpeed** current desired speed value to write in the shared memory.

The generator operates as virtual one, that is, independently of any particular servo drive. To connect its output to the servo drive, set **Servo.Mode = 1**.

### 3.3 Ramp generator

It defines the movement start and end from the viewpoint of the speed (acceleration and deceleration). It is used to ensure out smooth speed changes in the servo acceleration and deceleration. **TG Motion** offers four basic ramp shapes, which care selectable by the PG register type.

## Pg.Type – value meaning

### 0 – short acceleration, long deceleration:

*WriteSpeed is reached quickly, the deceleration to the desired position is gradual.*

### 1 – short acceleration as well as deceleration – it is used to fast positioning:

*WriteSpeed is reached quickly during the acceleration, the deceleration as well as the desired position achievement is fast.*

### 2 – harmonic acceleration as well as deceleration – this is the most frequently used course:

*It reflects the natural behavior of mechanical components and control systems. All movements are harmonic, respecting inertial forces acting on the servo drives and on related mechanical parts.*

### 3 – linear acceleration as well as deceleration – it is used extremely rarely:

*All of these movement types are linear and can be used for moderate speed changes. It is suited to use on condition that the **PG.Acc**, **PG.Dec** values are not exceeded (see following paragraphs).*

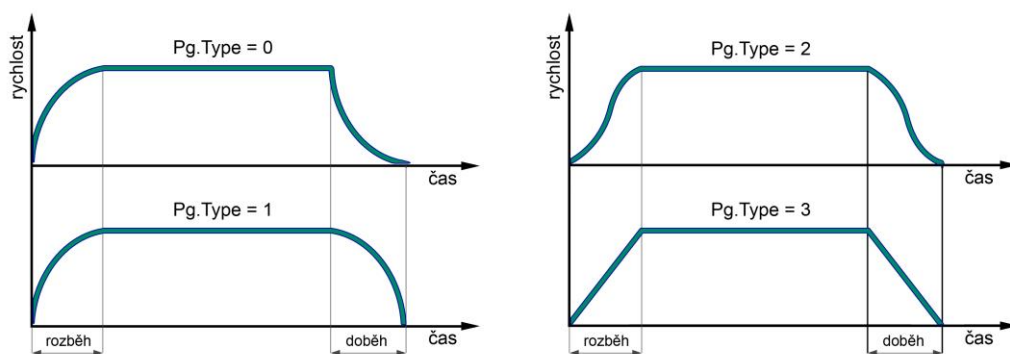


Fig. Generator ramp forms



**TG Motion** always complies with the acceleration and deceleration times of the ramp generator. These times are independent of the ramp form. For example, linear and harmonic movements have the same duration.

The values, which have been set in **PG.Acc**, **PG.Dec** registers, are valid throughout the acceleration and deceleration times for linear movement (**Pg.Type = 3**) only. In the case of harmonic movements the registers contain mean acceleration and deceleration values. For example, the real maximum acceleration/deceleration value amounts to 157 % of the **PG.Acc**, **PG.Dec** register value in the case of **Pg.Type = 2**. One has to count on that for non-linear movements the maximum value of the real acceleration and deceleration reaches the values that exceed those contained in **PG.Acc** a **PG.Dec** registers.

In the case of speed control (**PG.Mode = 0**) **TG Motion** uses only **PG.Acc** register for both the acceleration and deceleration. The value of **PG.Dec** register is not included in the desired position calculation.

## 3.4 Using profile generator

### a) Absolute positioning

It is used to position the servo drive to a particular position.

#### Connecting the position generator to the drive if Mode ≠ 1

Mode = 0

PG.APos = WritePosition – Offset – Correction

Mode = 1

Waiting one period of the Cycle\_Time

*disconnecting the position generator from the drive*

*initialization of output value*

*connecting the generator to the drive*

*pause for connecting (not necessary in Program\_04)*

#### Generator parameter setting

PG.Acc = 100000

PG.Dec = 200000

PG.PosSpeed = 50000

PG.DPos = 35000

PG.Type = 2

*maximum acceleration: 100000 inc/s<sup>2</sup>*

*maximum deceleration: 200000 inc/s<sup>2</sup>*

*maximum speed: 50000 inc/s*

*target position: 35000 increments [inc]*

*ramp type selection*

**Positioning start**

 PG.Rdy = 0  
 PG.Mode = 1

*positioning ready flag reset  
 generator start*
**Positioning test**

Waiting for PG.Rdy = 1

*waiting until target position is reached*
**Position interruption and stop**

 PG.Mode = 0  
 PG.Speed = 0  
 Waiting for PG.ASpeed = 0

*position stop  
 movement stop  
 stop test*
**Target position change while running**

 PG.Mode = 0  
 Waiting for 1 Cycle\_Time period  
 PG.DPos = new position  
 PG.Mode = 1

*position stop  
 pause for PG disconnecting  
 write a new position  
 generator start*
**b) Relative positioning**

It is used to displace the servo drive by a certain number of increments in the desired direction.

**Connecting the position generator to the drive if Mode ≠ 1**

 Mode = 0  
 PG.APos = WritePosition – Offset – Correction  
 Mode = 1  
 Waiting for one period of the Cycle\_Time

*disconnecting the position generator from the drive  
 initialization of output value  
 connecting the generator to the drive  
 pause for connecting (not necessary in Program\_04)*
**Generator parameter setting**

 PG.Acc = 100000  
 PG.Dec = 200000  
 PG.PosSpeed = 50000  
 PG.DPos = PG.APos + 10000  
 PG.Type = 1

*maximum acceleration: 100000 inc/s<sup>2</sup>  
 maximum deceleration: 200000 inc/s<sup>2</sup>  
 maximum speed: 50000 inc/s  
 target position = current position + 10000  
 ramp selection*
**Positioning start**

 PG.Rdy = 0  
 PG.Mode = 1

*positioning ready flag reset  
 generator start*
**Positioning test**

Waiting for PG.Rdy = 1

*waiting until target position is reached*
**Position interruption and stop**

 PG.Mode = 0  
 PG.Speed = 0  
 Waiting for PG.ASpeed = 0

*position stop  
 movement stop  
 stop test*
**c) Speed control**

It is used to set the speed of rotation or movement regardless of the end position.

**Connecting the position generator to the drive if Mode is not equal to 1**

 Mode = 0  
 PG.APos = WritePosition – Offset – Correction  
 Mode = 1  
 Waiting for one Cycle\_Time period

*disconnecting the position generator from the drive  
 initialization of output value  
 connecting the generator to the drive  
 pause for connecting*
**Generator parameter setting**

 PG.Acc = 100000  
 PG.PosSpeed = 20000

*maximum acceleration 100000 inc/s<sup>2</sup>  
 setting the speed to 20000 inc/s*

**Speed change**

PG.Speed = -20000

*speed change to -20000 inc/s***Stop**

PG.Speed = 0

Waiting for PG.ASpeed = 0

*movement stop**stop test***d) Reference setting**

It is used to set the reference position.

**Disconnecting the position generator from the drive**

Mode = 0

Waiting for one Cycle\_Time period

*PG generator is disconnected**pause for disconnecting***Reference setting, current position = 100000 inc**

Offset = WritePosition - 100000

*reference setting*

## 4. GEAR Generator of desired position

### 4.1 Description of the structure GEAR

The **GEAR** generator makes it possible to synchronize a drive with another arbitrary drive in a simple way, where the master position source may be a desired position, an actual position, a servo tick (time synchronization) or a position outputted by an external incremental position sensor. The synchronization can be linear (gearbox simulation) or non-linear (cam simulation). The cam simulation can be cancelling or incremental. The cam as well as the linear gear output position can be angularly shifted with adjustable speed.

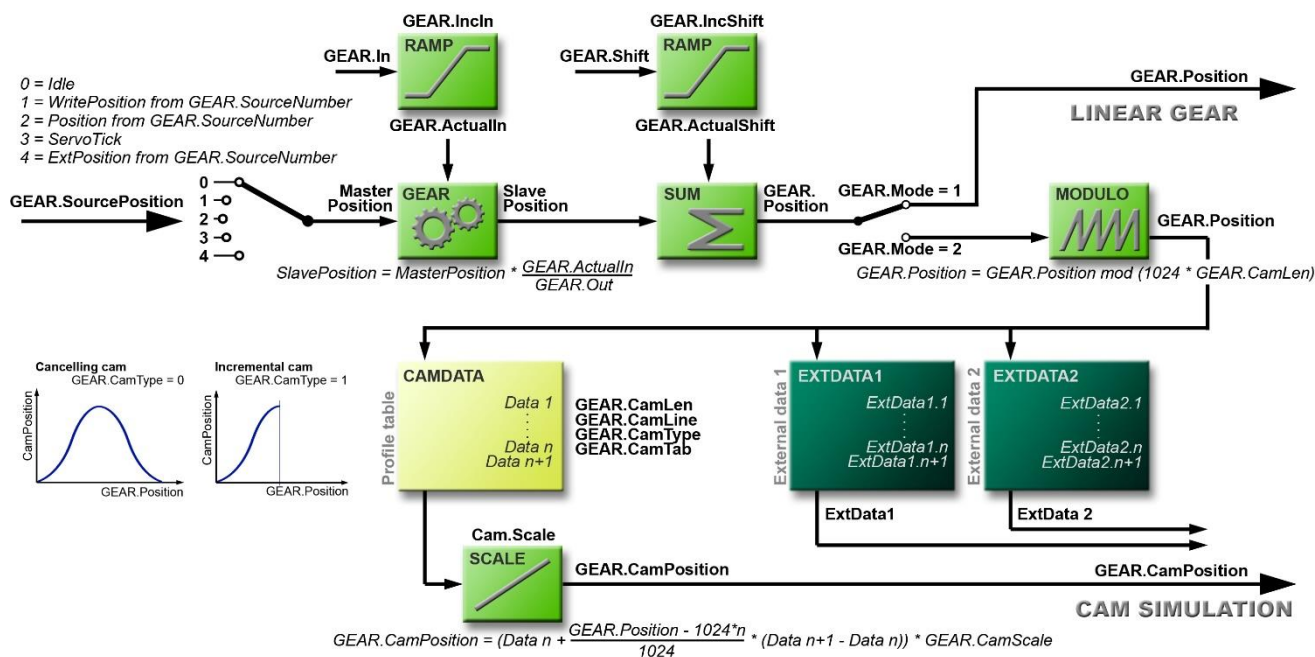


Fig. Block diagram of the GEAR generator of desired position

### 4.2 Important registers

**GEAR.Mode** – specifies the mode of GEAR generator:

- 0 – idle mode.
- 1 – linear gear mode (**TG Motion** calculates GEAR.Position).
- 2 – cam simulation mode (**TG Motion** calculates both GEAR.Position and GEAR.CamPosition).

**GEAR.Position** – current desired position in linear synchronization mode (gearbox) or  
 – cam angle in non-linear synchronization mode.

**GEAR.CamPosition** – current desired position in non-linear synchronization mode.

**GEAR.SourceNumber** – master position source.

**GEAR.SourcePosition** – master position type.

**GEAR.In** a **GEAR.Out** – gear ratio.

**GEAR.CamTab** – specifies the shared memory, where cam profile data are saved:

- 0 – TGM\_Cam\_Profile.
- 1 – TGM\_Data.

**GEAR.CamLine** – cam profile data beginning.

**GEAR.CamLen** – cam profile data length.

**GEAR.CamType** – cam type.

**GEAR.Shift** – angular shift of calculated position SlavePosition in increments.

**GEAR.IncShift** – change of this angular shift [inc/servo tick].

For a complete list of all Servo group registers, including their description, refer to Appendix.

At the generator output, there is a **GEAR.Position**, which specifies the current desired position value in the linear synchronization mode and the cam angle in the non-linear synchronization mode. The **GEAR.SourceNumber** and **GEAR.SourcePosition** registers specify the source and type of the master position for synchronization.

#### GEAR.SourcePosition – register values and their meaning

- 0 – gear is inactive.
- 1 – WritePosition of the master servo drive is the source of the position.
- 2 – Position of the master servo drive is the source of the position.
- 3 – Time (one servo tick per Cycle\_Time).is the source of the position.
- 4 – ExtPosition of the master servo drive is the source of the position.

The gear ratio is set by means of two registers, namely **GEAR.In** (the numerator) and **GEAR.Out** (the denominator) of the gear ratio. **GEAR.Incln** register, which has been designed to simulate the clutch, may also intervene into the servo desired position calculation. It is used to smooth linking to the master servo drive, which is already running. **GEAR.Shift** register determines the phase shift of the output position of **GEAR.Position** register, **GEAR.IncShift** determines the change in the phase shift until the shift, which is determined by **GEAR.Shift** register, is reached.

When choosing between the linear gear (gearbox) and non-linear gear (cam), **GEAR.Mode** register is used.

#### GEAR.Mode – register values and their meaning

- 1 – linear gear simulation – the **GEAR.Position** register makes the source of the desired position.
- 2 – cam simulation – the **GEAR.Position** value is used to calculate the cam data table index.

The cam simulation code operates with non-linear synchronization data, which constitute the table of the cam profile. The table for the cam profile is saved either in the **TGM\_Cam\_Profile** shared memory (size 1048576 bytes), or in the **TGM\_Data** shared memory (size 524288 bytes). The **GEAR.CamTab** register may be set to allow toggling between the tables in the memory areas. The same register determines the location of both tables of External data.

#### GEAR.CamTab – register values and their meaning

- 0 – data are read from **TGM\_Cam\_Profile**.
- 1 – data are read from **TGM\_Data**.

Windows application (for example, a visualization program) or PLC is taking care of filling the cam profile table. The **GEAR.CamLine** and **GEAR.CamLen** registers determine the beginning and data number in the cam profile.

In the non-linear synchronization mode, the **GEAR.CamPosition** register, which determines the current desired value of the servo drive position, makes the output of the GEAR generator of position. In the MODULO section and by means of the **GEAR.Position** register, an index aiming at the cam table is calculated, which is 1024-times finer than the table data, which means that one table step is divided into 1024 parts. It holds that a change of the **GEAR.Position** by 1024 increments results in a shift by one value in the cam profile table. The resulting value of the servo drive position (**GEAR.CamPosition**) is calculated using the linear interpolation between two neighboring table values.

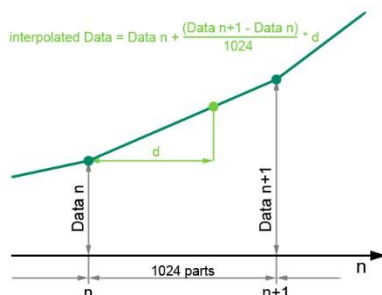


Fig. Linear interpolation between the data of CAMDATA table

### 4.3 Cancelling cam

It is used to carry out a periodical non-linear movement along a closed curve – simulating the cam on a shaft. **TG Motion** interpolates linearly between two neighboring table values by dividing the virtual interval into 1024 parts. When returning to the table beginning, it interpolates between the first and the last data of the table. It is therefore advisable to start the table with a non-zero value and end the data sequence by a zero value.

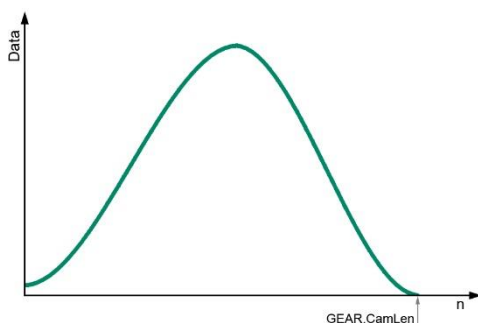


Fig. Cancelling cam

### 4.4 Incremental cam

It is used to carry out a non-linear movement, where the starting point of the next cycle inks smoothly to the ending point of the previous one. Here, **TG Motion** interpolates linearly between two neighboring table values by dividing the virtual interval into 1024 parts. At the table beginning, it is interpolated between **0** explicit value and the first value of the data table. When the end of the table is reached, the last data are taken as an increment (stroke) of the incremental cam, which is added – while next passing through the table – to the interpolated data of the incremental cam, so as to attain a smooth link in the servo drive movement. It is therefore advisable to start the table with a non-zero value.

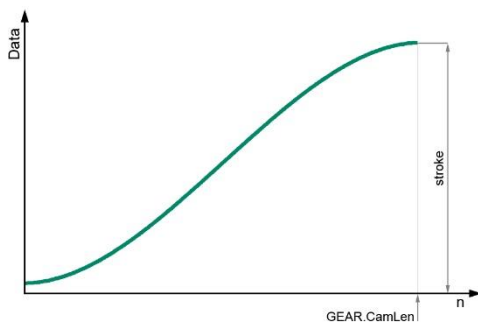


Fig. Incremental cam

## 4.5 External data table

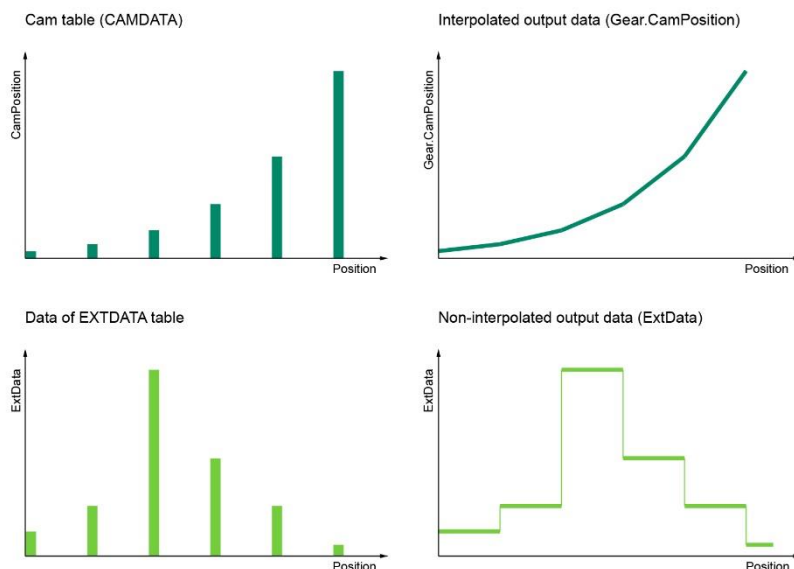
EXTDATA1 and EXTDATA2 independent data tables can be used in the case of several synchronized non-linear movements. They can also be used to control other peripheral devices, which must be synchronous with the gear, but their shapes differ from those of the cam. The same memory index as that of CAMDATA memory can be used to calculate their outputs, i.e., **ExtData1** and **ExtData2**. Subsequently, the output values are written to the Data memory at the offsets, which are specified by **CamExpAddress1** and **CamExpAddress2** registers. The sizes of CAMDATA, EXTDATA1 and EXTDATA2 do not have to be the same. The location of the tables in the shared memory is specified by the **GEAR.CamTab** register (it is the same register that specifies the location of the cam profile data table).

### GEAR.CamTab – register values and their meaning

- 0 – data are read from **TGM\_Cam\_Profile**.
- 1 – data are read from **TGM\_Data**.

Windows application (for example, a visualization program) or PLC is taking care of filling the cam profile table.

External table data are approached in the same way as the cam data table. However, when calculating the **ExtData1** and **ExtData2** registers, no interpolation between the neighboring data of the external tables is carried out. All the time, when the interpolation takes place in CAMDATA table, the last loaded value of EXTDATA1 and EXTDATA2 tables is used to calculate the **ExtData1** and **ExtData2** value. After further data in the table are reached (when the index is divisible by 1024 without rest), the use of new loaded data will start.



*Fig. Interpolated data of the cam table and non-interpolated data of external tables*

## 4.6 Use of GEAR generator

### a) Linear gear

It serves for linear synchronization of two servo drives.

#### Gear parameter setting

GEAR.SourceNumber = 3	<i>synchronization on servo drive No. 3</i>
GEAR.SourcePosition = 1	<i>desired position value of servo drive No 3 makes the position source</i>
GEAR.ActualIn = 0	<i>reset of current value of the gear (numerator)</i>
GEAR.IncIn = 10	<i>setting the increment of the loop gear</i>
GEAR.In = -1000	<i>specifies the desired value of the gear (numerator)</i>
GEAR.Out = 2000	<i>specifies the gear ratio (denominator)</i>
GEAR.Position = 0	<i>reset of the generator output position</i>
GEAR.Shift = 0	<i>reset of the desired value of the output position shift</i>
GEAR.ActualShift = 0	<i>reset of the actual value of the output position shift</i>

#### Starting up the generator

GEAR.Mode = 1	<i>connecting the position generator to the drive</i>
GEAR.Offset = WritePosition - Offset - GEAR.Position	<i>offset initialization</i>
Mode = 3	<i>connecting the generator to the drive</i>

#### Changing the gear while running

GEAR.IncIn = 10	<i>dynamics of the gear change</i>
GEAR.In = GEAR.In + 6000	<i>desired change of the gear</i>
Waiting for GEAR.ActualIn = GEAR.In	<i>test of the gear change completion</i>

#### Angular shift

GEAR.IncShift = 10	<i>shift dynamics</i>
GEAR.Shift = 2000	<i>status of shift by 2000 increments</i>
Waiting for GEAR.ActualShift = GEAR.Shift	<i>test of the shift completion</i>

#### Desynchronization

GEAR.IncIn = 40	<i>dynamics of the gear change</i>
GEAR.In = 0	<i>gear reset</i>
Waiting for GEAR.ActualIn = GEAR.In	<i>test of the desynchronization completion</i>
Mode = 0	<i>disconnecting the position generator from the drive</i>
GEAR.Mode = 0	<i>stopping the position generator</i>

### b) Non-linear gear

Serves for non-linear synchronization of two servo drives (cam simulation).

#### Gear parameter setting

GEAR.SourceNumber = 2	<i>synchronization on servo drive No. 2</i>
GEAR.SourcePosition = 4	<i>external encoder value makes the source of the position (e.g. IRC of servo drive No.2)</i>
GEAR.ActualIn = 0	<i>reset of current value of the gear (numerator)</i>
GEAR.IncIn = 10	<i>setting the increment of the gear – numerator – in every period of the Cycle_Time</i>
GEAR.In = 0	<i>reset of the desired value of the gear (numerator)</i>
GEAR.Out = 2000	<i>specifies the gear (denominator)</i>
GEAR.Position = 0	<i>reset of the generator output position</i>
GEAR.Shift = 0	<i>reset of the desired value of the output value shift</i>
GEAR.ActualShift = 0	<i>reset of the actual value of the output position shift</i>
GEAR.CamType = 0	<i>cancelling cam</i>
GEAR.CamScale = 1	<i>scale of data from cam table (1 corresponds to a 1:1 scale)</i>



## 5. COMMAND functional interface

### 5.1 Important registers

**Command.Control** – the function in question is executed by writing its number in this register.

**Command\_Par [0–14]** – parameters of each function of the **Command.Control** register.

For a complete list of all Servo group registers, including their description, refer to Appendix.

### 5.2 Description of the structure Command

**COMMAND** functional interface simplifies the application of PG and GEAR generators. Thanks to this simplification, the system by itself checks the time synchronization of the relevant register writing with the servo loop cycle, thus avoiding the use of a dwell delay when handling the position generators. The resulting PLC code is more economical in the number of instructions used and its execution is faster.

The interface is realized in general. The user calls the required function by writing the command value into the **Command.Control** register. As many as 15 parameters can be set, which are written in the variables of **Command.Par [0–14]**. The system subsequently signals the function execution by setting the **Command.Control** register to zero.

#### Command.Control – values and meaning of transferred parameters

##### **Command.Control = 0 – the previous function has been finished**

Having accepted the function, **TG Motion** sets the **Command.Control** register value to zero and is ready to execute the next function. The only exception is the deactivation of GEAR generator (**Command.Control = 7**, see below).

##### **Command.Control = 1 – PG – speed control**

Command.Par [0] = Acc	<i>desired acceleration</i>
Command.Par [2] = Speed	<i>desired speed</i>
Command.Par [4] = Enable_GEAR	<i>behavior towards the active GEAR drive:</i>
	<i>0 – if GEAR is active, it will be cancelled and PG will be activated</i>
	<i>1 – PG will be superimposed on GEAR</i>
Command.Par [5] = Type	<i>ramp type (0–3)</i>

##### **Command.Control = 2 – PG – absolute positioning**

Command.Par [0] = Acc	<i>desired acceleration</i>
Command.Par [1] = Dec	<i>desired deceleration</i>
Command.Par [2] = Speed	<i>desired speed</i>
Command.Par [3] = Pos	<i>absolute desired position</i>
Command.Par [4] = Enable_GEAR	<i>behavior towards the active GEAR drive:</i>
	<i>0 – if GEAR is active, it will be cancelled and PG will be activated</i>
	<i>1 – PG will be superimposed on GEAR</i>
Command.Par [5] = Type	<i>ramp type (0–3)</i>

**Command.Control = 3 – PG – relative positioning**

Command.Par [0] = Acc	<i>desired acceleration</i>
Command.Par [1] = Dec	<i>desired deceleration</i>
Command.Par [2] = Speed	<i>desired speed</i>
Command.Par [3] = Pos	<i>relative position change – by how many increments and in what direction the current servo position is to change</i>
Command.Par [4] = Enable_GEAR	<i>behavior towards the active GEAR drive: 0 – if GEAR is active, it will be cancelled and PG will be activated 1 – PG will be superimposed on GEAR</i>
Command.Par [5] = Type	<i>ramp type (0–3)</i>

**Command.Control = 4 – PG – reference setting**

Command.Par [3] = Ref	<i>the current position is declared to be the position specified by this parameter</i>
-----------------------	--

**Command.Control = 5 – PG – cancelling any positioning and stopping the servo drive**

Command.Par [1] = Dec	<i>desired deceleration</i>
Command.Par [5] = Type	<i>ramp type (0–3)</i>



To activate the GEAR generator, use **Command.Control = 6**. The activation of a **linear gear** differs from that of a **cam** in the **Command.Par [2]** value and in the setting of relevant parameters.

**Command.Control = 6 – GEAR – linear gear**


Command.Par [0] = SourceNumber	<i>number of the master servo drive</i>
Command.Par [1] = SourcePosition	<i>synchronization source</i>
Command.Par [2] = 1	<i>linear gear</i>
Command.Par [3] = GIn	<i>gear ratio (numerator)</i>
Command.Par [4] = GOut	<i>gear ratio (denominator)</i>
Command.Par [5] = GInInc	<i>dynamics of the gear change</i>
Command.Par [9] = GEAR.IniPosition	<i>initialization value of GEAR.Position at the GEAR turn-on moment</i>
Command.Par [12] = ServoMode	<i>setting the desired position source</i>

**Command.Control = 6 – GEAR – cam**

Command.Par [0] = SourceNumber	<i>number of the master servo drive</i>
Command.Par [1] = SourcePosition	<i>synchronization source</i>
Command.Par [2] = 2 + CamType	<i>cam</i>
Command.Par [3] = GIn	<i>gear ratio (numerator)</i>
Command.Par [4] = GOut	<i>gear ratio (denominator)</i>
Command.Par [5] = GInInc	<i>dynamics of the gear ratio change</i>
Command.Par [9] = GEAR.IniPosition	<i>initialization value of GEAR.Position at the GEAR turn-on moment</i>
Command.Par [12] = ServoMode	<i>setting the desired position source</i>
Command.Par [6] = CamLine	<i>beginning of cam data</i>
Command.Par [7] = CamLen	<i>length of cam data</i>
Command.Par [10] = CamTab	<i>localization of cam data memory (TGM_Cam_Profile, TGM_Data)</i>
Command.Par [8] = CamScale	<i>scale of cam output value (coefficient)</i>
Command.Par [13] = CamIncPosition	<i>explicitly entered stroke of the incremental cam for avoiding the error of the calculated stroke resulting from real value (double) to integer conversion (for incremental cam only)</i>

**Command.Control = 7 – GEAR – deactivation of GEAR generator**

Command.Par [5] = GInInc                      *a positive number as a linear decrement of the gear ratio numerator is entered – the servo drive starts braking*

 For **Command.Control = 7** (GEAR generation deactivation), the **Command.Control** register will be set to zero until the gear ratio is synchronized, i.e., until the whole ramp is carried out.

## 5.3 Use of Command functional interface

### a) Absolute positioning start

It is used to position the servo drive to a particular position.

Command_Par [0] = 1000000	<i>desired acceleration [inc/s<sup>2</sup>]</i>
Command_Par [1] = 1000000	<i>desired deceleration [inc/s<sup>2</sup>]</i>
Command_Par [2] = 300000	<i>desired speed [inc/s]</i>
Command_Par [3] = 200000	<i>desired position [inc]</i>
Command.Control = 2	<i>absolute positioning start</i>

<i>Waiting for Command.Control = 0</i>	<i>waiting for function execution</i>
<i>Waiting for PG.Rdy = 1</i>	<i>waiting for positioning</i>

### b) Synchronization start – non-linear gear (cam)

Serves to non-linear synchronization of two servo drives (cam simulation).

Command_Par [0] = 10	<i>number of the master servo drive</i>
Command_Par [1] = 2	<i>the Position master register of the servo drive makes the source of the position</i>
Command_Par [2] = 3	<i>incremental cam simulation mode</i>
Command_Par [3] = 32000	<i>numerator of the synchronization gear ratio</i>
Command_Par [4] = 32767	<i>denominator of the synchronization gear ratio</i>
Command_Par [5] = 10	<i>increment (decrement) of the numerator in the synchronization gear ratio during the synchronization of the gear ratio within one servo cycle (Cycle_Time)</i>
Command_Par [6] = 0	<i>address of the cam data profile in the memory</i>
Command_Par [7] = 100	<i>number of data in the cam profile</i>
Command_Par [8] = 1.0	<i>scale of cam table data conversion 1:1 (coefficient)</i>
Command_Par [9] = 1024	<i>initialization value of the cam table index</i>
Command_Par [10] = 1	<i>cam table in <b>TG Motion</b> shared memory</i>
Command = 6	<i>start of synchronization</i>


<i>Waiting for Command.Control = 0</i>	<i>waiting for function execution</i>
--	---------------------------------------

### c) Desynchronization start

Is used to disconnect the synchronized servo drive from the master position source.

Command_Par [5] = 20	<i>increment of the numerator in the synchronization gear ratio during the synchronization of the gear ratio within one servo cycle</i>
Command.Control = 7	<i>start of desynchronization</i>

<i>Waiting for Command.Control = 0</i>	<i>waiting for function execution</i>
--	---------------------------------------

 It is to be ensured in the PLC program that the **synchronization start** is not called repeatedly. If a new synchronization is intended, a desynchronization must be carried out first and only then the synchronization function can be called. A similar rule applies to the **desynchronization start** function.



*It holds for all Command types that **TG Motion** sets to zero the **Command.Control** register already after accepting a function, after its execution started. The function itself has not to be completed.*

*Only in the case of the **desynchronization start (Command.Control = 7)** the **Command.Control** register is zeroed after the gear ratio synchronization, after the whole ramp is completed.*

## 6. Interface for SDO communication

### 6.1 Description of the structure SDO

The structure, which uses Service Data Objects (SDO), is employed to parameterize particular servo amplifiers. Using the service manual of the particular servo amplifier, the user finds out the address (index and subindex) of the necessary parameter (object) and its size in bytes. Having done this, he can write the necessary value in the respective register or read the actual parameter value.

### 6.2 Important registers

**SDO.Control** – write and read control.

**SDO.Status** – determines the write, read or error status.

**SDO.Index** – index of a service data object.

**SDO.SubIndex** – subindex of a service data object.

**SDO.Data** – data to write or loaded data; in the case of error (SDO.Status = 2), error code.

For a complete list of all Servo group registers, including their description, refer to Appendix.

### 6.3 Registers: description and how to work with them

The interface is used write and read any parameters of a particular servo drive through the communication of SDO. The communication control is enabled by means of **SDO.Control** and **SDO.Status** registers. The parameters are addressed by means of **SDO.Index** and **SDO.SubIndex** registers. The values of the variables are servo drive type specific. The table showing the parameter assignment is provided in the respective servo drive manual.

### 6.4 Use of the SDO structure

#### a) Parameter writing

##### Writing start

SDO.NumberByte = 4

SDO.Index = 24672

SDO.SubIndex = 1

SDO.Data = 654

SDO.Control = 1

*number of bytes in given parameter (1–4)*

*index of given parameter*

*subindex of given parameter*

*data to be written*

*write request*

*Waiting for SDO.Control = 0*

*waiting for writing end*

##### Writing correctness test

*When SDO.Status = 0*

*When SDO.Status = 2*

*write process has been finished*

*write error – the error code is in the SDO.Data register*

**b) Parameter reading****Reading start**

SDO.NumberByte = 2

SDO.Index = 24672

SDO.SubIndex = 0

SDO.Control = 2

*number of bytes in given parameter (1–4)**index of given parameter**subindex of given parameter**read request*

Waiting for SDO Control = 0

*waiting for reading end***Reading correctness test**

When SDO.Status = 0

Then Data = SDO.Data

*reading finished**read data in SDO.Data*

When SDO.Status = 2

Then Error = SDO.Data

*reading error**error code is stored to SDO.Data register*

## 7. Capture interface

### 7.1 Description of the structure Capture

The Capture interface is used to record events with higher accuracy than that offered by Cycle\_Time. The captured event contains the exact position of the servo just at the event time. At the same time, other values are recorded and written in the respective registers. The servo amplifier is watching over whether the condition is met based on the parameter setting. Subsequently, **TG Motion** will read the event in the nearest Cycle\_Time.

### 7.2 Important registers

**CAPTURE.Control** – position capture control:

0 – capture is disabled.

1 – capture of external digital input is allowed (for AKD and ServoStar servo amplifiers only).

2 – one-shot capture of registers by **TG Motion** system is allowed.

**CAPTURE.Typ\_ExtCapture** – number of digital input and the edge type with which the capture will be synchronized:

bit 0 – capture by rising edge of digital input 1 is allowed.

bit 1 – capture by falling edge of digital input 1 is allowed.

bit 2 – capture by rising edge of digital input 2 is allowed.

bit 3 – capture by falling edge of digital input 2 is allowed.

bit 4 – capture by encoder zero pulse is allowed.

**CAPTURE.PPosition** – actual position of the servo drive at the digital input rising edge.

**CAPTURE.NPosition** – actual position of the servo drive at the digital input falling edge.

**CAPTURE.PExt\_Position** – position value of the external position encoder at the rising edge of the digital input.

**CAPTURE.NExt\_Position** – position value of the external position sensor at the falling edge of the digital input.

**CAPTURE.Write\_Position** – desired servo drive position captured.

**CAPTURE.Offset** – address offset of the variable captured in **CAPTURE.Data**.

**CAPTURE.DData** – captured data of the shared memory of **TGM\_Data** from the **Offset** as double (8 bytes).

**CAPTURE.Zero\_Position** – servo drive actual position at the arrival of the zero pulse of external incremental sensor (**CAPTURE.Typ\_ExtCapture** – bit 4).

For a complete list of all Servo group registers, including their description, refer to Appendix.

### 7.3 Description of the structure Capture

The CAPTURE interface is able to capture – at the arrival of an external digital signal – the current position of the servo drive and the position of an external sensor with a time tolerance less than 100µs. The external digital signal is fed directly to the servo amplifier digital input.

Furthermore, the CAPTURE interface is able to capture – within one Cycle\_Time – the servo drive desired position and any shared memory register.

After the **CAPTURE.Control** register is set to zero value, values will be loaded and saved to respective registers. Subsequently, the **CAPTURE.Control** register is set to zero to inform that **TG Motion** finished the capturing. Desired values can then be read.

### 7.4 Use of Capture interface

#### Capturing by an external digital signal

**CAPTURE.Control** = 1

Wait for **CAPTURE.Control** = 0

Read of captured position

*capturing is allowed*

*testing whether a capture has finished*

## 8. Appendix

### List and description of Servo group registers

#### General register

name	access	offset	description
Number	R	0	number of the mapped servo drive, logical number of the interface (Servo 0 – Number = 0, Servo 1 – Number = 1, etc.)
Node	R	4	address of the physical servo drive, which has been set on the servo amplifier
Axe	R	8	axis index on the given servo amplifier (in the case of multi-axis servo amplifiers)
Type	R	12	type of the servo connected to the servo amplifier (provided in <b>TGMotion4xx.ini</b> file): 0 = no servo drive connected 1 = ServoStar 700 servo drive connected 2 = TGA servo drive connected 21 = TGA (firmware 3.00 and higher) servo drives connected 3 = TGA300 servo drive connected 4 = TGP servo drive connected 41 = TGP servo drive with external position sensor connected 42 = TGP s servo drive with 32 bit control word connected 43 = TGP servo drive connected with external position sensor and 32 bit control word connected 5 = AKD servo drive connected 6 = TGA220 servo drive connected 7 = PIERCE retrofit servo drive connected 8 = DCM servo drive connected 9 = TGZ servo drive connected
Resolution	R	16	number of increments per servo motor turn [inc]
SerialNumber	R	20	servo amplifier serial number
Control	RW	24	bit 0 = fault reset bit 1 = is not under torque bit 2 = EtherCAT communication reset bit 3 = allows automatic calculation of TorqueFeedForward register
Status	R	28	specifies the actual status of the servo drive: -1 = does not communicate after previous communication 0 = does not communicate, communication could not be established 1 = servo drive fault 2 = ready without torque 3 = ready under torque
Mode	RW	32	specifies the desired position source: 0 = desired value disconnected (current position is being sent to the servo) 1 = position generator (PG) makes the desired position source 3 = GEAR generator makes the desired position source 4 = PG and GEAR generator make the desired position source 6 = INTERPOLATOR makes the desired position source. 7 = PG and INTERPOLATOR make the desired value source 8 = GEAR generator and INTERPOLATOR make the desired value source
Error	R	36	specifies the code of the first captured fault, if the drive is in fault; the value meaning depends on the particular type of the servo drive
EtherCATState	R	40	status of the servo drive communication from EtherCAT viewpoint: 0x01 = Init 0x02 = Pre-Operational 0x04 = Safe-Operational 0x08 = Operational
DigitalIn	R	44	digital input value
DigitalOut	RW	48	digital output value; digital outputs may be set and reset by means of bit setting and resetting
TorqueFeedForward	RW	52	torque value sent to servo drive; The function is supported for AKD a TGZ servo amplifiers: AKD: range -3000 to +3000, where 3000 = servo amplifier peak current TGZ: range -32767 to +32767, where 32767 = servo amplifier peak current
ControlWord	R	56	copy of CAN control word
StatusWord	R	60	copy of CAN status word
SysTimeDifference	R	64	last known difference between local time and reference time [ns]
WritePosition	RW	464	specifies the desired position value, which is sent to the servo drive [inc]
Position	R	472	servo drive actual position [inc]
RefPosition	R	480	specifies the actual position shifted by Offset (Position + Offset) [inc]
ExtPosition	R	488	position read from external position sensor of the servo amplifier
Offset	RW	496	allows to shift the motor position in increments [inc]
Correction	RW	504	correction value added to WritePosition if Mode ≠ 0; <b>TG Motion</b> does not write to this register [inc]

name	access	offset	description
PositionError	R	512	difference between the position written value and the real position read from the servo drive (WritePosition – Position) [inc]
WriteSpeed	R	520	calculated desired speed (derivative of WritePosition register) [inc/s]
Speed	R	528	calculated actual speed (derivative of Position register) [inc/s]
LimitCurrent	RW	536	servo amplifier current limitation; the value depends on the particular type of the servo drive
Current	R	544	actual current; the value depends on the particular type of the servo drive
MaxCurrent	R	552	specifies the servo amplifier maximum current (the register is not supported for all servo amplifier types)
AnalogIn	R	560	servo amplifier analog input value; units and range depend on the particular type of the servo amplifier
WriteAcc	R	568	calculated desired acceleration (derivative of WriteSpeed) [inc/s <sup>2</sup> ]
AccMaxTorqueFeedForward	RW	576	scale – when the bit 3 of Control register is set, the TorqueFeedForward is calculated from the values of WriteAcc and AccMaxTorqueFeedForward

### Profile generator (PG) registers

name	access	offset	description
Acc	RW	1296	acceleration desired value [inc/s <sup>2</sup> ]
Dec	RW	1304	deceleration desired value [inc/s <sup>2</sup> ]
APos	RW	1312	actual position [inc]
DPos	RW	1320	target position [inc]
ASpeed	R	1328	actual speed calculated by PG [inc/s]
PosSpeed	RW	1336	maximum speed in position control mode [inc/s]
Speed	RW	1344	desired speed in speed control mode [inc/s]
Mode	RW	1432	modes of PG: 0 = speed control or user-forced termination of position control 1 = position control (to be set by the user) 3 = braking phase in progress (ramp generator deceleration)
Rdy	RW	1436	PG status: 1 = target position is reached
Type	RW	1440	type of ramp, which controls the speed changes (0–3)

### SDO object registers

name	access	offset	description
Control	RW	1496	SDO write and read control: 0 = communication finished 1 = writing request 2 = reading request
Status	R	1500	communication progress or a message whether the communication was successful: 0 = communication was successful 1 = communication is in progress 2 = communication error
NumberByte	W	1504	SDO data size (bytes)
Index	W	1508	address from where to read the data or where to write the data (bytes)
SubIndex	W	1512	subaddress from where to read the data or where to write the data (bytes)
Data	RW	1516	data to write or having been read, or error code

**GEAR generator registers**

name	access	offset	description
Position	RW	1520	current desired value in linear synchronization mode, or cam angle in non-linear synchronization mode [inc]
CamPosition	RW	1528	current desired value in non-linear synchronization mode [inc]
Offset	RW	1536	output position offset [inc]
Shift	RW	1544	angular shift of the calculated SlavePosition value [inc]
ActualShift	R	1552	actual value of the output position displacement [inc]
IncShift	RW	1560	change of SlavePosition angular shift [inc/servotick]
Incln	RW	1568	GEAR ratio change time rate [inc/servotick]
CamIncPosition	RW	1576	incremental cam angular shift [inc]
CamScale	RW	1584	cam data scale – coefficient to multiply the cam data
SourceNumber	RW	1656	number of the mapped drive, which serves as a source of the position
SourcePosition	RW	1660	type of master position: 0 = GEAR inactive 1 = WritePosition of master servo drive (desired position) 2 = Position of master servo drive (actual value) 3 = ServoTick of master servo drive 4 = ExtPosition of master servo drive (current position from external encoder of master servo drive)
Mode	RW	1664	GEAR generator modes: 0 = GEAR inactive 1 = linear gear mode 2 = cam simulation mode
In	RW	1668	desired value of the gear ratio numerator
Out	RW	1672	gear ratio denominator
ActualIn	R	1676	actual gear ratio numerator
CamLine	RW	1680	offset of the cam profile data beginning [bytes]
CamLen	RW	1684	cam data number – cam data table length
CamType	RW	1688	cam type: 0 = cancelling cam 1 = incremental cam
CamTab	RW	1692	location of cam data table in the shared memory: 0 = TGM_Cam_Profile 1 = TGM_Data
CamExpControl	RW	1696	activity status of external data table: bit 0 = status of external data table 1 (0 = inactive, 1 = active) bit 1 = status of external data table 2 (0 = inactive, 1 = active)
CamExpLine1	W	1700	offset of the profile data beginning in the external data table 1 [bytes]
CamExpLine2	W	1704	offset of the profile data beginning in the external data table 2 [bytes]
CamExpAddress1	W	1708	offset of the variable in TGM_Data shared memory, where <b>TG Motion</b> writes the current output value of EXTDATA1 table [bytes]
CamExpAddress2	W	1712	offset of the variable in TGM_Data shared memory, where <b>TG Motion</b> writes the current output value of EXTDATA2 table [bytes]

**MAP\_CNC registers (interpolator)**

name	access	offset	description
Number	W	1776	specifies the number of the interpolator on which the servo drive is mapped: 0 = it is mapped on CNC 0 1 = it is mapped on CNC 1 2 = it is mapped on CNC 2
NumberAxes	W	1780	specifies the number of the axis of the selected interpolator (Cnc.Number register), on which the servo drive is mapped; the axis number corresponds to the axis in G-code, according to the table below: 0 = X 1 = Y 2 = Z 3 = C 4 = B 5 = U 6 = V 7 = W 8 = A 9 = O

**CAPTURE registers**

name	access	offset	description
Control	RW	1792	position capture control: 0 = capture is disabled 1 = capture by external digital input is allowed (for AKD and ServoStar servo amplifier only) 2 = one-shot capture of registers by <b>TG Motion</b> system is allowed
Typ_ExtCapture	W	1796	digital input number and type of edge on which the capture will be synchronized: bit 0 = capture by rising edge of digital input 1 allowed bit 1 = capture by falling edge of digital input 1 allowed bit 2 = capture by rising edge of digital input 2 allowed bit 3 = capture by falling edge of digital input 1 allowed bit 4 = capture by encoder zero pulse allowed
Offset	W	1800	address offset of the variable captured in CAPTURE.Data [bytes]
IData	RW	1804	captured data from TGM_Data shared memory from Offset; the data are of integer type (4 bytes)
PPosition	R	1824	servo drive actual position at the digital input rising edge [inc]
NPosition	R	1832	servo drive actual position at the digital input falling edge [inc]
PExt_Position	R	1840	external encoder position at the digital input rising edge [inc]
NExt_Position	R	1848	external encoder position at the digital input falling edge [inc]
Write_Position	R	1856	servo drive captured desired position [inc]
DData	R	1864	captured data from TGM_Data shared memory from Offset; the data are of double type (8 bytes)
Zero_Position	R	1872	drive actual position at the zero pulse of external encoder (CAPTURE.Type_ExtCapture – bit 4 set) [inc]

**Command registers**

name	access	offset	description
Control	RW	1920	the function in question will be executed by writing the function number into this register (refer to Chapter 5)
Reserve		1924	length reserve of one integer-type variable (4 bytes)
Par [0–14]	W	1928	15 double-type variables for the parameters of the transferred executed function; the meaning of the parameters depends on the function being subsequently called by inserting a non-zero value into the Command.Control register)