S7 - Functional block to control SEC-AC-... via Profibus

FESTO

Description

Functional block
S7

Type
SEC-AC-...-PB-S7
(Version 3.8)

PROFI
PROCESS FIELD BUS
BUS

Description
538 513
en 0803 NH
[676 917]

Original .... de

Layout .... Festo AG & Co KG., Abtl. KG-GD

Type setting .... DUCOM

Edition .... en 0803 NH

Designation P.BE-SEC-AC-PB-S7-EN

Order no. 538 513

Internet: http://www.festo.com
E-Mail: service_international@festo.com

Contents

Designated use

The functional block (FB) is used to control SEC-AC servo controllers in a SIMATIC-S7 controller via Profibus DP with integrated DP-Master module (e.g. CPU315-2DP)
The module allows the user to incorporate the various functions of the axis controller directly into their program.

Memory requirements for FB10 V3.8

	Program memory	Storage memory
functional block:	3316 byte	2826 bytes
Instance data module:	388 bytes	186 bytes

With multi-axis applications, the FB is accessed for each axis with a separate Instance-DB or Multi-instance. Further FBs and DBs are not required.
The simultaneous use of other FBs for controlling the same controller is not permitted.
The FB uses the System Functions SFC14 and SFC15 for consistent data transmission.

The module is incorporated into the user program and called there cyclically. A ”start_init” marker is used for initialisation and parameterisation; this is deleted the first time it is called (e.g. via OB100).

Fundamentals of programmed software

Please note that it is not possible with the present state of the art of technology to create programmed software which functions faultlessly for and is compatible with all applications and combinations intended by the user.

The software is therefore to be used exclusively for the functions described in the programming manual and in the user instructions.

At the time of handing-over or availability of the software, the latter is in a form in which it functions under normal operating and application conditions.

Festo does not guarantee that the software is satisfactory for all applications and purposes of use intended by the user, or especially that it functions faultlessly and is compatible with other programs which are used. The responsibility for the correct selection and the consequences arising from use of the software in the environment selected by the user, as well as the therewith intended and achieved results lies with the user. The same applies to the printed material accompanying the software.

The use of the programmed software does not therefore exonerate the user from his obligations or from his responsibility for the observance and adherence to the machine and technical safety regulations as well as a comprehensive functional check.

Target group

This manual is aimed at technicians trained in control and automation technology, fitters etc. who

– are working for the first time with digital drive controllers

– are already familiar with digital drive controllers, but who are installing a digital drive controller like the SEC-AC for the first time.

Information:
Your contact partner for technical support with this product can be found on the Festo Internet pages under the address:
www.festo.com [‹country-specific homepage›/Industrial Automation/‹country column›]

Information:
Supplementary software for your product (e.g. GSD Profibus file) can be found on the Festo Internet pages under the address:
www.festo.com [Industrial Automation/Service & Support/Download Area].

Important user instructions

Danger categories

This manual contains instructions on the possible dangers which may occur if the product is not used correctly. These instructions are marked (Warning, Caution, etc), printed on a shaded background and marked additionally with a pictogram. A distinction is made between the following danger warnings:

Warning
... This means that failure to observe this instruction may result in serious personal injury or damage to property.

Caution
... This means that failure to observe this instruction may result in personal injury or damage to property.

Please note
... This means that failure to observe this instruction may result in damage to property.

The following pictogram marks passages in the text which describe activities with electrostatically sensitive components.

Electrostatically sensitive components may be damaged if they are not handled correctly.

Marking special information

The following pictograms mark passages in the text containing special information.

Pictograms

Information:
Recommendations, tips and references to other information sources.

The parameters thus marked affect the internal controller structure and must be used with caution.

Accessories:
Information on necessary or sensible accessories for the Festo product.

Text markings

- The bullet indicates activites which may be carried out in any order.

1. Figures denote activities which must be carried out in the numerical order specified.

– Hyphens indicate general activities.

Chapter 1

1.1 Block functions

1.1.1 Positioning (default operation mode)

Travelling to a defined position with a defined speed and acceleration/braking ramp. The position can be approached absolute or relative to the actual position.
Profile movements are also possible through the specification of a final speed greater than zero with direct execution of the next positioning job.

1.1.2 Homing

Finding or specifying a reference position to reset the position counter. There are 15 different ways of doing this.

1.1.3 Jogging

Movement at variable speed in the selected direction, providing the relevant input is set.

1.1.4 Torque regulation

A defined force can be exerted using the drive without consideration of position and speed through the specification of a variable setpoint torque value.

1.1.5 Synchronous operation

Synchronising the drive to an external encoder signal provided via plug X10 as an additional setpoint position value.

This function can be selected in parallel to positioning mode.

	Inputs	Outputs	
Input varibles	LADDR_I		
	LADDR_O		
	feed_constant		
	EN_drive		
	homing_method		
	home_offset		
	homing_speed_switch		
	homing_speed_zero		
	homing_acceleration		
	start_homing		
	absolut_0_relativ_1		
	change_set_immediately		
	target_position		
	profile_velocity		
	end_velocity		
	profile_acceleration		
	profile_deceleration		
	new_set_point		
	jogging_velocity		
	jogging_minus		
	jogging_plus		
	hold		
	fault_reset		
	enable_torque		
	target_torque	en_OK	Output variables
	enable_sync	following_error	
	encoder_resolution	position_ok	
	encoder_numerator	home_complete	
	encoder_divisor	set_point_acknowledge	
		digital_inputs	
		position_actual_value	
		current_actual_value	
Transit variable	init_flag	error	

Fig. 1/1: SEC_control

Chapter 2

Contents

2.1 General

Variable		Bit	Range	Comment
LADDR_I / LADDR_O	I	16	Address range	Projected I/O start addresses of the slave
EN_drive	I	1	0/1	1 = drive ON / 0 = drive OFF
EN_ok	O	1	0/1	Drive active. No fault
feed_constant	I	32	2 . . $(2^{31}-1)$	Feed constant (in µm) see chapter A.1
init_flag	D	1	0/1	Must be deleted for initialisation
fault_reset	I	1	0/1	Reset fault (module and controller)
hold	I	1	0/1	Hold drive / abort function. Effect depends on operation mode
position_actual_value	O	32	$(-2^{31})..(2^{31}-1)$	Actual position (in µm)
current_actual_value	O	16	$(-2^{15})..(2^{15}-1)$	Actual motor current (in 1/1000 x nominal motor current)
digital_inputs	O	8	0 . . 255	Digital inputs of the controller (see chapter 3.1)
error	O	16	0 . . 127	Error bits (see chapter 3.1)

Fig. 2/2:

2.1.1 LADDR_I / LADDR_O

The slave I/O addresses of the respective controller are automatically assigned using Step7 during configuration of the hardware.

2.1.2 EN_drive / EN_ok

The **EN_drive** input enables the drive and the module functions.
In order for this to happen, the drive enable logic must be parameterised to "Input Din5 plus fieldbus" in the controller and the digital input 5 must be supplied with 24 V.
The **EN_ok** output indicates that the drive is operating and that there are no faults.

2.1.3 feed_constant

The **feed_constant** variable informs the controller of the feed distance in µm covered by the axis during one motor revolution. Any gear conversion ratio must be taken into consideration.

For example:

axis = DGE-25-750-ZR (63 mm feed), motor = MTR-AC-70-GA (i = 4).
Feed constant = 15750 µm.
For the standard values see Appendix A.

The controller uses this information to convert the remaining position parameters so that the following units can be used:

- Position specifications, actual position = µm
- Speeds = µm/s
- Acceleration = µm/s^2

The feed constant is read in once upon initialisation of the module.

2.1.4 init_flag

A marker is required in order to initialise the module and the associated controller. This marker is deleted the first time it is called (for example via OB100). Once initialisation has been performed the marker is set by the module. Resetting the marker allows a re-initialisation to be forced at any time. This variable must not be held at 'logical 0' by a fixed instruction.

2.1.5 Position_actual_value / current_actual_value / digital_inputs

These output variables are a constant source of the following status information:

- the actual position (in µm)
- the actual current (in 1/1000 x nominal motor current)
- the signal status of the controller's digital inputs.

2.2 Homing

Variable		Bit	Range	Comment
start_homing	I	1	0/1	Enables and starts homing
homing_method	I	8	1 . . 15	Homing method
home_offset	I	32	$(-2^{31})..(2^{31}-1)$	Offset of the reference position (in µm)
homing_speed_switch	I	32	$0 .. (2^{31}-1)$	Finds the speed switch [µm/s]
homing_speed_zero	I	32	$0 .. (2^{31}-1)$	Exits the speed switch [µm/s]
homing_acceleration	I	32	$0 .. (2^{31}-1)$	Acceleration / delay [µm/s^{2}]
home_complete	O	1	0/1	Homing successfully completed

Fig. 2/3:

The purpose of homing is to reset the positioning system. It is required only once following a power-on, but can be performed any number of times.
Homing is activated by setting the **start_homing** input. This input must remain set until homing is complete, otherwise it will be aborted.
Successful conclusion is indicated with the output **home_complete**. The output is reset with:
Start reference travel, fault bit 0, 1 or 3, new initialization.

Using the hold input to abort homing results in an error message

The **home_offset** variable can be used to move the machine zero point in a positivee or negativee direction relative to the reference point.

There are 15 different ways of determining the reference point.

2.2.1 Methods of homing

<table>
<tr><th>No.</th><th>Direction</th><th>Target</th><th>Connec-tion</th><th>Reference point for zero position</th><th>Remarks on zero pulse</th></tr>
<tr><td>1</td><td>negative</td><td>Limit switch</td><td>(X1.22)</td><td>Zero pulse</td><td rowspan="15">The resolver as actual-value sensor supplies absolute values within a revolution.

It is not therefore necessary to make a movement in order to evaluate the zero impulse.
The absolute value of the resolver position is simply added direction-related to the current position or accepted as the current actual position.

Method 11 sets the position counter directly to 0.</td></tr>
<tr><td>2</td><td>positive</td><td>Limit switch</td><td>(X1.10)</td><td>Zero pulse</td></tr>
<tr><td>3</td><td>positive</td><td>Reference switch</td><td>(X1.11)</td><td>Zero pulse</td></tr>
<tr><td>4</td><td>negative</td><td>Reference switch</td><td>(X1.11)</td><td>Zero pulse</td></tr>
<tr><td>5</td><td>negative</td><td>Limit switch</td><td>(X1.22)</td><td>Limit switch</td></tr>
<tr><td>6</td><td>positive</td><td>Limit switch</td><td>(X1.10)</td><td>Limit switch</td></tr>
<tr><td>7</td><td>positive</td><td>Reference switch</td><td>(X1.11)</td><td>Reference switch</td></tr>
<tr><td>8</td><td>negative</td><td>Reference switch</td><td>(X1.11)</td><td>Reference switch</td></tr>
<tr><td>9</td><td>negative</td><td>No travel</td><td></td><td>Zero pulse</td></tr>
<tr><td>10</td><td>positive</td><td>No travel</td><td></td><td>Zero pulse</td></tr>
<tr><td>11</td><td></td><td>No travel</td><td></td><td>Current actual position</td></tr>
<tr><td>12</td><td>positive</td><td>Stop</td><td></td><td>Stop</td></tr>
<tr><td>13</td><td>negative</td><td>Stop</td><td></td><td>Stop</td></tr>
<tr><td>14</td><td>positive</td><td>Stop</td><td></td><td>Zero pulse</td></tr>
<tr><td>15</td><td>negative</td><td>Stop</td><td></td><td>Zero pulse</td></tr>
</table>

Fig. 2/4:

2.3 Positioning

Variable		Bit	Range	Comment
new_set_point	I	1	0/1	executes new positioning profile
set_point_acknowledge	O	1	0/1	Handshake: 0 = new profile can be transferred, 1 = new profile has been transferred, but not yet executed
change_set_immediatly	I	1	0/1	New positioning profile will be executed: 0 = once the current profile has been processed, 1 = immediately; the current profile will be aborted
absolut_0_relativ_1	I	1	0/1	0 = absolute positioning, 1 = relative positioning
target_position	I	32	$(-2^{31})..(2^{31}-1)$	Positioning profile: target position [µm]
profile_velosity	I	32	$(-2^{31})..(2^{31}-1)$	Positioning profile: positioning speed [µm/s]
end_velosity	I	32	$(-2^{31})..(2^{31}-1)$	Positioning profile: final speed [µm/s]
profile_acceleration	I	32	$0 .. (2^{31}-1)$	Positioning profile: acceleration [µm/s²]
profile_deceleration	I	32	$0 .. (2^{31}-1)$	Positioning profile: delay [µm/s²]
position_ok	O	1	0/1	Drive is in target window
following_error	O	1	0/1	Following error

Fig. 2/5:

The **new_set_point input** must be set before the axis can move.

It transmits the positioning profile consisting of the target position, positioning speed, final speed, acceleration and delay to the controller. The controller must also be informed of whether the position is absolute or relative (incremental value).

The **position_ok output** is set once the profile has been processed and the drive is in the defined target window.

The final speed is usually zero. The speed at which the directly following positioning job is to be executed is only specified here if a so-called profile movement is to be executed. With individual positioning tasks this variable is not relevant.

As soon as one positioning profile has been transferred from the controller and executed, the next one can be transferred (**set_point_acknowledge output** is zero).
If the **change_set_immediately** input is not set, the second profile will be executed immediately upon completion of the first. If the input is set, the current profile will be aborted and the new one executed immediately.

The **following_error** output indicates that the difference between the actual position value and the target position value is too great. The reason for this could include the acceleration being too high, the current being too low or the load on the drive being too heavy. The following error is indicated exclusively via the output; no error message is generated.

Positioning can be aborted using the **hold** input. The specified **profile_deceleration** is used here.

➔

Please note
Positioning mode is always active if no other function is selected. This means that the drive is always in position control mode, provided the drive is enabled and there are no errors.

If another function is activated while a positioning job is active, it will be aborted immediately.

2.4 Jogging

Variable	Bit	Range	Comment
jogging_minus I	1	0/1	Jogging in minus direction
jogging_plus I	1	0/1	Jogging in positive direction
jogging_velocity I	32	$(-2^{31})..(2^{31}-1)$	Positioning speed in jogging mode [µm/s]

Fig. 2/6:

Setting a **jogging_minus** or **jogging_plus** input moves the drive in the selected direction at a controlled speed.
As soon as the respective input is reset, the drive brakes and returns to position control mode.
The positioning profile data is transferred for the acceleration and braking ramps.

The speed in jogging mode is specified dynamically with the variable **jogging_velocity**.

Caution
A negative value inverts the direction function of the jogging buttons.

This function facilitates manual operation or setting operation. Since the current actual position is always available, it is also a simple way of realising teach-in operation.

The **hold** input sets the speed to zero.

2.5 Torque regulation

Variable	Bit	Range	Comment
enable_torque I	1	0/1	Activates torque regulation
target_torque I	16	$(-2^{15})..(2^{15}-1)$	Desired torque value in 1/1000 x nominal motor torque[1]

[1] The rated torque of a motor is derived from the parametrized rated current multiplied by the torque constant.

MTR-AC-55-3S	= 0,457	MTR-BSM-50	= 0,75
MTR-AC-70-3S	= 0,32	MTR-BSM-63	= 0,74
MTR-AC-100-3S	= 0,711	MTR-BSM-80	= 0,79
MTR-AC-100-5S	= 1,49	MTR-BSM-90	= 0,79

Example: motor MTR-AC-100-3S parametrized with 2.6 A rated current.
Rated torque = 2.6 A × 0.711Nm/A = 1.85 Nm.
0.9 Nm should be specified: (1000 / 1.85 Nm) × 0.9 Nm = 486

Fig. 2/7:

If the **enable_torque** input is set, the controller supplies the motor with a constant current which it converts to a constant torque, without considering speed and position.

The torque value is specified by the **target_torque** variable in 1/1000 x nominal torque.
This results in sensible values in the range -1000 to +1000 (1000 = rated torque). Higher values can only have a short effect, as the I^2t monitoring automatically regulates back to the rated current. The parametrized peak current cannot be exceeded.

The **hold** input holds the torque at zero.

The nominal torque value is passed over a ramp generator. The ramp time is 100 ms for a modification of 0 Nm on the rated torque.

At the end of the operating mode, the controller can be reset if Enable is switched off for approx. 100 ms.

Warning
As no speed control or limitation is effective here, the drive can reach non-permitted high speeds if no appropriate torque has a counter effect.

2.6 Synchronous operation

Variable		Bit	Range	Comment
enable_sync	I	1	0/1	Activates synchronous operation
encoder_resolution	I	32	4 . . 500000000	Encoder resolution at X10 in increments per revolution
encoder_numerator	I	16	$(-2^{15})..(2^{15}-1)$	Numerator (conversion factor for encoder to motor revolution)
encoder_divisor	I	16	$(-2^{15})..(2^{15}-1)$	Divisor (conversion factor for encoder to motor revolution)

Fig. 2/8:

In synchronous operation, the position controller is supplied with an additional desired position value via the connection X10.
This allows the drive to be connected as a slave to a master (by connecting input X10 Slave with output X11 Master), for example.

A further application is the "flying saw" function. This function is implemented by supplying the signal of an encoder at X10 which is connected to a conveyor belt, for example.

Activating synchronous operation via **enable_sync synchronises** the axis to the movement of the belt. As soon as the drive is in synchronicity with the belt (indicated by the **position_ok** output), the saw can be activated. Following the saw stroke, synchronous operation is deactivated once more and the saw is returned to its original position.

The ratio of encoder revolutions to motor revolutions is made known to the controller via the **encoder_numerator** and **encoder_divisor** variables so that the encoder can be adapted. The encoder resolution must also be made known via **encoder_resolution**.

Since the controller uses 16-bit arithmetic to perform the conversion of the encoder signals, there are certain restrictions when it comes to the resolution and the conversion factors. The following points must therefore be observed:
The encoder at X10 should have a binary increment, i.e. a resolution of 4096, for example.
Decimal increments (e.g. 4000) restrict the conversion factor range and result in rounding errors.

Synchronous operation is only permissible in parallel with positioning. No other function can be activated as long as synchronous operation is active. However positioning jobs can be performed while synchronous operation is active.

➔

Please note
The synchronous operation variables for the encoder resolution and the conversion factor are transferred once to the controller during initialisation of the module. If these are changed while the module is active, it must be re-initialised.

Chapter 3

Contents

3.1 Bit mappings of the output variables

3.1.1 8-bit output variable "digital_inputs"

Bit	Function		X1-Pin
0 1 2	Negative limit switch actuated Positive limit switch actuated Status sample input	Irrespective of normally-open or normally-closed contact Irrespective of normally-open or normally-closed contact Reference switch	22 10 11
3 4 5 6 7	Target selector POS0 Target selector POS1 Target selector POS2 Target selector POS3 Start input	The standard functions of these inputs are deactivated in Profibus mode	19 7 20 8 23

Fig. 3/1:

3.1.2 16-bit output variable "error"

Bit	Message	
0 1	Return error SFC14 Return error SFC15	Possible causes: incorrect slave address, slave not ready to operate, bus interruption. Specific fault code in Instance-DB, address 138 / 140
2 3 4 5 6	Hardware input, output stage enable or drive enable missing Controller in error state (error shown on controller display) Drive enable logic in controller not set to BUS Reference travel started with invalid method Invalid feed constant during initialization	

Fig. 3/2:

3.2 Notes on Profibus mode

Once a drive has been parameterised, optimised and commissioned using the parameterisation software, the Profibus DP interface must be activated and the bus address specified. This is also done using the parameterisation software via the **"Transfer"** command in the **"File"** menu.

- The following configuration command must be entered in the Transfer dialog box:
  BUS:NN:0000:1000

NN specifies the slave address in hexadecimal form, the value 1000 activates the bus.

- The following command must be entered to deactivate the bus:
  BUS:NN:0000:0000
  Transfer using **Enter** or **"Send"**

The bus configuration can be queried using **BUS?**

Once parameterisation has been completed and the bus activated, do not forget to back up the parameters (in the controller) and transfer the parameter record to a data volume. This means that a substitute controller can be immediately commissioned on the bus at any time simply by transferring the parameter record.

3.3 Rotary axis

If a rotary axis is to be dimensioned circularly, this can be carried out with the following two steps:

1. Adapt the feed constant.
   The feed constant no longer specifies the feed in 1/1000 mm but in 1/1000 degrees for a motor revolution. The unit for positioning is therefore 1/1000 degrees.

2. Limit the positioning range (via RS232).
   The positioning range is defined by specification of a lower and upper limit, e.g. 0 to 360.000 degrees. If a limit value is exceeded, the position counter will jump automatically to the other limit value.

3.3.1 Procedure

The range limits are defined with Object 607Bxx (application see chapter 3.4).
The lower limit with 607B01; the upper limit with 607B02 (parameters 32 bits each).
The range limits are activated with Object 651020 (parameter 16 bits).
The feed constant is transferred to the FB with the input variable **feed_constant**.

Example

A motor with gears i = 4 drives a rotary gripper by means of a toothed belt with the ratio 3 to 1. The
total ratio is 3 x 4 = 12. The value 360000 / 12 = 30000 [1/1000 degrees] is specified as feed constant
(1 motor revolution ≙ 30,000 degrees gripper revolution).

Limit positioning range by means of RS232 commands

Lower limit	(0)	=607B01:00000000
Upper limit	(360000)	=607B02:00057E40
Activate limits		=651020:0001
(deactivate)		=651020:0000)

Feed constant

feed_constant: =L#30000

3.3.2 Notes on application

If referencing is necessary, a method with a separate reference switch must be selected. Limit switches are not possible on a rotary axis which moves through more than 360 degrees. As an alternative, the zero impulse can be used as the reference point, providing there is a total ratio of 1.

Absolute positioning can be made within the 360 degrees, whereby the shortest path will automatically be traversed.

With relative positioning, paths of any desired length are possible (multiples of 360 degrees). The direction of rotation is determined by the sign in front of the length specification.

Positioning over several revolutions with absolute target within the circle must be carried out in two steps:
1. number of relative revolutions, 2. absolute circle position.

However, if it must be assured that absolute positioning always take place in the same direction, we recommend that the total positioning path be calculated from the current position and that the complete path be traversed relatively.

3.4 Direct access to individual parameters in the controller

The parameters of the controllers are addressed via objects. These objects are identified by an index and a subindex. A listing and description of the objects can be found in the PROFIBUS-DP product handbook.

These objects can be read aMnd written to via the serial interface. For example using the parameterisation software in the Transfer dialog box (menu: **File / Transfer** . .).
This means that objects which cannot be reached via the standard functions in the parameterisation software or the functional block can be used.

Written form

Read Object:	?XXXXSI
Write Object	=XXXXSI:WWWWWWWW

XXXX	Index of the object (hexadecimal).
SI	Subindex of the object (hexadecimal). Subindex 00 is optional.
WW	Value of the object (8 bit hexadecimal).
WWWW	Value of the object (16 bit hexadecimal).
WWWWWWWW	Value of the object (32 bit hexadecimal).

If the bit width of the object value is not known, it is recommended to first read out the object. The length is correctly output in the response.

Example Read

=607B01	Answer:	=607B01:00004E20
=651020		=651020:0001

Example Write

=607D01:80000000	
=604D:04	(the subindex 00 does not have to be specified)

Please note

– Modifications entered are effective immediately.
– Incorrect entries or modifications to non-permitted Objects can result in malfunctioning or serious faults.
– Before issuing a write command check that the address is correct (Index, Subindex), in order that an incorrect Object is not overwritten by mistake.

Please note

The following applies here also:
if parameters were changed, they must be backed up so that they remain permanently effective. (menu: Parameters / Back up parameter record).

Appendix A

A.1 Standard feed constants of the Festo linear drives with servo motors type MTR-AC-...

Size	Type	Motor with/without gears MTR-AC-...-A MTR-AC-...-G	Feed constant [µm]
Spindle axis			
18	DGE-18-...-SP-...	Without gears	4000
25	DGE-25-...-SP-...	Without gears	10000
40	DGE-40-...-SP-...	Without gears	20000
	DGE-40-...-SP-...	With gears i = 4	5000
63	DGE-63-...-SP-...	Without gears	30000
	DGE-63-...-SP-...	With gears i = 4	7500

Fig. A/3:

Size	Type	Motor with/without gears **MTR-AC-...-A** **MTR-AC-...-G**	Feed constant [µm]
Toothed belt axis			
18	DGE-18-...-ZR-...	Without gears	52000
25	DGE-25-...-ZR-...	Without gears	63000
	DGE-25-...-ZR-...	With gears i = 4	15750
40	DGE-40-...-ZR-...	Without gears	100000
	DGE-40-...-ZR-...	With gears i = 4	25000
63	DGE-63-...-ZR-...	Without gears	176000
	DGE-63-...-ZR-...	With gears i = 4	44000
18	DGEA-18-...-ZR-...	Without gears	81000
	DGEA-18-...-ZR-...	With gears i = 4	20250
25	DGEA-25-...-ZR-...	Without gears	81000
	DGEA-25-...-ZR-...	With gears i = 4	20250
40	DGEA-40-...-ZR-...	Without gears	120000
	DGEA-40-...-ZR-...	With gears i = 4	30000
25	DGE-25-...-RF-...	Without gears	90000
	DGE-25-...-RF-...	With gears i = 4	22500
40	DGE-40-...-RF-...	Without gears	125000
	DGE-40-...-RF-...	With gears i = 4	31250
63	DGE-63-...-RF-...	Without gears	232000
	DGE-63-...-RF-...	With gears i = 4	58000

Fig. A/4: