7.3 PLC command syntax

Operand	Comment
Cxx, Cyy	Counter index 00-10
Hxxx, Hyyy	Variable index 000-127
Fxxx, Fyyy	Variable index 000-127
Zxx, Zyy	Timer index 00-10
Ny	Line number 001-254
PARA[n, i]	Parameter number n 000-999 Parameter index i 000-255
Mxxx, Myyy	Flag index 000-255
Ippi	Inputs ppi = A00, A00, E00-E07, S00-S03 (CDB3000), S00-S06 (CDE3000), S00-S02 (CDF3000)
Oppi	Outputs ppi = E00-E03, S00-S02 (CDB3000), S00-S04 (CDE3000), S00, S03-S05 (CDF3000)

Operand	Comment
b	Value 1-32
d	Counter reading 0 ...65535 (16 bit)
t	Timer reading 0 ... 4.294.967.295 (32 bit)
f	Numerical floating point value (32 bit)
z	Integer numerical value ±2147483648 (32 bit)

Logic operands:

Operand	Comment
&	AND
|	OR
^	Exclusive OR
!=	≠
<=	≤
>=	≥
ABS	Absolute-value generation

Mathematical operands:

Operand	Comment
+	Addition
-	Subtraction
*	Multiplication
:	Division
%	Modulo
ABS	Absolute-value generation
ROUND	Rounding

7.3.1 Overview

Comm and	Operand		Comment
Jump instructions			
JMP		Ny/END	unconditional jump
	(ACTVAL = < > Hxxx,Fyyy)	Ny/END	Actual value
	(ACTVAL <= >= Hxxx,Fyyy)	Ny/END	
	(ACTVAL != Hxxx,Fyyy)	Ny/END	
	(ACTVAL = != 0)	Ny/END	
	(REFVAL = < > Hxxx,Fyyy)	Ny/END	Setpoint
	(REFVAL <= >= Hxxx,Fyyy)	Ny/END	
	(REFVAL != Hxxx,Fyyy)	Ny/END	
	(REFVAL = != 0)	Ny/END	
	(REF = 0/1, =Mxxx)	Ny/END	Axis status setpoint reached
	(ROT_0 = 0/1, =Mxxx)	Ny/END	Axis status standstill
	(Ippi = 0/1)	Ny/END	Status of an input
	(Oppi = 0/1)	Ny/END	Status of an output
	(Mxxx = 0/1, = != Myyy)	Ny/END	Status of a flag
	(spec. flag = 0/1, = != Myyy)	Ny/END	Status of a special flag, e. g. STA_REF
	(Mxxx & | ^ Ippi)	Ny/END	Logic operation flag input
	(Mxxx & | ^ Oppi)	Ny/END	Logic operation flag output
	(Hxxx = != 0)	Ny/END	
	(Hxxx = != < <= > >= Hyyy)	Ny/END	Value of integer variables
	(Fxxx = != 0.0)	Ny/END	
	(Fxxx= != < <= > >= Fyyy)	Ny/END	Value of floating point variables
	(Cxx = != d)	Ny/END	Counter status
	(Zxx = != 0)	Ny/END	Timer status
	END		Jump to program end
Sub-program invocation			
CALL	Ny		Sub-program invocation after line Ny Maximum nesting depth. 250
RET			Return to the line of sub-program invocation
BRKPT	SET BRKPT=1		Activates breakpoint; the set breakpoint is evaluated
	SET BRKPT=0		Deactivates breakpoint; the set breakpoint is not evaluated

Command	Operand	Comment
Setting commands		
SET	Oppi = 0/1, Mxxx	Output direct or with flag
	OUTPUT = Hxxx	Set output image
	Mxxx = 0/1, Ippi, Oppi, Myyy, M[Cxx]	Set flag
	Mxxx = Hxxx	Set flag (LSB of Hxxx)
	M[Cxx] = 0/1	
	M[Cxx] = Myyy	Set flag (indexed*)
	Mxxx & | ^ Myyy	Link flag logically
	Mxxx = STA_ERR	Read error status (1 -> error)
	Mxxx = STA_WRN	Read warning status (1 -> Warning)
	Mxxx = STA_ERR_WRN	Read warning/error status (1 -> Warning/Error)
	Mxxx = STA_ACTIV	Control active
	Mxxx = STA_ROT_R	Motor turning clockwise
	Mxxx = STA_ROT_L	Motor turning anti-clockwise
	Mxxx = STA_ROT_0	Motor standstill
	Mxxx = STA_LIMIT	Setpoint limitation
	Mxxx = STA_REF	Setpoint reached
	Mxxx = STA_HOMATD	Reference point defined
	Mxxx = STA_BRAKE	Quick stop active
	Mxxx = STA_OFF	Deenergized state
	Mxxx = STA_C_RDY	Control standby state
	Mxxx = STA_WUV	Undervoltage warning
	Mxxx = STA_WOV	Overvoltage warning
	Mxxx = STA_ WIIT	Warning I^2*t
	Mxxx = STA_WOTM	Warning motor overtemperature
	Mxxx = STA_WOTI	Warning heat sink temperature
	Mxxx = STA_WOTD	Warning inside temperature
	Mxxx = STA_WIS	at present no function (always 1)
	Mxxx = STA_WFOUT	at present no function (always 1)
	Mxxx = STA_WFDIG	at present no function (always 1)
	Mxxx = STA_ WIT	Warning I*t motor protection
	Mxxx = STA_ WTQ	Warning torque
	Mxxx = STA_INPOS	Setpoint position reached
	ENCTRL = 0/1, Mxxx	Controller off / on
	INV = 0/1, Mxxx	Invert setpoint (only with speed and torque control)
	ERR = 1, Mxxx	Trigger error

Command	Operand	Comment
	ERRRQ = 1, Mxxx	Reset error
SET	BRKPT = 0/1, Mxxx	Breakpoints off / on
	BRAKE = 0/1, Mxxx	Quick stop off / on
	HALT = 0/1, Mxxx	Halt/Feed off / on
	PCTRL = 0/1, Mxxx	no function
	Hxxx = EGEARPOS, EGEARSPEED	Read reference encoder increments, reference encoder speed
	F[CXX], H[Cxx], M[Cxx] = Value	Indexed assignment
	Hxxx = z, Hyyy, H[Cyy], Fxxx, Mxxx, Cyy, Zxx	Set variable
	H[Cxx] = z, Hyyy	Set integer variable (indexed*)
	Hxxx + - * : % z, Hyyy	Calculate variable
	Hxxx << >> z, Hyyy	Displace variable
	Hxxx = ABS Hyyy	Variable absolute-value generation
	Hxxx = PARA[n], PARA[n, i]	Set variable
	Hxxx, Fxxx = REFPOS	Position setpoint
	Hxxx, Fxxx = ACTPOS	Actual position value
	Hxxx, Fxxx = ACTFRQ	Assign actual frequency [Hz]
	Hxxx, Fxxx = ACTSPEED	Assign actual speed [min^{-1}]
	Hxxx, Fxxx = ACTTORQUE	Assign actual torque [Nm]
	Hxxx, Fxxx = ACTCURRENT	Assign actual current (effective) [A]
	Hxxx = OSA0	Analog output value
	Hxxx = ISA0, ISA1	Assign analog input 0 / 1
	Hxxx = OUTPUT, INPUT	Read variable with output or input image
	EGEARPOS = Hxxx	Set reference encoder increments
	OSA0 = Hxxx	Assign analog value
	REFVAL = Hxxx, Fxxx	Assign setpoint (only with speed and torque control)
	INPOSWINDOW = Hxxx	Setpoint reaches window
	Fxxx = f, Hxxx, F[Cxx], Fyyy	Set floating point variable
	F[Cxx] = f, Fyyy	Set floating point variable (indexed)
	Fxxx + - * : f, Fyyy	Calculate floating point variable
	Fxxx = ROUND Fyyy	Round floating point variable
	Fxxx = ABS Fyyy	Floating point variable absolute-value generation
	Fxxx = PARA[n, i], PARA[n], PARA[Hyyy,Hzzz], PARA[Hyyy]	Set parameter
	Cxx = d, Cyy, Hyyy	Set counter
	Cxx + - d, Hyyy	Calculate counter
	Zxx = t, Hyyy	Set timer
	PARA[n] = Hxxx, Fxxx	Parameter number direct

Comm and	Operand	Comment
	PARA[Hxxx] = Hyyy, Fxxx	Parameter number via integer variable
SET	PARA[n,i] = Hxxx, Fxxx	Input parameter number, direct
	PARA[Hxxx, Hyyy] = Hzzz, Fxxx	Specification parameter number and index via integer variable
	ACCR = Hxxx	Change acceleration
	DECR = Hxxx	
	ACCR = 0 ...150%	Scaling
	DECR = 0 ...150%	Scaling
Wait commands		
WAIT	d, Hxxx	Wait time in ms (0 ... 4.294.967.295 ms)
	ROT_0	Setpoint position = target position
	REF	Actual position in position window
	PAR	Wait until parameter is written.
Travel commands (only with positioning)		
GO	W A Hxxx	Travel **absolute** by value of Hxxx with speed acc. to parameter 724_POSMX and wait with program processing, until target position is reached.
	W R Hxxx	Travel **relative** by value of Hxxx with speed acc. to parameter 724_POSMX and wait with program processing, until target position is reached.
	A Hxxx	Travel **absolute** by value of Hxxx with speed acc. to parameter 724_POSMX (program processing continues)
	R Hxxx	Travel **relative** by value of Hxxx with speed acc. to parameter 724_POSMX (program processing continues)
	0	perform selected referencing
	0+Hxxx	perform selected referencing and set reference position=Hxxx
	A Hxxx V Hyyy	Travel **absolute** by value of Hxxx with speed Hyyy (program processing continues)

Command	Operand	Comment
	R Hxxx V Hyyy	Travel **relative** by value of Hxxx with speed Hyyy (program processing continues)
GO	T[Hxxx]	Position via table
	T[Cxx]	Travel via table entry Cxx
	W T[Hxxx]	Travel via table entry Hxxx, wait
	W T[Cxx]	Travel via table entry Cxxx, wait
	T[xxx]	Travel via table entry xxx
	W T[xxx]	Travel via table entry xxx, wait until position is reached
	V Hxxx	Travel endless via variable
	W A Hxxx V Hyyy	Travel **absolute** by value of Hxxx with speed Hyyy and wait with program processing, until target position is reached
	W R Hxxx V Hyyy	Travel **relative** by value of Hxxx with speed Hyyy and wait with program processing, until target position is reached
	SYN 1 / SYN 0	Switching synchronous travel on and off
Command to stop the drive		
STOP	B	Braking with parameterized deceleration (only with positioning)
STOP	M	Braking with quick stop ramp (only with positioning)
STOP	0	Braking with quick stop ramp and shut-down of control, if control location=PLC (only with positioning)
SET	BRAKE = 0/1, Mxxx	Perform quick stop acc. to quick stop reaction (see 6.2.3): 1: Perform quick stop 0: End quick stop
SET	HALT = 0/1, Mxxx	Stop feed acc. to reaction (see 6.2.3): 1: Stop axis 0: Enable axis
Further commands		
NOP		Instruction without function
INV	Oppi, Mxxx, Hxxx	Inverting
END		Quits the program, all other lines will be ignored. Do not enter line number.

Comm and	Operand	Comment
BRKPT		Insert breakpoint into program line, evaluation with active breakpoints, see page 7-11

7.3.2 Detailed explanations

Jump instructions and sub-program invocation (JMP)

- Unconditional jump instructions will be executed in any case (without condition).
- Conditional jump instructions will only be executed when the specified condition is fulfilled. The condition for execution is specified in parenthesis (...).
- A line number or the end of the program is always specified as jump target.

Attention: If a JMP/SET command is set to non-existing inputs/outputs, no error message will be generated.

Unconditional jump instructions

These commands are not linked to any prerequisites (axis position, status of programmed variables) and are thus executed directly and unconditionally.

```
JMP  Ny     Jump to set with number y
JMP  END    Jump to program end
```

Conditional jump instructions

Conditional jump instructions / sub-program invocations are linked with certain conditions, which are specified in parenthesis. If this condition is fulfilled, the jump to the specified set number or the end of the program will be executed. If the condition is not fulfilled, the program will continue with the next successive set.

Note: The execution of a conditional jump can be linked to one of the following conditions.

Actual value

reached:

```
JMP (ACTVAL =   Hyyy,Fyyy)   Ny/END
```

exceeded:

```
JMP (ACTVAL >   Hxxx,Fyyy)   Ny/END
JMP (ACTVAL >=  Hxxx,Fyyy)   Ny/END
```

fallen short of:

```
JMP (ACTVAL <   Hxxx,Fyyy)   Ny/END
JMP (ACTVAL <=  Hxxx,Fyyy)   Ny/END
```

compare:

```
JMP (ACTVAL !=  Hxxx,Fyyy)   Ny/END
JMP (ACTVAL =   0)           Ny/END
JMP (ACTVAL !=  0)           Ny/END
```

Note: The command REFVAL is of relevance for the speed control. In case of positioning the command REF is processed, because this command refers to "Setpoint reached".

Setpoint

reached:

```
JMP (REFVAL   =   Hxxx,Fyyy)   Ny/END
```

exceeded:

```
JMP (REFVAL   >   Hxxx,Fyyy)   Ny/END
JMP (REFVAL   >=  Hxxx,Fyyy)   Ny/END
```

fallen short of:

```
JMP (REFVAL   <   Hxxx,Fyyy)   Ny/END
JMP (REFVAL   <=  Hxxx,Fyyy)   Ny/END
```

compare:

```
JMP (REFVAL   !=  Hxxx,Fyyy)   Ny/END
JMP (REFVAL   =   0)           Ny/END
JMP (REFVAL   !=  0)           Ny/END
```

Axis status

REF reached:

```
JMP (REF = 1)        Ny/END       Actual value in setpoint window
```

REF not reached:

```
JMP (REF = 0)        Ny/END       Actual value not in setpoint
window
```

in dependence on a flag:

```
JMP (REF = Mxxx)     Ny/END       Flag: Mxxx=1; Mxxx=0
```

Axis stopped:

```
JMP (ROT_0 = 1)      Ny/END
```

Axis moves:

```
JMP (ROT_0  = 0)     Ny/END
```

in dependence on a flag:

```
JMP (ROT_0 = Mxxx)   Ny/END
```

Status of a digital input

Status = 0:

```
JMP (Ippi = 0)       Ny/END
```

Status = 1:

```
JMP (Ippi = 1)       Ny/END
```

Status of a digital output

Status = 0:

```
JMP (Oppi = 0)       Ny/END
```

Status = 1:

```
JMP (Oppi = 1)       Ny/END
```

Status of a logic flag

```
JMP (Mxxx = Myyy)     Ny / END
JMP (Mxxx != Myyy)    Ny / END
JMP (Mxxx = 0)        Ny / END
JMP (Mxxx = 1)        Ny / END
JMP (Mxxx & Ippi)     Ny / END
JMP (Mxxx | Ippi)     Ny / END
JMP (Mxxx ^ Ippi)     Ny / END
JMP (Mxxx & Oppi)     Ny / END
JMP (Mxxx | Oppi)     Ny / END
JMP (Mxxx ^ Oppi)     Ny / END
```

Status of a special flag

```
JMP (spec. flag = Mxxx)            Ny / END
JMP (spec. flag != Mxxx)           Ny / END
JMP (spec. flag = 0)               Ny / END
JMP (spec. flag = 1)               Ny / END
```

Value of an integer variable (direct comparison)

compare:

```
JMP (Hxxx = 0)        Ny / END
JMP (Hxxx != 0)       Ny / END
```

Value of an integer variable (comparison with second variable)

compare:

```
JMP (Hxxx = Hyyy)     Ny / END
JMP (Hxxx != Hyyy)    Ny / END
```

exceeded:

```
JMP (Hxxx >= Hyyy)    Ny / END
JMP (Hxxx > Hyyy)     Ny / END
```

fallen short of:

```
JMP (Hxxx <= Hyyy)    Ny / END
JMP (Hxxx < Hyyy)     Ny / END
```

Value of a floating point variable (direct comparison)

compare:

```
JMP (Fxxx = 0.0)      Ny / END
JMP (Fxxx != 0.0)     Ny / END
```

Value of a floating point variable (comparison with second variable)

compare:

```
JMP (Fxxx = Fyyy)     Ny / END
JMP (Fxxx != Fyyy)    Ny / END
```

exceeded:

```
JMP (Fxxx >= Fyyy)    Ny / END
JMP (Fxxx > Fyyy)     Ny / END
```

fallen short of:

```
JMP (Fxxx <= Fyyy)    Ny / END
JMP (Fxxx < Fyyy)     Ny / END
```

Status of a counter

```
JMP (Cxx = d)         Ny/END      Jump if value is reached
JMP (Cxx != d)        Ny/END      Jump if value is not reached
```

Status of a timer

```
JMP (Zxx = 0)        Ny/END       Timer run out?
JMP (Zxx != 0)       Ny/END       Timer not yet run out?
```

Note: A query for equality is only possible with a run-out timer (i.e. "= 0"), because it cannot be assured that a certain intermediate status ("=t") is reached at the time of the query.

Sub-programs (CALL, RET)

A sub-program is a part of the main program. No independent program header, e. g. P01, is generated. The invocation is not realized by means of JMP, but via CALL.

```
CALL Ny         Invocation of a sub-program, or a jump to
                the first program line of the sub-program

RET             Return from the sub-program
```

Possible structure of the program
(the line numbers only serve as examples)

```
N010 ...            ; Start of main program
...
N050 CALL N110      ; Sub-program invocation
...
N100 JMP ...        ; End of main program


N110 ...            ; Start of sub-program
...
N200 RET            ; End of sub-program
```

After processing of the sub-program the program is continued with the set following the invocation (CALL). The maximum nesting depth for sub-programs is 250. If this number is exceeded an error message will be issued and the running program will be aborted.

Setting a breakpoint (BRKPT)

With this command the sequential program can be interrupted at any line.

How to use breakpoints in a sequential program:

Activating/deactivating breakpoints in the sequential program

```
Ny SET BRKPT = 1 / 0
```

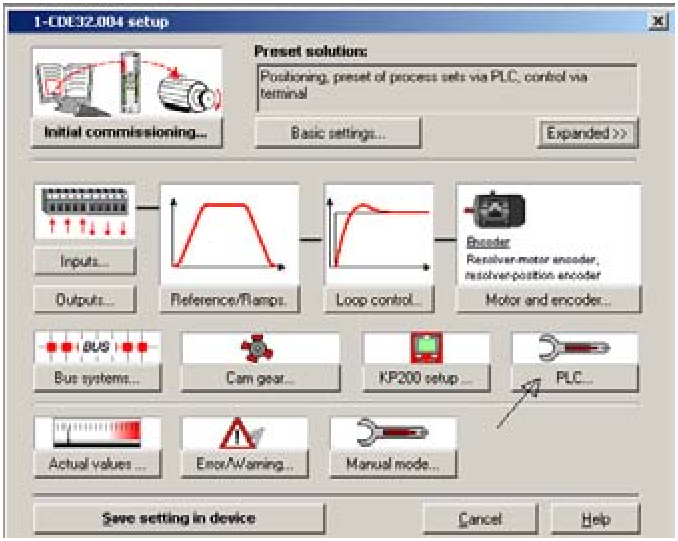
Setting breakpoints in a line in the sequential program

```
Ny BRKPT
```

With activated breakpoints the program processing is interrupted in line Ny (parameter 450 PLCST = BRKPT).

By starting (parameter operation status on "Start" in the PLC window, 450-PLCST = GO) the program processing is continued with the next command line.

Note: Breakpoints can also be set via the user interface of the DRIVEMANAGER.



Switching off the PLC (e.g. via parameter 450 PLCST = OFF) the program processing is ended.

; Example program

```
%P00
N010     NOP                         ; no instruction
N020     SET BRKPT = 1               ; activate breakpoints
N030     SET H000 = 0                ; assign variable
N040     SET H001 = 10               ; assign variable
N050     BRKPT                       ; Breakpoint
N060     SET H000 + 1                ; increment variable
N070     JMP (H000 < H001) N100      ; H000 smaller 10 ?
N080     SET BRKPT = 0               ; deactivate breakpoints
N100     JMP N040                    ; continue incrementing
END
```

With deactivated breakpoints this function is similar to an blank instruction (NOP).

Blank instruction (NOP)

This is an instruction without function, i.e. the program processes the line, but no reaction will occur. The processing requires (as with other commands) computing time.

How to use this function in the sequential program:

```
Ny NOP   Instruction without function
```

Program end (END)

Both the text declaration as well as the actual sequential program must be quit with this command. All subsequently following lines will be ignored. In case of a missing END an error message will be emitted.

How to use this function in the sequential program

```
END      No line number is specified!
```

Setting commands (SET)

Note: The results of calculations etc. are always saved in the left variable.
F001 = 10; F002 = 15, Set F001 - F002;
"-5" is generated in F001

With the help of setting commands a vast variety of operations can be executed in the travel programs:

- Setting of outputs (direct, via flags)
- Setting of flags (direct, indexed, via logic operations, ...)
- Setting, calculation of variables, ...
- Setting, incrementing, decrementing of counters
- Setting and starting of timers
- Access to device parameters (e. g. controller settings, override functions, setpoint tables, etc.)
- Changing of acceleration parameters

Setting a digital output

direct:

```
SET Oppi = 0
SET Oppi = 1
```

via flag:

```
SET Oppi = Mxxx
```

Output image:

```
SET OUTPUT = Hxxx
```

Attention: Only the outputs will be set, which have their function selector FOppi=PLC set.

Setting logic flag

direct:

```
SET Mxxx = 0
SET Mxxx = 1
```

indexed:

```
SET M[Cxx] = 0
SET M[Cxx] = 1
```

via 2. flag:

direct:

```
SET Mxxx = Myyy       assign flag value
```

indexed:

```
SET M[Cxx] = Myyy
```

via logic operation:

```
SET      Mxxx & Myyy     Logic AND
SET      Mxxx | Myyy     Logic OR
SET      Mxxx ^ Myyy     Logic EXCLUSIVE-OR
```

via integer variable

```
SET Mxxx = Hxxx          Assignment of LSB for Hxxx
```

via digital inputs and outputs

```
SET Mxxx = Ippi          assign status input
SET Mxxx = Oppi          assign status output
```

Setting special markers – variables (status variables)

```
SET Mxxx  = STA_ERR      Drive in error status
SET Mxxx  = STA_WRN      Drive in warning status
SET Mxxx  = STA_ERR_WRN  Drive in status error / warning
SET Mxxx  = STA_ACTIV    Control active
SET Mxxx  = STA_ROT_R    Motor rotating clockwise
SET Mxxx  = STA_ROT_L    Motor rotating anti-clockwise
SET Mxxx  = STA_ROT_0    Motor stopped
SET Mxxx  = STA_LIMIT    Limit reached
SET Mxxx  = STA_REF      Setpoint reached
SET Mxxx  = STA_HOMATD   Axis referenced
SET Mxxx  = STA_BRAKE    Drive in braking state
SET Mxxx  = STA_OFF      Drive in de-energized state
SET Mxxx  = STA_C_RDY    Drive in status "Controller ready"
SET Mxxx  = STA_WUV      Warning undervoltage
SET Mxxx  = STA_WOV      Warning overvoltage
SET Mxxx  = STA_WIIT     Warning warning I^2*t
SET Mxxx  = STA_WOTM     Warning motor overtemperature
SET Mxxx  = STA_WOTI     Warning heat sink temperature
SET Mxxx  = STA_WOTD     Warning inside temperature
SET Mxxx  = STA_WIS      Warning apparent current - limit value
SET Mxxx  = STA_WFOUT    Warning output frequency - limit value
SET Mxxx  = STA_WFDIG    Warning setpoint master error
SET Mxxx  = STA_WIT      Warning I*t motor protection
SET Mxxx  = STA_WTQ      Warning torque
SET Mxxx  = STA_INPOS    Position setpoint reached
                         (only with positioning controller
switched on)
```

Setting special flags – variables (control variables)

```
SET ENCTRL = 0 / 1, Mxxx Control off / on (only with control
location PLC)
SET INV = 0 / 1, Mxxx    Invert setpoint
                         (only with speed control, not with
                         endless positioning)
```

```
SET ERR   = 0 / 1, Mxxx  Trigger error
SET ERRRQ = 0 / 1, Mxxx  Reset error
                         Attention: PLC must not be switched off
                         with controller. Observe the control
                         location when switching on via PLC!
SET BRKPT = 0 / 1, Mxxx  Breakpoints off / on
SET ACCR = 0 ... 150%    Scaling of acceleration from 0
                         percent to 150 percent
SET ACCR = 0 ... 150%    Scaling of deceleration
                         from 0 percent to 150 percent
SET HALT = 0/ 1, Mxxx   Stop feed acc. to stop reaction,
                        see 6.2.3 and
                        “Braking the drive (STOP, SET HALT/
BRAKE)”, page 7-38
SET BRAKE = 0/ 1, Mxxx  Trigger quick stop acc. to quick stop
                         reaction, see 6.2.3 and
                        “Braking the drive (STOP, SET HALT/
BRAKE)”, page 7-38
SET EGEARPOS = Hxxx     Set run-in reference encoder
                        increments
SET Hxxx = EGEARPOS     Read run-in reference encoder
                        increments
SET Hxxx = EGEARSPEED   Read reference encoder speed in rpm
```

Indexed assignment of a constant value

```
SET F[Cxxx] = Value
SET H[Cxxx] = Value
SET M[Cxxx] = Value
```

Setting integer variable

direct:

```
SET Hxxx = z
```

indexed:

```
SET H[Cxx] = z
```

with 2. variable:

direct:

```
SET Hxxx = Hyyy
```

indexed:

```
SET H[Cxx] = Hyyy
```

with 2. indexed variable:

```
SET Hxxx = H[Cyy]
```

with 2. floating point variable:

```
SET Hxxx = Fxxx
```

Assignment of a floating point variable with limitation to +/- 2147483647 no rounding with flag:

```
SET Hxxx = Mxxx
```

with counter status:

```
SET Hxxx = Cyy
```

with timer status:

```
SET Hxxx = Zxx
```

via acceleration - direct: 2)

```
SET Hxxx +z           Addition
SET Hxxx -z           Subtraction
SET Hxxx *z           Multiplication
SET Hxxx :z      z ≠ 0 1)Division
SET Hxxx % z          Modulo
```

via displacement with constant:

to the right:

```
SET Hxxx >> z         Division Hxxx by 2^z
```

to the left:

```
SET Hxxx<< z          Multiplication Hxxx with 2^z
```

Calculation via second variable - direct: 2)

```
SET Hxxx + Hyyy       Addition
SET Hxxx - Hyyy       Subtraction
SET Hxxx * Hyyy       Multiplication
SET Hxxx : Hyyy       Hyyy ≠ 0 1)  Division
SET Hxxx % Hyyy       Modulo
```

Calculation via displacement with second variable:

Right:

```
SET Hxxx >> Hyyy      Division Hxxx by 2^Hyyy
```

Left:

```
SET Hxxx << Hyyy      Multiplication Hxxx with 2^Hyyy
```

Calculation by means of absolute-value generation:

```
SET Hxxx = ABS Hyyy
```

[1] `z` or `Hyyy = 0` is not permitted (division by 0)!
(error message will be triggered).

[2] With this operation one must make sure that no value range overflow takes place.

Setting special integer variable

with value of parameter:

direct:

```
SET Hxxx = PARA[n]
```

with value of field parameter:

direct:

```
SET Hxxx = PARA[n,i]
```

with actual values:

direct:

```
SET Hxxx = ACTPOS     Assign actual position value
SET Hxxx = ACTFRQ     Assign actual frequency value (only for U/f)
SET Hxxx = ACTSPEED   Assign actual speed value
SET Hxxx = ACTTORQUE  Assign actual torque
SET Hxxx = ACTCURRENT Assign actual current value
```

with setpoints:

direct:

```
SET Hxxx = REFPOS     Assign position setpoint
```

with input and output functions:

```
SET Hxxx = OSA0       Read value of analog output
                      (0..10.000 = 0V..10V)
SET Hxxx = ISA0       Assign value of analog input 0
                      (0 ... 1.000 = 0V ... 10V).
SET Hxxx = ISA1       Assign value of analog input 1
                      (0 ... 1.000 = 0V ... 10V)
SET Hxxx = Input      Assign input image
SET Hxxx = Output     Assign output image

SET OSA0 = Hxxx       Assign CDB3000 analog output (0..10.000 =
0V..
                      10V).
SET Oppi = 0          Set digital output to Low
SET Oppi = 1          Set digital output to High
SET Oppi = Mxxx       Assign flag value to digital output
```

The function selector of the outputs must be set to PLC.

```
SET REFVAL = Hxxx     Assign setpoint
                      (only for torque/speed control=
SET INPOSWINDOW = HxxxAssign window setpoint reached
                      (only with positioning)
```

1 2 3 4 5 6 7 8 A DE EN

Setting floating point variable

direct:

```
SET Fxxx = f
```

with 2. variable:

direct:

```
SET Fxxx = Fyyy       Assignment of floating point variable
```

indexed:

```
SET F[Cxx] = Fyyy     Indexed assignment
```

with 2. indexed variable

```
SET Fxxx = F[Cxx]           Indexed assignment
```

with 2. integer variable:

```
SET Fxxx = Hxxx             Assignment of integer variables
```

via calculation - direct:

```
SET Fxxx + f                Addition of floating constants
SET Fxxx - f                Subtraction of floating constants
SET Fxxx * f                Multiplication of floating constants
SET Fxxx : f                Division of floating constants
```

Calculation via 2. variable - direct:

```
SET Fxxx + Fyyy             Addition of floating variables
SET Fxxx - Fyyy             Subtraction of floating variables
SET Fxxx * Fyyy             Multiplication of floating variables
SET Fxxx : Fyyy             Division of floating variables
```

Calculation by rounding:

```
SET Fxxx = ROUND Fyyy       Mathematically rounded
                            2.8  ->  3.0     -2.8 -> -3.0
```

Calculation by means of absolute-value generation:

Setting special floating point variable

```
SET Fxxx = ABS Fyyy         Absolute-value generation -2.8 -> 2.8
SET Fxxx = PARA[Hyyy, Hzzz] Assign field parameter value
SET Fxxx = PARA[Hyyy]       Assign parameter value
SET Fxxx = PARA[n, i]       Assign field parameter value
SET Fxxx = PARA[n]          Assign parameter value
SET Fxxx = ACTFRQ           Actual frequency value (only with U/f)
SET Fxxx = ACTSPEED         Actual speed value
SET Fxxx = ACTTOURQUE       Actual torque value
SET Fxxx = ACTTOURQUE       Actual current value
SET Fxxx = ACTPOS           Assign actual position value
SET Fxxx = REFPOS           Assign position setpoint
SET REFVAL= Fxxx            Assign setpoint via
                            floating point variable
                            (only for torque/speed control)
```

Set counter

direct:

```
SET Cxx = d
```

with variable:

```
SET Cxx = Hyyy
```

with counter:

```
SET Cxx = Cyy
```

Incrementing / decrementing counter:

```
SET Cxx + d
SET Cxx - d
```

Incrementing / decrementing counter via variable:

```
SET Cxx + Hyyy
SET Cxx - Hyyy
```

Setting and starting timers

After assigning a timer (time counting element) with a value, this value is automatically reduced by 1 every millisecond, until finally the value of 0 is reached.

The timer Z11 must not be used when working with the command WAIT, because this timer is used to execute the WAIT commands.

direct:

```
SET Zxx = t
```

with variable:

```
SET Zxx = Hyyy
```

The timer value is specified in ms.

Set parameter

with integer variable:

```
SET PARA[n] = Hxxx    Direct specification of parameter number
SET PARA[Hxxx] = Hyyy Specification of parameter number via
                      floating point variable
```

with floating point variable

```
SET PARA[n] = Fxxx    Direct specification of parameter number
SET PARA[Hxxx] = Fyyy Specification of parameter number via
integer variable
```

Note: Saving the sequential program, the parameters and the travelling data into the Flash-EPROM may also be triggered by the program. (SET PARA [150] =1).

Setting field parameters

with integer variable:

```
SET Para [n,i] = Hxxx       Direct specification of parameter
number
                            and index
SET PARA [Hxxx,Hyyy] = Hzzz Specification of parameter number
                            and index via integer variables
```

with floating point variable:

```
SET  PARA [n,i] = Fxxx      Specification of parameter number
                            and index direct
SET PARA [Hxxx, Hyyy] = FxxxSpecification of parameter number
                            and index via integer variables
```

Note: The data type must be observed during read / write operations.
Example: Do not assign floating point values to an integer type parameter (value range violations possible).

Data types	Value range	Function	Suitable for PLC variable
USIGN8	0 ... 255	unsigned	Hxxx, Fxxx
USIGN16	0 ... 65535		
USIGN32	0 ... 4294967295		
INT8	-128 ... 127	Integer, signed	
INT16	-32768 ... 32767		
INT32	-2147483648 ... 2147483647		
INT32Q16	-32767,99 ... 32766,99	32 bit number with standardization 1/65536, i. e. the low-word indicates the fractional digits.	Fxxx
FIXPOINT16	0,00 ... 3276,80	Fixed-point number with standardization 1 /20, i. e. increment value 0.05	
FLOAT32	see IEEE	32 bit floating point number in IEEE-format	

Table 7.1 Data types

Inverting (INV)

The INV-command can be used to logically invert an integer variable, a flag or the status of a digital output. With this e. g. an output with Low-Level is inverted to High-Level, whereby it can be used in the program as a status indicator.

How to use this function in the sequential program:

```
Ny INV Hxxx       Logic inverting of an integer variable
Ny INV Mxxx       Logic inverting of a flag
Ny INV Oppi       Logic inverting of a digital output
```

Travel commands in positioning (GO)

These commands can be used to move the driven positioning axis. These commands must only be used in positioning mode, the setpoint channel must be set to PLC (preset solution with setpoint via PLC). With torque/speed control GO-commands are evaluated as NOP. Effect of the individual positioning modes see chapter 5.2.1.

There are generally five methods to move the axis:

- **Absolute positioning**: Travelling to a certain position
  (GO A ..)
- **Relative positioning**: Travelling over a certain distance
  (GO R ..)
- **Endless positioning:** Travelling with defined speed
  (GO V ...)
- **Start referencing**:
  (**GO 0**)
- **Synchronous travel**: Electronic transmission
  (**GO SYN ..**)

Travelling with or without continuation of program

- with continuation of program (GO ...)

  If this command is submitted within the program, the program will immediately continue with the following program line, after the axis has been started. In this way several commands can be processed parallel to an ongoing positioning.

  If this command is submitted during an ongoing positioning, the travel to the new target position will be continued with the changed speed. The new command is executed immediately, i.e. the position specified in the previous command is no longer approached. Reference for relative positioning is always the last position setpoint.

- without continuation of program (GO W ...)

  With this command the next successive program line is only processed after the actual position has reached the position window. As long as the axis is not in the positioning window - e.g. due to a trailing error - the program is not continued.

  The "W“ is an abbreviation for "Wait", GO W = "go and wait“.

Travelling with continuation

Position or path via variable / speed via variable

```
GO A Hxxx V Hyyy            Absolute travel by value of Hxxx
                            with speed Hyyy
                            (program processing continues)
GO R Hxxx V Hyyy            Relative travel by value of Hxxx
                            with speed Hyyy
                            (program processing continues)
```

Position via variable / speed via parameter

```
GO A Hxxx     Absolute travel by value of Hxxx
              (program processing continues)
GO R Hxxx     Relative travel by value of Hxxx
              (program processing continues)
```

Relative travel commands with continuation must not be processed in a "short" endless loop, as this would lead to a position overflow. See following example:

```
N010 SET H001 = 360
N020 GO R H001
N030 JMP N020
```

Position or path from table

```
GO T[Hxxx]           Travel acc. to table entry
                     (program processing continues)
GO T[Cxx]            Travel acc. to table entry
                     (program processing continues)
GO T[xxx]            Travel acc. to table entry
                     (program processing continues)
```

Travelling without continuation

Position or path via variable / speed via variable

```
GO W A Hxxx V Hyyy  Absolute travel by value of Hxxx
                    with speed Hyyy
                    and wait for further program processing until
                    target position is reached
GO W R Hxxx V Hyyy  Relative travel by value of Hxxx
                    with speed Hyyy
                    and wait for further program processing until
                    target position is reached
```

Position via variable / speed via parameter

```
GO W A Hxxx       Absolute travel by value of Hxxx
                  and wait for further program processing until
                  target position is reached
GO W R Hxxx       Relative travel by value of Hxxx
                  and wait for further program processing until
                  target position is reached
```

Position or path from table

```
GO W T[Hxxx]      Travel acc. to table entry Hxxx,
                  wait until position is reached
GO W T[Cxxx]      Travel acc. to table entry Cxxx,
                  wait until position is reached
GO WT[xxx]        Travel acc. to table entry,
                  wait until position is reached.
```

Referencing

Referencing is performed using the specified referencing type and the associated speeds (727 HOSPD).

If this command is submitted within a program, the next successive set will only be effective, after referencing has been completed.

```
GO 0             Referencing is performed,
                 in dependence on the method specified in parameter
730
                 depending on software status
GO 0 + Hxxx      Referencing is performed, position
                 0 results from this. Thereafter this zero
                 position is set to the value specified in Hxxx.
```

The GO 0 - command is flank triggered. Referencing can therefore only be stopped by a cancellation condition (e. g. STOP B).

The status of referencing can be monitored with the special flag STA_HOMATD:

Example for referencing with status query:

```
N010 SET H000 = 30                ; (30 degree zero offset)
N020 GO 0 + H000
N030 JMP (STA_HOMATD = 1) N050    ; HOMATD = 1 -> Reference point
                                  ;               defined
                                  ; HOMATD = 0 -> Reference point
                                  ;               not defined
N040 JMP N030                     ; Return in query
N050 ....                         ; further program run
```

after referencing the thus detected zero position will have the value 30° assigned (in the device)

Endless travel

via variable:

```
GO V Hxxx       Hxx= Index of variables with speed value
```

The sign of the value in Hxxx determines the travel direction.

Speed synchronism

Switching on synchronous travel:

```
GOSYN 1
```

Switching off synchronous travel:

```
GOSYN 0
```

With speed synchronism (configuration of input see chapter 6.2.4) the speed of the reference encoder in rpm is switched to the setpoint structure. The speed acceleration ramps (see chapter 6.2) are active, i.e. "soft" coupling and decoupling.

Note: Speed synchronism is only active with speed control.

The speed setpoint of the reference sensor always refers to the motor shaft. When using a gearbox on motor and target and the drive shaft speed is to be determined by the reference sensor, the gearbox ratio must be parameterized in the reference sensor configuration.

Angular synchronism (electronic transmission)

With angular synchronism (configuration of input see chapter 6.2.4) the drive controller converts the incoming square wave pulses of a reference encoder directly to a position setpoint and approaches this point in a position controlled manner.

The configuration of the reference encoder input is described in detail in chapter 6.2.4.

Switching on synchronous travel:

```
GOSYN 1
```

Switching off synchronous travel:

```
GOSYN 0
```

After switching on synchronous travel with the command GOSYN 1 the sequential program is immediately continued with the next successive set.

Note: Switching synchronous travel on / off occurs abrupt, without limitation of the axis dynamics by ramps. Soft coupling / decoupling on a rotating leading axis is not possible.

The reference sensor position refers to the motor shaft. The unit is always in increments (65536 Incr = 1 motor revolution). If the reference sensor position is to be directly related to the output shaft, the transmission ration must be entered for the reference sensor. A transmission ratio in the standardizing assistant will be ignored when using the reference sensor.

Example for the CDB3000:

System structure:

- HTL reference sensor as setpoint specification connected to terminal X2 on CDB3000.
- CDB3000 with gear motor (i = 56 /3)
- A transmission ratio of 56/3 was entered in the standardizing assistant (under basic settings).

Conclusions:

➢ with a reference sensor transmission ratio of 1/1 the reference sensor setpoint refers to the motor shaft of the gear motor.

➢ with a reference sensor transmission ratio of 56/3 the reference sensor setpoint refers to the output shaft of the gear motor.

Position and speed of the reference encoder can be read with the help of special PLC variables:

```
SET Hxxx = EGEARPOS; Reading the reference encoder position in
increments
```

The submitted reference encoder increments are the actual increments of the reference encoder, multiplied with the transmission ratio of the reference encoder.

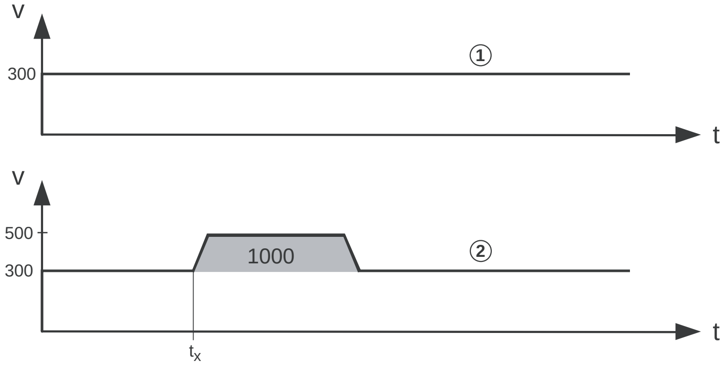
```
SET Hxxx = EGEARSPEED; Reading the reference encoder speed in rpm
```

The output is the reference encoder speed, multiplied with the transmission ratio of the reference encoder.

The position of the reference encoder can also be changed via the PLC:

```
SET EGEARPOS = Hxxx; Setting the reference encoder position in
increments
```

A GOR-command (relative positioning) during synchronous travel results in a superimposed positioning.



(1) leading axis,   (2) following axis

Fig. 7.4   Relative positioning during synchronous travel. t_x=time of command GO R H000 V001 with H000 = 1000 and H001 =200

A GOA-command (absolute positioning) during synchronous travel aborts this travel. The axis continues travelling with the transmitted travelling speed and performs the requested absolute positioning, by observing the set ramps.

GO A and GO R positions, as always, refer to the output shaft. The required transmission ratio can be configured through the standardizing assistant.

Path optimized positioning of a round table

The target position is specified as an absolute value and the positioning controller moves the axis in the direction with the shortest path. Relative movements do not take place in a path optimized way. See also chapter 5.2.3.

This type of positioning assumes that an endless travel path has been selected. For the round table function the settings in the travel profile are decisive. If round table function, direction optimization and length of circumference are specified there under, the commands will be executed in a path optimized manner.

Braking the drive (STOP, SET HALT/BRAKE)

Various commands with and without controller stop are available to brake the drive.

Stop feed

With the command

```
SET HALT = 1
```

the drive is braked to standstill according to the reaction "Stop Feed" (see chapter 6.2.3). The drive thus remains energized.

With the command

```
SET HALT = 0
```

the drive is set in motion again with the previously specified travel set. The braking process can be terminated at any time.

Quick stop

With the command

```
SET BRAKE = 1
```

the drive is braked according to the reaction "Quick Stop" (see chapter 6.2.3). The drive controller is in "Quick stop" system state. The controller is now switched off, if switching off has been parameterized in the quick stop reaction and if it has been enabled via PLC (SET ENCTRL = 1, control location PLC).

With the command

```
SET BRAKE = 0
```

the quick stop condition is terminated. This command must always be executed before the drive can be switched on again. Termination of the quick stop and return to the previous travel set is possible, as long as the drive is energized.

Braking with deceleration ramp (only positioning)

For normal braking with programmed deceleration ramp the command

```
STOP B
```

is available. The braking process cannot be aborted. The travel set that had been valid when the STOIP command was triggered, becomes invalid. The command is valid with positioning.

Braking with quick stop ramp (only positioning)

For quick braking with quick stop ramp the command

```
STOP M
```

is available. The braking process cannot be aborted. The travel set that had been valid when the STOIP command was triggered, becomes invalid. The command is valid with positioning.

Emergency stop (speed = 0) and shut-down of control (only positioning)

for quickest possible braking (speed setpoint=0) and subsequent shut down of the control the command

```
STOP 0
```

is available. The control is only switched off if it had been switched on via PLC (SET ENCTRL = 1, control location PLC).

1
2
3
4
5
6
7
8
A
DE
EN

The braking process cannot be aborted. The travel set that had been valid when the STOIP command was triggered, becomes invalid. The command is valid with positioning.

Wait commands (WAIT)

Time

This command can be used to realize a certain time delay in milliseconds. After expiration of this time the program will continue with the next successive program line. The WAIT command is executed via the timer Z11.

direct:

```
WAIT d
```

via variable:

```
WAIT Hxxx
```

Axis status

The program is continued, if the following condition is fulfilled.

Position window reached

```
WAIT REF      Actual position in position window 1)
```

Axis stopped:

```
WAIT ROT_O    Position setpoint = Target position 2)
```

```
1)   Positioning finished,
     Output "Axis in position" will be set
2)   Positioning mathematically finished,
```

Parameter write access

```
WAIT PAR      Wait until parameter write access has taken place.
```

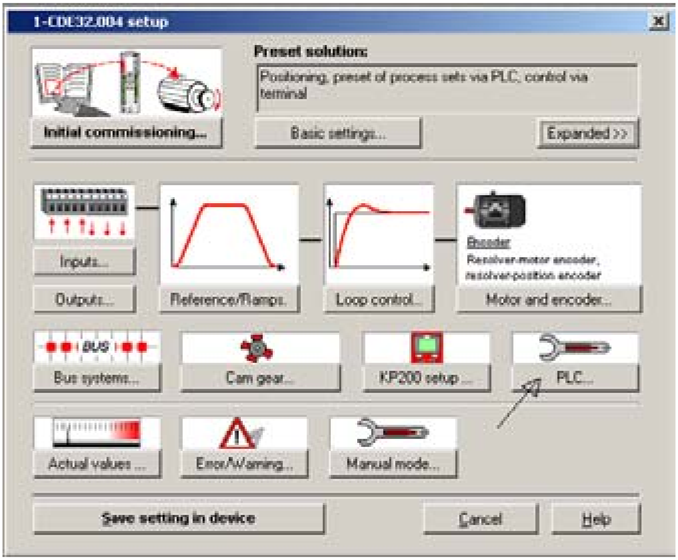
If the parameter write access is mandatory for the further processing of the program, a WAIT PAR should be inserted after the parameter assignments.

Example program

```
%P00
N010  SET H000 = 1           ; Assign value 1 to variable H000
N020  SET PARA[460,1] = H000 ; Write (field) parameter 460,
                             ; Index 1
N030  SET PARA[460,2] = H000 ; Write (field) parameter 460,
                             ; Index 2
N040  SET PARA[270] = H000   ; Write parameter 270
N050  WAIT PAR               ; Wait with program processing until
                             ; all parameter write access
                             ; have taken place
END                          ; End of program
```

7.4 PLC control and parameters

An uncomplicated setting of the specified PLC control parameters enables the PLC function mask (extended main window -> PLC or via "Basic settings/PLC with the corresponding PLC presetting):



1-CDE32.004 setup

Preset solution:

Positioning, preset of process sets via PLC, control via terminal

Fig. 7.5 DRIVEMANAGER - PLC function mask

7.4.1 PLC variables

All PLC variables are shown by means of parameters. These parameters can be edited via the DRIVEMANAGER in a PLC function mask (see Fig. 7.5).

DRIVEMANAGER	Meaning	Value range	Changing ONLINE	Parameter
Integer variables (32 bit)	Integer variables are integer numerical values. In combination with floating point variables or parameters the digits after the decimal point are not taken into consideration. Rounding will also not take place. Access in the sequential program H000...H127	2^{-31} to 2^{31}	yes	460-PLC_H (_PLCP)
Flag (0/1)	Access in the sequential program M000...M255	0/1	yes	461-PLC_M (_PLCP)
Timer (32 bit)	Time base 1 ms Access in the sequential program Z00...Z11 Timers are set to a certain value and run back to 0.	0 to 2^{32}	yes	462-PLC_Z (_PLCP)
Counter for indexed addressing (8 bit)	Access in the sequential program C00...C10	0 to 65535	yes	463-PLC_C (_PLCP)
Image of the digital outputs (bit coded)	The image can also be written in the program as special variable OUTPUT. OSD00-OSD02 Bit 0 - Bit 2 OED00-OED03 Bit 4 - Bit 6 OV00-OV01 Bit 7 - Bit 8 In order to set outputs from within the program, the corresponding function selector must be set to FOppi = PLC.		yes	464-PLC_O (_PLCP)
Floating point variables	Access in the sequential program F000...F127	-3,37x10^{38} to 3,37x10^{38}	yes	465-PLC_F (_PLCP)
Image of digital and analog inputs (bit coded)	The image can also be written in the program as special variable INPUT. ISD00-ISD03 Bit 0 - Bit 3 IED00-IED07 Bit 4 - Bit 11 ISA00 - ISA01 Bit 12 - Bit 13		read only	466-PLC_I (_PLCP)

Table 7.2 PLC Variables and flags

7.4.2 PLC control parameters

The PLC control parameters enable a flexible configuration of the PLC-program or of its sequence.

DRIVEMANAGER	Meaning		Changing ONLINE	Parameter
Name of the PLC program (Project name)	The project name is defined when generating the sequential program (text declaration). The name directly designates the text declaration file (project name.txt) (max. 32 characters without special characters, spaces will be ignored)		yes	468- PLCPJ (_PLCC)
Operating status of the sequencing control	This parameter enables the starting/stopping (depending on parameter 452-PLCCT=PARA) or indicates the current operating status of the sequential program.		yes	450-PLCST (_PLCC)
	OFF (0)	PLC program sequence shut-down / switched off		
	GO(1)	Start PLC program sequence / in progress		
	BRKPT(2)	PLC program sequence interrupted The GO command continues the operation. The program processing can be interrupted (BRKPT) or ended (OFF) with the parameter at any time, irrespective of the control location. With GO the processing of the program can be resumed from the cancellation line, as long as the control location is still valid (e.g. terminal still set). If this conditions is no longer fulfilled, the parameter is set to OFF.		
Current program line	Shows the currently processed program line. The line number is also visible in the digital oscilloscope.		read	451-PLCPL (_PLCC)

Table 7.3 PLC control parameters

DriveManager	Meaning		Changing ONLINE	Parameter
Start conditions of the sequencing control	Parameter PLCCT defines the location from which the sequential program is started.		yes	452-PLCST (_PLCC)
	TERM(0)	PLC start via input The function selector for an input must be set to Fixxx = PLCGO. (0 -> Program stopped, 1 -> Program started)		
	PARA(1)	PLC start via parameter "Operation status" Manual change of operation status PLCST		
	AUTO(2)	Automatic PLC start when starting the device, parameter "Operation status" is set to GO and serves as status indicator		
	CTRL(3)	PLC start together with activation of controller PLC start together with deactivation of controller		
	BUS(4)	PLC is started via field bus in EasyDrive-ProgPos control word with the bit "Start PLC". When resetting the bit the PLC-sequence is directly terminated by jumping to line 0.		
Program stop in line x (breakpoint)	The program is interrupted at the line specified under PLCBN; the parameter 450-PLCST changes to status BRKPT. The program is restarted with 450-PLCST=GO(1).		yes	455-PLCBN (_PLCC)
Start with program line (0 = first program line).	Processing of the program starts with the line specified in PLCSN. This is very sensible, if a program contains different independent routines.			456-PLCSN (_PLCC)

Table 7.3 PLC control parameters

7.5 PLC program examples

The examples in this chapter are solely intended as programming exercises. Neither the problem definitions, nor the suggested solutions have been checked under the aspects of safety.

The examples shall demonstrate the possible solutions with the integrated sequencing control and what a typical program section could look like. A preset solution, which access the PLC, must be set up, e.g. "PCT_3 (18) positioning, travel set specification via PLC, control via terminal“.

The specified values for path unit, speed and acceleration are only examples and should strictly be adapted to the application described hereunder.

Basis for these examples is a gear motor with a rated speed of 1395 min^{-1} and a transmission ratio of ü=9,17.

Lust Antriebstechnik GmbH therefore does not assume any responsibility and will not accept any liability for damage resulting from the type of use of this programming material or of parts thereof.

The numerical values for path. speed and acceleration solely refer to the programming units specified in the positioning controllers.

7.5.1 Conveyor belt

After the start the conveyor belt drive shall advance the belt by 1m (corresponds with 10 revolutions of the output shaft) with a speed of 35 mm/s. After a waiting time of 5 s the process shall be repeated, until the input is reset. (Input used ISD03).

Setting units and standardization in the standardization assistant:

Position:	mm
Speed	mm/s
Acceleration:	mm/s^2
Feed constant:	1000 mm corresponds with 10 revolutions of the output shaft
Gear:	Motor shaft revolutions 917 Output shaft revolutions 100

Adapting the travel profile:

Max. speed:	250 mm/s
Max. starting acceleration:	50 mm/s^2
Max. braking acceleration:	50 mm/s^2

The example program can be transferred to the controller, after referencing has been parameterized as described in chapter 5.2.4.

```
%TEXT (Conveyor Belt)
DEF H001 = Path
DEF H002 = Speed
END

%P00
N001 SET H001 = 1000    ; Path in mm
N002 SET H002 = 35      ; Speed in mm/s

N010 GO 0               ; Perform referencing
N020 JMP (IS03=0) N020  ; continue, if input = high
N030 GO W R H001 V H002 ; Travel to position direction with 35
mm/s
N040 WAIT 5000          ; Wait 5 s
N050 JMP N020           ; Restart cycle
END
```

7.5.2 Absolute positioning

The fourth position is to be approached with a speed of v=80 mm/s absolute, followed by a wait period of always 1 s. The travel back to initial position is to take place with three times the speed (240mm/s).

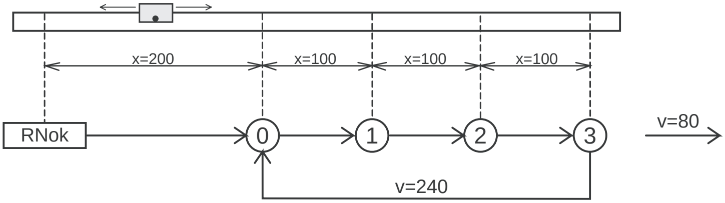


Fig. 7.6 Approach position

Setting units and standardization in the standardization assistant:

Position:	mm
Speed	mm/s
Acceleration:	mm/s^2
Feed constant:	100 mm corresponds with 1 revolution of the output shaft
Gear:	Motor shaft revolutions 917 Output shaft revolutions 100

Adapting the travel profile:

Max. speed:	250 mm/s
Max. starting acceleration:	50 mm/s^2
Max. braking acceleration:	50 mm/s^2

The example program can be transferred to the controller, after referencing has been parameterized as described in chapter 5.2.4.

Positions and speeds are directly transferred as values, the specification of the acceleration takes place according to the machine parameters.

```
; Standardization in s=mm and v=mm/s
%TEXT (Absolute Positioning)
DEF H000 = Position_0
DEF H001 = Position_1
DEF H002 = Position_2
DEF H003 = Position_3
DEF H004 = Speed_v1
DEF H005 = Speed_v2
END

%P00
N001 SET H000 = 200
N002 SET H001 = 300
N003 SET H002 = 400
N004 SET H003 = 500
N005 SET H004 = 80
N006 SET H005 = 240

N020 GO 0                  ; Referencing
N030 GO W A H000 V H004    ; Approach initial position
N040 WAIT ROT_0            ; Wait until axis has stopped
N050 WAIT 1000             ; Wait 1 s
N060 GO W A H001 V H004    ; Approach position 1 and wait until
                           ; axis has stopped
N070 WAIT 1000
N080 GO W A H002 V H004    ; Position 2
N090 WAIT 1000
N100 GO W A H003 V H004    ; Position 3
N110 WAIT 1000
N120 GO W A H000 V H005    ; return to initial position

N130 JMP N050
END
```

7.5.3 Relative positioning

In the previous example the axis has always travelled further by the same distance, this opens the possibility for a solution with relative positioning. A counter always holds the actual position; units and standardization see previous example.

```
%TEXT (Relative Positioning_1)
DEF H000 = Position_0
DEF H001 = Distance_between_positions
DEF H002 = Speed_v1
DEF H003 = Speed_v2
END

%P00
N001 SET H000 = 200      ; Position 0 in mm
N002 SET H001 = 100      ; Distance between two positions in mm
N005 SET H002 = 80       ; Speed in mm/s
N006 SET H003 = 240      ; Speed in mm/s

N010 GO 0                ; Referencing
N020 GO W A H000 V H002  ; Approach initial position and wait
N030 SET C00 = 0         ; Set counter = 0
N040 WAIT 1000
N050 GO W R H001 V H002  ; Approach next position
N060 SET C00+1           ; Count position counter
N070 WAIT 1000
N080 JMP (C00 != 3) N050 ; Position 3 not yet reached
N090 GO W A H000 V H003  ; return to initial position
N100 JMP N030
END
```

The solution is even simpler and more elegant when doing without the counter and the comparison is made with the position setpoint (SP).

```
%TEXT (Relative Positioning_2)
DEF H000 = Position_0
DEF H001 = Distance_between_positions
DEF H002 = Speed_v1
DEF H003 = Speed_v2
END

%P00
N001 SET H000 = 200           ; Position 0 in mm
N002 SET H001 = 100           ; Distance between two positions in
mm
N003 SET H002 = 80            ; Speed in mm/s
N004 SET H003 = 240           ; Speed in mm/s
N005 SET H004 = 500           ; Position setpoint 3, used for
comparison

N010 GO 0                     ; Referencing
N020 GO W A H000 V H002       ; Approach initial position and wait
N030 WAIT 1000

N040 GO W R H001 V H002       ; Approach next position
N050 WAIT 1000
N060 JMP (REFVAL < H004) N040 ; Position 3 not yet reached

N070 GO W A H000 V H003       ; return to initial position

N080 JMP N030
END
```

7.5.4 Sequential program

Here the positioning controller is used as a freely programmable sequencing control for a speed profile.

An endless conveyor belt is operated with two speeds. The belt is to be stopped when a target position (≥ 10000) has been reached. The cycle is repeated by a new release input. In order to maintain the structure clear, sub-programs are used. The main program takes over the initialization and call up the sub-programs 1 to 3 in an endless loop.

Parameterization of inputs (DRIVEMANAGER):	IS00	Start(1) = Start of control
	IS01	PLC (35) = Input can be used in sequential program
	IS02	PLC (35) = Input can be used in sequential program
	IS03	/HALT (Feed release, must have High-Level)

Input (Program):	ISD01	Selection of speed 0 = v1 / 1 = v2
	ISD02	Release
Output (Program)	OSD00	Target position reached

Setting units and standardization in the standardization assistant:

Position:	Degree
Speed	Degree/s
Acceleration:	Degrees/s^2
Feed constant:	360° corresponds with 1 revolution of the output shaft
Gear:	Motor shaft revolutions 917 Output shaft revolutions 100

Adapting the travel profile:

Max. speed:	900 degree/s
Max. starting acceleration:	320 Degrees/s^2
Max. braking acceleration:	320 Degrees/s^2

The example program can be transferred to the controller, after referencing has been parameterized as described in chapter 5.2.4.

```
%TEXT (Sequencing control)
DEF H000 = Speed
DEF H001 = Position
END

%P00                    ; Main program

N005 GO 0               ; Perform referencing
N010 SET M000 = 1       ; Flag = 1:
                        ; Axis is not to be started
N015 SET M001 = 0       ; Flag = 0: Axis is not moving
N020 SET H001 = 10000   ; Target position for comparison

N025 CALL N045          ; Sub-program query inputs
N030 CALL N080          ; Sub-program start axis
N035 CALL N105          ; Sub-program position comparison
N040 JMP N025           ; Repeat
```

; Sub-program 1: Query inputs

```
N045 JMP (M001 = 1) N075; If drive is in motion, jump to RET
N050 JMP (IS02 = 0) N075; no query
N055 SET M000 = 0       ; Start took place, set flag = 0

N060 SET H000 = 300     ; Set speed 1
N065 JMP (IS01 = 0) N075; Speed 1 selected
N070 SET H000 = 600     ; Speed 2 selected + set
N075 RET
```

; Sub-program 2: Start axis

```
N080 JMP (M000 = 1) N100
N085 GO R H001 V H000   ; Axis starts with
                        ; speed H000, target position H001
N090 SET M000 = 1       ; Release detected, reset flag
N095 SET M001 = 1       ; Drive in motion
N100 RET
```

; Sub-program 3: Position comparison

```
N105 JMP (REF = 1) N120
N110 SET OS00 = 0
N115 JMP N135
N120 SET M000 = 1
N125 SET M001 = 0       ;Drive stopped
N130 SET OS00 = 1
N135 RET

END
```