Description of Functions 03/2004 Edition

sinumerik

SINUMERIK 840D/840Di/810D
Synchronized Actions

SIEMENS

SIEMENS

SINUMERIK 840D/840Di/810D

Synchronized Actions

Description of Functions

Valid for

Control	*Software Version*
SINUMERIK 840D powerline	7
SINUMERIK 840DE powerline	7
SINUMERIK 840Di	3
SINUMERIK 840DiE (export version)	3
SINUMERIK 810D powerline	7
SINUMERIK 810DE powerline	7

03.2004 Edition

SINUMERIK® Documentation

Printing history

Brief details of this edition and previous editions are listed below.

The status of each edition is shown by the code in the “Remarks” column.

Status code in the “Remarks” column:

A . . . . . New documentation.
B . . . . . Unrevised reprint with new order no.
C . . . . . Revised edition with new status.
If factual changes have been made on the page in relation to the same software version, this is indicated by a new edition coding in the header on that page.

06.94	6FC5 297–0AC30–0BP0	**A**
08.94	6FC5 297–0AC30–0BP1	**C**
02.95	6FC5 297–2AC30–0BP0	**C**
04.95	6FC5 297–2AC30–0BP1	**C**
09.95	6FC5 297–3AC30–0BP0	**C**
03.96	6FC5 297–3AC30–0BP1	**C**
08.97	6FC5 297–4AD40–0BP0	**A**[1)]
12.97	6FC5 297–4AD40–0BP1	**C**
12.98	6FC5 297–5AD40–0BP0	**C**
08.99	6FC5 297–5AD40–0BP1	**C**
04.00	6FC5 297–5AD40–0BP2	**C**
10.00	6FC5 297–6AD40–0BP0	**C**
09.01	6FC5 297–6AD40–0BP1	**C**
11.02	6FC5 297–6AD40–0BP2	**C**
03.04	6FC5 297–7AD40–0BP0	**C**

1) This document replaces the S5 function described for older software versions in the “Description of Functions: Extended Functions” brochure.

This book is part of the documentation on CD-ROM **(DOCONCD)**

Edition	**Order No.**	**Remarks**
03.04	6FC5 298–7CA00–0BG0	**C**

Further information is available on the Internet under:
http://www.siemens.com/motioncontrol

This publication was produced with Interleaf V7.

Other functions not described in this documentation might be executable in the control. However, no claim can be made regarding the availability of these functions when the equipment is first supplied or for service cases.

We have checked that the contents of this document correspond to the hardware and software described. Nonetheless, differences might exist and therefore we cannot guarantee that they are completely identical. The information contained in this document is, however, reviewed regularly and any necessary changes will be included in the next edition. We welcome suggestions for improvement.

Subject to changes without prior notice

Order No. 6FC5 297-7AD40-0BP0
Printed in Germany

Siemens Aktiengesellschaft

Preface

Notes for the reader

The SINUMERIK documentation is organized on three levels:

- General documentation
- User documentation
- Manufacturer/service documentation

This document is designed for machine tool manufacturers. The brochure contains a detailed description of the scope of functions offered by SINUMERIK 840D/810D controllers.

The function descriptions are only valid for the specific software version or up to the software version specified. You should request valid function descriptions for new software versions. Old function descriptions are only partly applicable for new software versions.

For more detailed information on SINUMERIK 840D/840Di/810D publications and other publications covering all SINUMERIK controls (e.g. universal interface, measuring cycles, etc.) please contact your local Siemens office.

Note

It may be possible to run functions that are not described in this document in your controller. This does not, however, represent an obligation to supply such functions with a new control or when servicing.

Hotline

If you have queries regarding the control system, please contact the following hotline:

A&D Technical Support Phone: ++49-(0)180-5050-222
Fax: ++49-(0)180-5050-223
Email: adsupport@siemens.com

If you have queries regarding documentation (suggestions, corrections), please send a fax to the following fax address:
Fax: ++49-(0)9131-98-2176
Email: motioncontrol.docu@erlf.siemens.de

Fax form: See the feedback page at the end of the document

Internet address for SINUMERIK

http://www.siemens.com/motioncontrol

SINUMERIK 840D powerline

Improved-performance variants

- SINUMERIK 840D powerline and
- SINUMERIK 840DE powerline

will be available from 09.2001 onwards. For a list of available **powerline** modules, please refer to Section 1.1 of the Hardware Description /PHD/.

SINUMERIK 810D powerline

Improved-performance variants

- SINUMERIK 810D powerline and
- SINUMERIK 810DE powerline

will be available from 12.2001 onwards. For a list of available **powerline** modules, please refer to Section 1.1 of the Hardware Description /PHC/.

Objective

This document describes the synchronized actions function for SINUMERIK 840D SW 4 and later and for SINUMERIK 810D SW 2 and later. It replaces the S5 function described for older software versions in the “Description of Functions: Extended Functions” brochure.

The function descriptions provide the information required for configuration and installation.

Target groups

The information contained in the function descriptions is designed for:

- Design engineers
- PLC programmers creating the PLC user program with the signals listed
- Start-up engineers once the system has been configured and set up
- Maintenance personnel inspecting and interpreting status signals and alarms

!

Important

This document is valid for the following controls:

- SINUMERIK 840D powerline, software version 7
- SINUMERIK 810D powerline, software version 7
- SINUMERIK 840Di, software version 3

Equivalent NC software versions

The SW versions indicated in this document refer to the SINUMERIK 840D and 810D controls. For detailed information about which function is enabled for which controller, see /BU/, Catalog NC 60 (this information is not provided in this document). Equivalents are as follows:

Table 1-1 Equivalent software version

SINUMERIK 840D powerline		SINUMERIK 810D powerline	SINUMERIK 840Di
7.1	Corresponds to	7.1	3.1

Symbols

Danger

Indicates an imminently hazardous situation which, if not avoided, **will** result in death or serious injury or in substantial property damage.

Warning

Indicates a potentially hazardous situation which, if not avoided, **could** result in death or serious injury or in substantial property damage.

Caution

Used with the safety alert symbol indicates a potentially hazardous situation which, if not avoided, **may** result in minor or moderate injury or in property damage.

Caution

Used without safety alert symbol indicates a potentially hazardous situation which, if not avoided, **may** result in property damage.

Notice

Used without the safety alert symbol indicates a potential situation which, if not avoided, **may** result in an undesirable result or state.

Important

Important indicates an important or especially relevant item of information.

Note

This note contains additional important information.

Contents

Notes

1 Brief Description

Definition of synchronized actions

Motion-synchronous actions (or “synchronized actions” for short) are instructions programmed by the user, which are evaluated in the interpolation cycle of the NCK in synchronism with part program execution. If the condition programmed in the synchronized action is fulfilled or if none is specified, then actions assigned to the instruction are activated in synchronism with the remainder of the part program run.

Applications

The following selection from the wide range of possible applications indicates how actions programmed in synchronized actions can be usefully employed.

- Output of auxiliary functions to PLC
- Writing and reading of real-time variables
- Positioning of axes and spindles
- Activation of synchronous procedures such as:
  - Readin disable
  - Deletion of distance-to-go
  - End preprocessing stop
- Activation of technology cycles
- Online calculation of function values
- Online tool offsets
- Activation/deactivation of couplings/coupled motion
- Take measurements
- Enabling/disabling of synchronized actions

All possible applications of this function are described in the “Detailed Description” chapter.

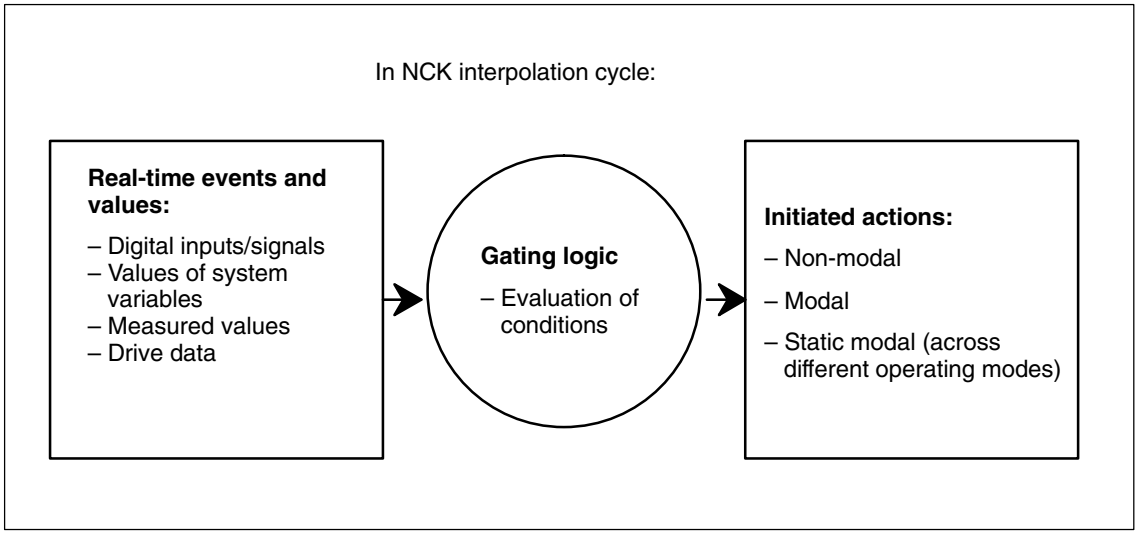


Fig. 1-1 Schematic diagram of synchronized actions

For details of how to program synchronized actions, please see

Reference: /PGA/, Programming Guide Advanced

The following chapters describe:

- Functional conditions for synchronized actions (Chapter 2),
- The required machine data (Chapter 4),
- Example applications (Chapter 6).

Note

This Description of Functions applies to the functionality provided in SW 5. The functions of synchronized actions in SW versions up to and including 3 are described in:
References: /FB/, S5, “Synchronized Actions”

■

2 Detailed Description

2.1 Components of synchronized actions

Structure of a synchronized action

Component:	Validity, identification number	Frequency	G code for cond. and action	Condition	Action code word (fixed)	G code for action	Action or technology cycle see 2.5
Example:	IDS=1	EVERY	G70	$AAA_IM[B] > 15	DO	G71	POS[X]= 100

The components of a synchronized action, i.e.:

- Validity:
  - With identification number
  - Without identification number
- Frequency
- G code for condition and action (SW 5 and later)
- Condition
- G code for actions (SW 5 and later)
- Action(s)/Technology cycle

are explained individually below.

Validity, ID number

There are three possible for defining the scope of validity of a synchronized action:

- No status
- ID
- IDS

No specified validity

Synchronized actions that have no specified validity have a non-modal action, i.e. they apply only to the next block.

Non-modal synchronized actions are operative only in **AUTOMATIC** mode.

In **SW 6.1 and later**, non-modal synchronized actions are active modally for all preprocessing stop blocks (incl. implicitly generated ones) and for implicitly generated intermediate blocks.

ID

Synchronized actions with validity identifier **ID** are **modally** active in subsequently programmed blocks. They are operative only in **AUTOMATIC** mode.

Limitation:

- ID actions remain operative only until another synchronized action with the same identification number is programmed
- Until they are canceled with CANCEL(i), see Subsection 2.5.1.

IDS

Static synchronized actions that are programmed with keyword "**IDS**" are active **in all operating modes**. They are also referred to as static synchronized actions. Option.

Synchronized actions programmed with ID or IDS are deleted from the part program.

Identification numbers

For modal synchronized actions (ID, IDS) identification numbers between 1 and 255 are allocated. They are important for the functions of mutual coordination of synchronized actions. See Subsection 2.5.1. Modal/static synchronized actions with identification numbers between 1 and 64 can be disabled and enabled from the PLC. See Subsection 2.6.1.

Unique identification numbers must be allocated in the channel.

Applications for static synchronized actions:
– AC grinding (also active in JOG mode)
– Gating logic for Safety Integrated
– Monitoring functions, reaction to machine states in all operating modes
– Optimization of tool change
– Cyclic machines

Examples:

IDS=1 EVERY $A_IN[1]==1 DO POS[X]=100	All operating modes
ID=2 EVERY $A_IN[1]==0 DO POS[X]=0	AUTOMATIC

Note

The following actions are operative only in AUTOMATIC mode when the program is running:
STOPREOF,
DELDTG

Frequency

Keywords (see table) are programmed to indicate how often the subsequently specified condition must be scanned and the associated action executed if the condition is fulfilled. These keywords are an integral component of the synchronized action condition.

Table 2-1 Effect of frequency keywords

Keyword	Scanning frequency
None	If no scanning frequency is programmed, then the action is executed **cyclically** in every interpolation cycle.
WHENEVER	The associated action/technology cycle is executed cyclically in every interpolation cycle provided that **the condition is fulfilled**.
FROM	If the **condition has been fulfilled once**, the action/technology cycle is executed **cyclically** in every interpolation cycle for as long as the synchronized action remains active.
WHEN	As soon as the **condition has been fulfilled**, the action/technology cycle is executed **once**. Once the action has been executed a single time, the condition is no longer checked.
EVERY	The action/technology cycle is activated **once** if the condition if fulfilled. The action/technology cycle is executed every time the **condition changes from the "FALSE" to the "TRUE" state**. In contrast to keyword WHEN, checking of the condition continues after execution of the action/cycle until the synchronized action is deleted or disabled.

For details of technology cycles, please see Section 2.5.

Deletion

Deselecting (deleting) an active synchronized action from the part program with **CANCEL** has no effect on the active action. Positioning motions are completed as programmed. The CANCEL command can be programmed to delete a modal or static synchronized action. If a synchronized action is deleted while the positioning axis motion it has initiated is still in progress, the positioning motion continues until properly executed. A channel stop also cancels the positioning movement from synchronized actions/technology cycles.

G code for condition and action

In **SW 5** and later, G codes can be programmed in synchronized actions. This allows defined settings to exist for the evaluation of the condition and the action/technology cycle to be executed, independent of the current part program status. It is necessary to separate the synchronized actions from the program environment, because synchronized actions are required to execute their actions at any time from a defined initial state as a result of fulfilled trigger conditions.

Applications:
Definition of the systems of measurement for condition evaluation and action through G codes G70, G71, G700, G710.

Note

In **SW 5**, the use of the G codes in synchronized actions is limited to these 4 G codes.

A G code specified for the condition is valid for the evaluation of the condition **and** for the action if no separate G code is specified for the action.

Only one G code of the G code group may be programmed for each part of the condition.

Conditions

Execution of actions/technology cycles can be made dependent upon a condition (**logical expression**).

The condition is checked in the interpolation cycle. If no condition is programmed, the action is performed once in every IPO cycle. In SW version 3 and earlier, two conditions are permitted, i.e. the comparison of a real-time variable with an expression calculated during preprocessing or the comparison of two real-time variables.
Examples:
WHENEVER **$AA_IM[X] > 10.5*SIN(45**) DO .... or
WHENEVER **$AA_IM[X] > $$AA_IM[X1]** DO ...

An additional condition is available in SW 4 and later, i.e. the linking of comparisons using Boolean operations. Boolean operators in NC language may be used for this purpose:

NOT, AND, **OR**, **XOR, B_OR, B_AND, B_XOR, B_NOT.**

Examples:
WHENEVER ($A_IN[1]==1) **OR** ($A_IN[3]==0) DO ...
; while input 1 is applied or input 3 is not applied ...

Two or more real-time expressions may be compared with one another within one condition.
Comparisons may be made between variables **of the same type** or between partial expressions.
Example:
WHEN $AA_IM[X2] <= **$AA_IM[X1] +.5** DO $AA_OVR[X1]=0
; Stop when safety clearance is exceeded

The options for applying real-time expressions are described in the “Calculations in real time” chapter. When conditions are programmed, all the system variables listed in Subsection 2.3.11 can be addressed. In addition:

- Machine data, e.g. $$MN_..., $$MC_..., $$MA_...
- Setting data, e.g. $$SN_..., $$SC_..., $$SA_...

Note

- GUD variables cannot be used
- R parameters are addressed with $R....
- Setting and machine data whose value may vary during machining must be programmed with **$$**S._... / **$$**M._...

Further examples of conditions can be found in Section 6.1.

G code for the action

This G code may specify a G code for all actions in the block and technology cycles, which differs from the one set in the condition. If technology cycles are contained in the action part, the G code remains modally active for all actions until the next G code, even after the technology cycle has been completed. Only one G code of the G code group may be programmed for each action part.

Actions

Every synchronized action contains one or more programmed actions or one technology cycle. These are executed when the appropriate condition is fulfilled. If several actions are programmed in one synchronized action, they are executed within the same interpolation cycle.

Example: WHEN $AA_IM[Y] >= 35.7 DO **M135 $A_OUT[1]=1**
If the actual value of the Y axis is greater than or equal to 35.7, then M135 is output to the PLC and output 1 set at the same time.

Program/technology cycle

A program (name) can also be specified as an action. This program may contain any of the actions, which can be programmed individually in synchronized actions. Such programs are also referred to as **technology cycles** below. A technology cycle is a sequence of actions that are processed sequentially in the interpolation cycle. See Section 2.5.

Application: Single axis programs, cyclic machines.

Machining process

The blocks of a part program are prepared at the program preprocessing stage, stored and then executed sequentially on the interpolation level (main run). Variables are accessed during block preparation. When **real-time** variables (e.g. actual values) are used, block preparation is interrupted to allow current real-time values up to the preceding block to be supplied. Synchronized actions are transported to the interpolator in preprocessed form together with the prepared block. The real-time variables used are evaluated in the **interpolation cycle**. Block preparation is not interrupted.

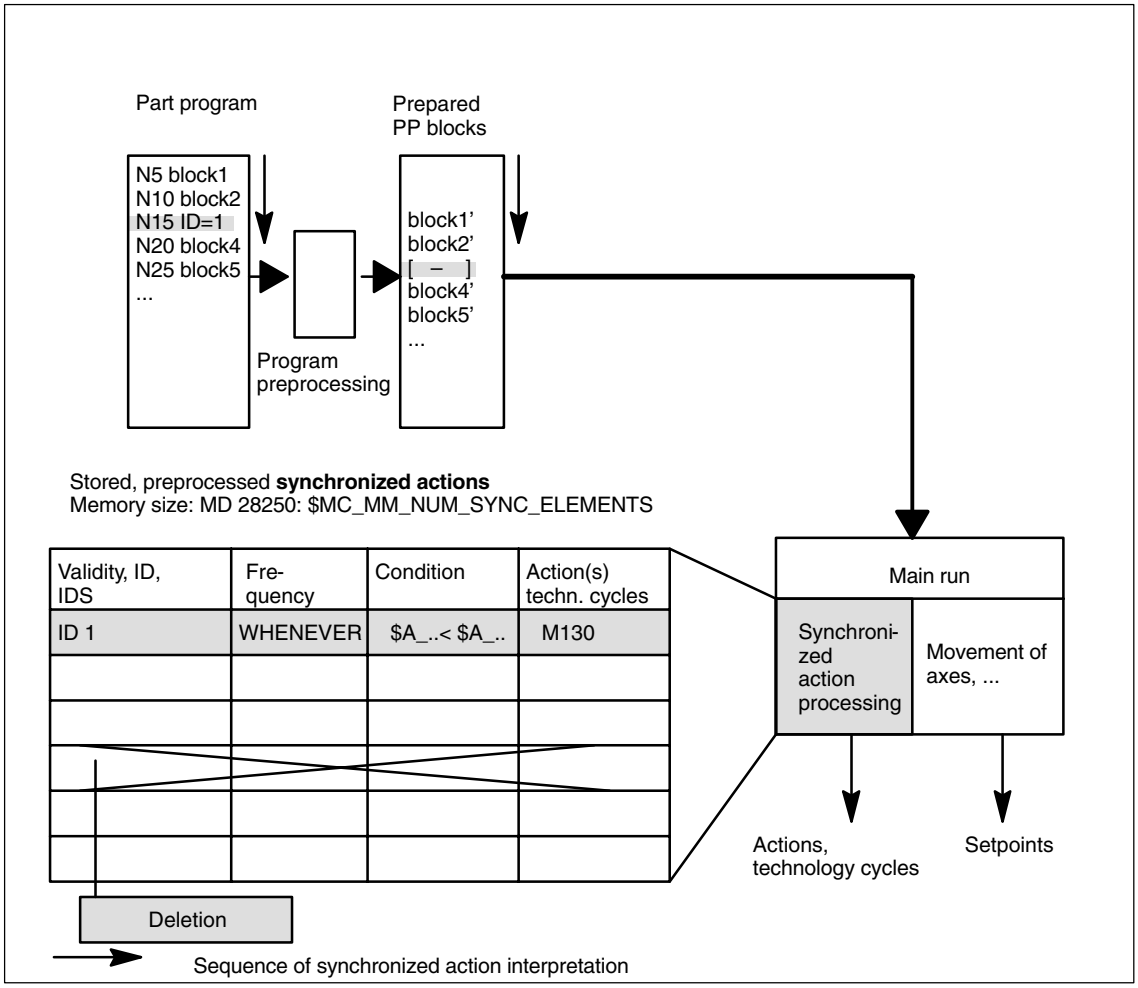


Fig. 2-1 Schematic diagram illustrating processing of synchronized actions

Processing of synchronized actions

Synchronized actions are checked in the interpolation cycle to determine whether they contain actions to be activated. Action(s) are executed in synchronism with path control if the preconditions programmed on the left of the action(s) are fulfilled.

Order of execution

Within an interpolation cycle, modal synchronized action instructions are processed in order of their ID number (i.e. block with ID number 1 before block with ID number 2, etc.). Once the modal synchronized action instructions have been executed, non-modal synchronized action instructions are processed in the order in which they are programmed.

2.1.1 Definition of motion-synchronous actions

Defining programs

Motion-synchronous actions can be defined in the following ways:

- In the part program
- Static synchronized actions in an asynchronous subprogram activated by the PLC

2.1.2 Execution of synchronized actions

Conditions for execution

The actions programmed in motion-synchronous actions are executed if

- The synchronized action exists and has not been deselected with CANCEL(ID) (see Subsection 2.5.1)
- The synchronized action is not disabled, i.e. no LOCK(ID) (see Subsection 2.5.1)
- Evaluation of the action is due as a result of the programmed frequency keyword or
- The appropriate condition is fulfilled

For further details, please see the following subsections.

2.1.3 List of possible actions

- Output of M, S and H auxiliary functions to the PLC
- Real-time variables can be set (written) to obtain the following functionality:
  - Overlaid movement ($AA_OFF), option.
  - Feedrate control ($AC_OVR, $AA_OVR), disabling of a programmed axis motion
  - ...
- Changes to SW cam positions and times (setting data) and alteration of other setting data
- Modification of coefficients and limits from FCTDEF
- SYNFCT (polynomial evaluation)
- FTOC (online tool offsets)
- RDISABLE (read-in disable)
- STOPREOF (preprocessing stop cancellation)
- DELDTG (delete distance-to-go)
- Calculation of curve table values
- Axial feedrate from synchronized actions
- Axial frames
- Moving/positioning axes from synchronized actions
- Spindle motions from synchronized actions
- Actual-value setting from synchronized actions (Preset)
- Activation/deactivation of couplings and coupled motion
- Measurements from synchronized actions
- Setting and deletion of wait markers for channel synchronization
- Set alarm/error reactions
- Travel to fixed stop FXS (FXST, FXSW)
- Travel with limited torque FOC (FOCON/FOCOF)
- Extended stop and retract (Description of Functions M3)
- Reading and, if tagged accordingly, writing of system variables from the list in Subsection 2.3.11.

These actions are described in detail in Section 2.4.

2.2 Real-time evaluations and calculations

Restriction

Calculations carried out in real time represent a subset of those calculations that can be performed in the NC language. It is restricted to data types REAL, INT, CHAR and BOOL.

Implicit type conversions, such as in the part program, do not take place. See data type below.

Scope of application

The term “Real-time expression” refers below to all calculations that can be carried out in the interpolation cycle. Real-time expressions are used in conditions and in assignments to NC addresses and variables.

Real-time variables

All real-time variables are evaluated (read) in the interpolation cycle and can be written as part of an action.

Real-time variable identifiers

Real-time variables are all variables that begin with:
$A... (main run variable) or
$V... (servo values).
So that they can be clearly identified, these variables can be programmed with

$$ in synchronized actions.
E.g. $AA_IM[X] or $$AA_IM[Y]: Actual value for X axis or Y axis in the machine coordinate system.

Note

Setting data and machine data must be programmed with **$$**S... / **$$**M... if its value changes during machining.

Data type

Only real-time variables of the **same data type** may be linked by a logic operation within the same expression. However, in order to process various types of data, you can use the conversion routines provided for type matching (SW 5.2, see conversion routines). In contrast to full expressions in the NC language, calculations are performed in the data type of the real-time variables involved.

```
... DO $R10 = $AC_PARAM[0]      ; permissible REAL, REAL
... DO $R10 = $AC_MARKER[0]     ; not permissible: REAL, INT
```

The following examples of real-time evaluations were already available in SW version 3.2 (they employ only real-time variables of this SW version):

Example 1 for SW 3.2

On the left of the comparison, there is a comparison variable calculated in real time and, on the right, an expression, which is none of the permitted real-time processing variables that begin with $$.

WHEN $AA_IM[X] > **$A_INA**[1] DO M120

M120 is output during execution of the motion programmed in the following block if the X axis actual value exceeds the value applied at analog input 1. With this programming, the actual value is re-evaluated in every interpolation cycle while the value of the analog input is generated at the instant of interpretation.

Example 2 for SW 3.2

On the left of the comparison, there is a comparison variable calculated in real time and, on the right, an expression, which is one of those permitted for the synchronized action (beginning with $$).

WHEN $AA_IM[X] > **$$A_INA**[1] DO M120

The current actual value of the X axis is compared in the IPO cycle with analog input 1 because an $$ variable is programmed on the right.

Both variables are compared to one another in the interpolation cycle.

Example 3 for SW 3.2

$$ variables may also be programmed on the left of the comparison.

WHEN **$$AA_IM[X]** > **$$A_INA[1]** DO M120

Identical to example 2. The left-hand and right-hand sides are always compared in real time.

Extensions in SW 4

The real-time variables available in synchronized actions are listed in Subsection 2.3.11. New system variables, which have been added in subsequent software versions are indicated in the table.

- Machine and setting data

In the case of machine and setting data, $$S... or $$M... must be programmed for online access. The access instruction to be evaluated during interpretation/ decoding must be preceded by a $ sign. Real-time variables that may legally be accessed from synchronized actions are addressed preceded only by a $ sign.

Conversion routines (SW 5.2)

There is no implicit type conversion from REAL to INT and vice versa for synchronized actions. However, the user may explicitly call two conversion routines **RTOI( ) and ITOR( )** for type conversion. The functions can be called

- In the part program and
- From the synchronized action.

ITOR

REAL ITOR(INT) – Converting integer to real

The function converts the integer value transferred to a real value and returns this value. The transferred variable is not changed.

Example:

$AC_MARKER[1] = 561
ID=1 WHEN TRUE DO $AC_PARAM[1] = **ITOR**($AC_MARKER[1])

RTOI

INT RTOI(REAL) – Converting from real to integer

The function RTOI() converts the Real value presented to a rounded INT value and returns this integer value. If the value transferred lies outside the range that can be unambiguously represented as an integer value, alarm 20145 “Motion-synchronous action: Arithmetic error” is output and no conversion is performed. The transferred variable is not changed.

Note

The function RTOI() does not produce an unambiguous result when inverted, i.e. it is not possible to determine the original Real value from the value returned as the decimal places are lost during conversion!

Example RTOI:

```
$AC_PARAM[1] = 561.4378
ID=1 WHEN TRUE DO $AC_MARKER[1] = RTOI( $AC_PARAM[1] )
                                  ; Result: 561

...

$AC_PARAM[1] = –63.867
ID=1 WHEN TRUE DO $AC_MARKER[1] = RTOI( $AC_PARAM[1] )
                                  ; Result:–64

...

$AC_MARKER[1]= 10 $AC_PARAM[1] = –6386798797.29
ID=1 WHEN TRUE DO $AC_MARKER[1] = RTOI( $AC_PARAM[1] )
                  ;Result: Alarm 20145

                  ;$AC_MARKER[1] = 10 (unchanged due to alarm)
```

Implicit type conversion (SW 6.4)

In SW 6.4 and later, variables of various data types can be assigned to one another in synchronized actions without having to call the RTOI or ITOR function, e.g. REAL to INT and vice versa.

If values outside of the interval [INT_MIN, INT_MAX] would result from the conversion from REAL to INTEGER, alarm 20145 “Motion-synchronous action: Arithmetic error” is output and no conversion is performed.

Examples:
Previously

```
$AC_MARKER[1] = 561
ID=1 WHEN TRUE DO $AC_PARAM[1] = ITOR( $AC_MARKER[1] )
```

In software version 6.4 and later

```
$AC_MARKER[1] = 561
ID=1 WHEN TRUE DO $AC_PARAM[1] = $AC_MARKER[1]
```

Previously

```
$AC_PARAM[1] = 561.4378
ID=1 WHEN TRUE DO $AC_MARKER[1] = RTOI( $AC_PARAM[1] )  ; 561
```

In software version 6.4 and later

```
$AC_PARAM[1] = 561.4378
ID=1 WHEN TRUE DO $AC_MARKER[1] = $AC_PARAM[1]          : 561
```

Basic arithmetic operations

Real-time variables of the REAL and INT type can be linked logically by the following basic arithmetic operations:

- Addition
- Subtraction
- Multiplication
- Division
- Integer division
- Modulo division

Only variables of the same type may be linked by these operations.

Expressions

Expressions from basic arithmetic operations can be bracketed and nested.
See priorities for operators on the next page.

Comparisons

The following relational operators may be used:

==	Equal to
<>	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Boolean operators

The following Boolean operators may be used:

NOT	NOT,
AND	AND,
OR	OR,
XOR	Exclusive OR

Bit operators

The following bit operators may be used:

B_OR	Bit-serial OR
B_AND	Bit-serial AND
B_XOR	Bit-serial exclusive OR
B_NOT	Bit-serial negation

Operands are variables and constants of the INT type.

Priority of operators

In order to produce the desired logical result in multiple expressions, the following operator priorities should be observed in calculations and conditions:

1. NOT, B_NOT	Negation, bit-serial negation
2. *, /, DIV, MOD	Multiplication, division
3. +, –	Addition, subtraction
4. B_AND	Bit-serial AND
5. B_XOR	Bit-serial exclusive OR
6. B_OR	Bit-serial OR
7. AND	AND
8. XOR	Exclusive OR
9. OR	OR
10.	Not used
11.	Relational operators
==	Equal to
<>	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

and parentheses should be used where necessary. The logic operation result for a condition must be a BOOL data type.

Example of a multiple expression:
WHEN ($AA_IM[X] > VALUE) AND ($AA_IM[Y] > VALUE1) DO ...

Functions

A real-time variable of the REAL type can be used to create function values sine, cosine, etc.
The following functions are possible:

SIN, **COS**, **ABS**, **ASIN**, **ACOS**, **TAN**, **ATAN2,**
TRUNC, **ROUND, LN, EXP, ATAN, POT, SQRT,**
CTAB, CTABINV

Example:
... DO $AC_PARAM[3]=**COS**($AA_IM[X])

For a description of how to use these functions, please see:

References: /PG/, Programming Guide
/PGA/, Programming Guide Advanced

Indexing

The index of a real-time field variable can in turn be a real-time variable.

Example:
WHEN ... DO $AC_PARAM[**$AC_MARKER[1]**] = 3
The current value of the index $AC_MARKER[1] is evaluated in each interpolation cycle.

Restrictions:

- It is not permissible to nest indices with real-time variables.
- A real-time index cannot be generated by a variable that is not generated itself in real time. The following programming would lead to errors:
  $AC_PARAM[1]=**$P_EP[$AC_MARKER[0]]**

2.3 Special real-time variables for synchronized actions

A **complete list** of system variables that may be addressed in synchronized actions can be found in Subsection 2.3.11. The characteristics of a few special real-time variables are described below:

- Marker/counter variables
  - Channel-specific markers
- Timers
- Synchronized action parameters
- R parameters
- Machine and setting data
- FIFO variables (circulating memory)

SW 4

Special real-time variables, i.e. timers, R parameters, machine and setting data and FIFO variables are available in SW 4 and later.

2.3.1 Marker/counter variables

Channel-specific markers

The **$AC_MARKER[n]** variable serves as a marker or counter in the INTEGER data type.

n: Number of marker: 0-n

The number of markers per channel is set via machine data
MD 28256: NUM_AC_MARKER.

Markers exist once in each channel under the same name. They are stored in the dynamic memory and reset to 0 on POWER ON, NC RESET and End of Program, ensuring identical start conditions for every program run.

Marker variables can be read and written in synchronized actions.

Also available in SW 6.3 and later

In software version 6.3 and later, it is possible to select the memory location for $AC_MARKER[n] between DRAM and SRAM using
MD 28257: MM_BUFFERED_AC_MARKER.

0: Dynamic memory DRAM, (default)
1: Static memory SRAM

A maximum value of 2000 can be assigned to machine data 28256: NUM_AC_MARKER. One element requires 4 bytes. You must ensure that sufficient memory of the correct type is available.

Markers saved in SRAM can be included in the data backup. See Subsection 2.3.7

2.3.2 Timers

System variable **$AC_TIMER[n]** permits actions to be started after defined waiting times.

n: Number of timer variable
Unit: Seconds
Data type: REAL

The number of available timer variables is programmed in machine data
MD 28258: MM_NUM_AC_TIMER.

Setting timers

Incrementation of a timer variable is started by means of value assignment:

$AC_TIMER[n]=value
n: Number of timer variable
Value: Start value (normally 0)

Stopping timers

Incrementation of a timer variable can be stopped by assigning a negative value:

$AC_TIMER[n]=–1

Reading timers

The current timer value can be read whether the timer variable is running or has been stopped. Once a timer variable has been stopped by assigning –1, the current timer value remains stored and can be read.

Example

Output of an actual value via analog output 500 ms after detection of a digital input:

```
WHEN $A_IN[1]==1 DO $AC_TIMER[1]=0 ;Reset and start timer
WHEN $AC_TIMER[1]>=0.5 DO $A_OUTA[3]=$AA_IM[X] $AC_TIMER[1]=–1
```

2.3.3 Synchronized action parameters

$AC_PARAM[n] variables serve as a buffer in synchronized actions.
Data type: REAL
n: Number of parameter 0 – n

The number of available AC parameter variables in each channel is programmed via machine data
MD 28254: .MM_NUM_AC_PARAM.

These parameters exist once in each channel under the same name. $AC_PARAM parameters are stored in the dynamic memory and reset to 0 on POWER ON, NC RESET and End of Program, ensuring identical start conditions for every part program run. $AC_PARAM variables can be read and written in synchronized actions.

Also available in SW 6.3 and later

In software version 6.3 and later, it is possible to select the memory location for $AC_PARAM[n] between DRAM and SRAM using
MD 28255: MM_BUFFERED_AC_PARAM.

0: Dynamic memory DRAM, (default)
1: Static memory SRAM

A maximum value of 2000 can be assigned to machine data 28255: NUM_AC_PARAM. One element requires 8 bytes. You must ensure that sufficient memory of the correct type is available.

Synchronization parameters saved in SRAM can be included in the data backup. See Subsection 2.3.7

2.3.4 R parameters

Definition

R parameters are variables of the REAL time that are stored in battery-backed memory.

For this reason, they retain their settings after end of program, RESET and POWER ON.

Application in synchronized actions

By programming the **$** sign in front of R parameters, they can also be used in synchronized actions.

Example:
WHEN $AC_MEA== 1 DO **$R10**= $AA_MM[Y]
; if valid measurement available, transfer measured value to R parameter

---

Note

It is advisable to apply a given R variable either normally in the part program or in synchronized actions. If an R variable that has been used in synchronized actions must be later applied “normally” in the part program, then a STOPRE instruction must be programmed to ensure synchronization. Example:
WHEN $A_IN[1] == 1 DO $R10 = $AA_IM[Y]
G1 X100 F150
STOPRE
IF R10 > 50 .... ; evaluation of R parameter

2.3.5 Machine and setting data

In SW version 4 and later it is possible to read and write machine and setting data from synchronized actions. Access must be programmed according to the following criteria:

- MD, SD that remain unchanged during machining and
- MD, SD, whose settings change during machining

Reading invariable MD, SD

Machine and setting data whose settings do not vary are addressed from synchronized actions in the same way as in normal part program commands. They are preceded by a **$** sign.
Example:
ID=2 WHENEVER $AA_IM[z]< **$**SA_OSCILL_REVERSE_POS2[Z]–6 DO $AA_OVR[X]=0
; Here, reversal range 2, which is assumed to remain static during operation, is
; addressed for oscillation

For a complete example of oscillation with infeed within the reversal range, please see Section 6.2 and:

References: /FB/, P5, Oscillation

Reading variable MD, SD

Machine data and setting data whose values may change during machining are addressed from a synchronized action preceded by the **$$** sign.

Example:
ID=1 WHENEVER $AA_IM[z]< **$$**SA_OSCILL_REVERSE_POS2[Z]–6 DO $AA_OVR[X]=0
In this situation, it is assumed that the reversal position can be changed at any time by an operator action.

Writing MD, SD

Requirement:
The currently set access authorization level must allow write access. It is not meaningful to change MD and SD from synchronized actions unless the change takes immediate effect. The effectiveness of changes is stated individually for all MD and setting data in:

References: /LIS/, Lists

Addressing:
Machine and setting data to be changed must be addressed preceded by the **$$** sign.

Example:

```
ID=1 WHEN $AA_IW[X]>10 DO $$SN_SW_CAM_PLUS_POS_TAB_1[0]= 20
                          $$SN_SW_CAM_MINUS_POS_TAB_1[0]= 30
; Alteration of switching positions of SW cams
```

2.3.6 FIFO variables (circulating memory)

Application

Up to 10 FIFO variables are provided to allow storage of related data sequences: **$AC_FIFO1[n]** to **$AC_FIFO10[n]**.

Structure

Fig. 2-3 shows the memory structure of a FIFO variable.

Number

The number of available AC FIFO variables is programmed in machine data
MD 28260: NUM_AC_FIFO.

Size

The number of values that can be stored in a FIFO variable is defined via machine data
MD 28264: LEN_AC_FIFO.
All FIFO variables are equal in length.

Data type

Values in FIFO variables are of the REAL data type.

Meaning of index

Index n:

Indices 0 to 5 have special meanings:

n= 0:	When the variable is written with index 0, a new value is stored in the FIFO When it is written with index 0, the oldest element is read and deleted from the FIFO
n=1:	Access to oldest stored element
n=2:	Access to most recent stored element
n=3:	Sum of all FIFO elements MD 28266: MODE_AC_FIFO determines the mode of summation: Bit 0 = 1 Update sum on every new write Bit 0 = 0 Summation not possible.
n=4:	Number of elements available in FIFO. Every element in the FIFO can be read and write-accessed. FIFO variables are reset by resetting the number of elements, e.g. for the first FIFO variable: FIFO variable parameter: **$AC_FIFO1[4]=0**
n=5	Current write index relative to beginning of FIFO
n= 6 to 6+nmax:	Access to nth FIFO element:

Note

FIFO access is a special form of R parameter access (see below). FIFO values are stored in the R parameter area, i.e. FIFO values are stored in the static storage area. They are not deleted by end of program, RESET or POWER ON. FIFO values are stored simultaneously when R parameters are archived.

Machine data

MD 28262: START_AC_FIFO

defines the number of the R parameter, which marks the beginning of FIFO variable storage in the R parameter area.

The current number of R parameters in a channel is programmed in machine data

MD 28050: MM_NUM_R_PARAM.

The following two diagrams show a schematic representation of part lengths of parts on a belt that have been stored in FIFO variables.

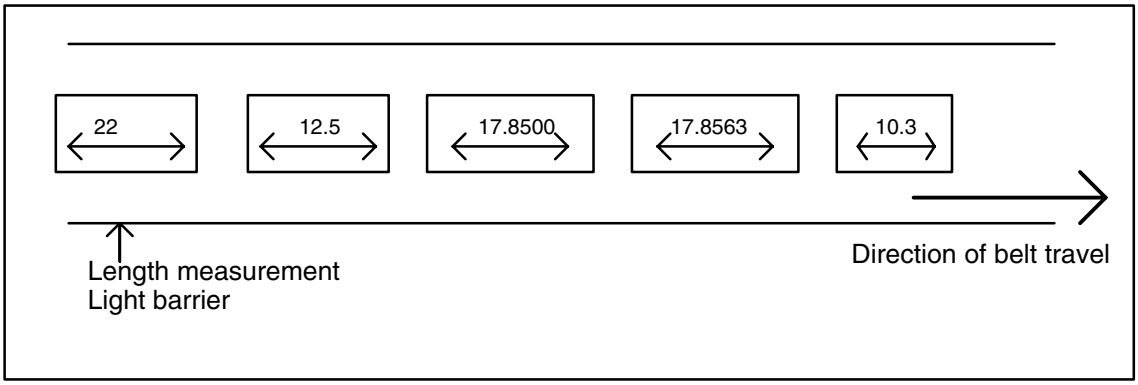


Fig. 2-2 Product lengths of sequence of parts on conveyor belt

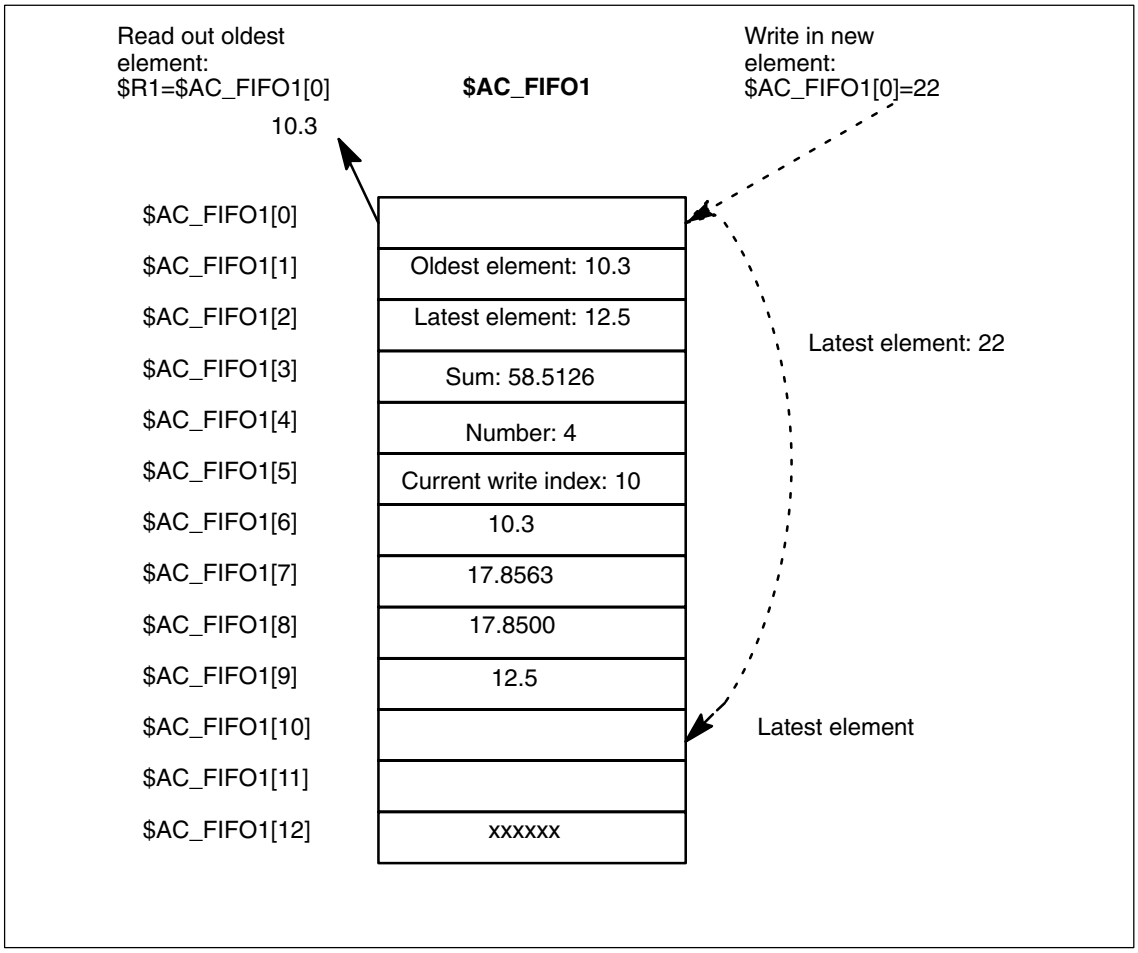


Fig. 2-3 Example of FIFO variables

2.3.7 System variables saved in SRAM (SW 6.3 and later)

RESET response

System variables $AC_MARKER and $AC_PARAM saved in SRAM retain their values after RESET and POWER ON.

Note

In the case of part programs and synchronized actions that used system variables saved in SRAM, you must make sure that the variables are not initialized to 0 after RESET. This may require some adaptation if system variables saved in DRAM have been used previously.

Data backup

System variables $AC_MARKER and $AC_PARAM saved in SRAM can be included in the data backup. The following backup modules are present for each channel:

_N_CHi_ACM	for $AC_MARKER values and
_N_CHi_ACP	for $AC_PARAM values.

i denotes the relevant channel number.

Sequence

The saved modules are entered in the full backup file _N_INITIAL_INI according to R parameters.

References: /IAD/, Installation and Start-Up Guide

2.3.8 Determining the path tangent in synchronized actions

$AC_TANEB

The system variable $AC_TANEB
(**T**angent **AN**gle at **E**nd of **B**lock), which can be read in synchronized actions, calculates the angle between the path tangent at the end of the current block and the path tangent at the start of the programmed following block.

The tangent angle is always output as a positive value between 0.0 and 180.0 degrees. If there is no following block in the main run, the angle –180.0 degrees is output.

The system variable $AC_TANEB should not be read for blocks generated by the system (intermediate blocks). The system variable $AC_BLOCKTYPE is used to determine whether the block is a programmed block (main block).

Programming example:

```
ID=2 EVERY $AC_BLOCKTYPE==0 DO $R1 = $AC_TANEB;
```

2.3.9 Determining the current override

Current override

The current override (NC component) can be read and written in synchronized actions with the system variables:
$AA_OVR Axial override
$AC_OVR Path override.

PLC override

The override preset by the PLC is made available for reading for synchronized actions in the system variables:
$AA_PLC_OVR Axial override
$AC_PLC_OVR Path override.

Resulting override

The resulting override is made available for reading for synchronized actions in the system variables:
$AA_TOTAL_OVR Axial override
$AC_TOTAL_OVR Path override.

The resulting override is calculated as follows:
$AA_OVR * $AA_PLC_OVR or
$AC_OVR * $AC_PLC_OVR

2.3.10 Capacity evaluation using time requirement for synchronized actions

In an interpolation cycle, synchronized actions must be interpreted and movements, etc. calculated by the NC. The system variables described below can be used by synchronized actions to find out information about the current time components of the synchronized actions in the interpolation cycle and about the computing time of the position controller.

The variables only have valid values if machine data $MN_IPO_MAX_LOAD is greater than 0. If it is not, the variables will only ever indicate the gross computing time. This is calculated as follows:
– Synchronized action time
– Position control time and the
– Remaining IPO computing time

The variables always contain the values of the **previous** IPO cycle.

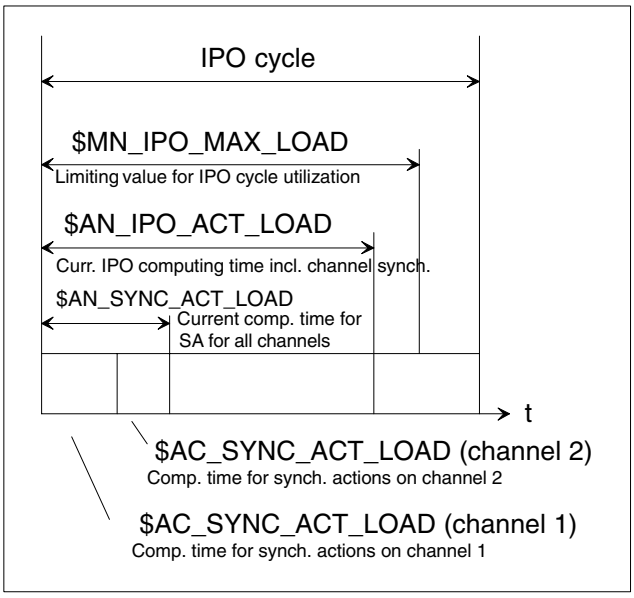


Fig. 2-4 Time components of synch. actions in IPO cycle

System variables	Meaning
$AN_IPO_ACT_LOAD	Current IPO computing time (incl. synchronized actions for all channels)
$AN_IPO_MAX_LOAD	Longest IPO computing time (incl. synchronized actions for all channels)
$AN_IPO_MIN_LOAD	Shortest IPO computing time (incl. synchronized actions for all channels)
$AN_IPO_LOAD_PERCENT	Current IPO computing time as a percentage of the IPO cycle
$AN_SYNC_ACT_LOAD	Current computing time for synch. actions for all channels

System variables	Meaning
$AN_SYNC_MAX_LOAD	Longest computing time for synch. actions for all channels
$AN_SYNC_TO_IPO	Percentage share of all synch. actions in the overall IPO computing time (for all channels)
$AC_SYNC_ACT_LOAD	Current computing time for synchronized actions in channel
$AC_SYNC_MAX_LOAD	Longest computing time for synchronized actions in channel
$AC_SYNC_AVERAGE_LOAD	Average computing time for synchronized actions in channel
$AN_SERVO_ACT_LOAD	Current computing time of position controller
$AN_SERVO_MAX_LOAD	Longest computing time of position controller
$AN_SERVO_MIN_LOAD	Shortest computing time of position controller

Overload information

MD 11510: IPO_MAX_LOAD can be used to specify the IPO gross computing time (as a % of the IPO cycle) at and above which the system variable $AN_IPO_LOAD_LIMIT should be set to TRUE.

If the current load falls back below this limit, the variable is reset to FALSE.

If the MD = 0, the entire diagnostic function is deactivated.
The user can evaluate $AN_IPO_LOAD_LIMIT to define a specific strategy for avoiding plane overflow.

Writeable system variables

The system variables described above can be **written to** from synchronized actions:

System variables	Meaning
$AN_SERVO_MAX_LOAD	Longest computing time of position controller
$AN_SERVO_MIN_LOAD	Shortest computing time of position controller
$AN_IPO_MAX_LOAD	Longest IPO computing time (incl. synchronized actions for all channels)
$AN_IPO_MIN_LOAD	Shortest IPO computing time (incl. synchronized actions for all channels)
$AN_SYNC_MAX_LOAD	Longest computing time for synch. actions for all channels
$AC_SYNC_MAX_LOAD	Longest computing time for synchronized actions in channel

On every write access, these variables are reset to the current load, regardless of the value written.

Programming example

```
$MN_IPO_MAX_LOAD = 80 ; MD: Utilization limit for IPO cycle
                      ; As soon as $AN_IPO_LOAD_PERCENT > 80%,
                      ; $AN_IPO_LOAD_LIMIT is set to TRUE.

N01 $AN_SERVO_MAX_LOAD=0
N02 $AN_SERVO_MIN_LOAD=0
N03 $AN_IPO_MAX_LOAD=0
N04 $AN_IPO_MIN_LOAD=0
N05 $AN_SYNC_MAX_LOAD=0
N06 $AC_SYNC_MAX_LOAD=0

N10 IDS=1 WHENEVER $AN_IPO_LOAD_LIMIT == TRUE DO M4711
                                                  SETAL(63111)
N20 IDS=2 WHENEVER $AN_SYNC_TO_IPO > 30 DO SETAL(63222)
N30 G0 X0 Y0 Z0
...
N999 M30
```

The first synchronized action generates an auxiliary function output and an alarm if the overall utilization limit is exceeded.

The second synchronized action generates an alarm if the synchronized actions account for more than 30 % of the IPO computing time (all channels).

2.3.11 List of system variables relevant to synchronized actions

Note

In software version 7.1 and later, the system variables listed previously at this point, which can be accessed by synchronized actions, appear in a separate brochure:

/PGA1/ Lists Manual, System Variables

System variables with the corresponding ID can be used by synchronized actions.

2.4 Actions in synchronized actions

Actions

After action code word **DO ...**, each synchronized action contains

– One or more (max. 16) actions or a technology cycle (these two components are referred to generally as **actions** in the following description).

These are executed when the appropriate condition is fulfilled.

Several actions

Several actions contained in a synchronized action are activated in the same interpolation cycle if the appropriate condition is fulfilled.

List of possible actions

The following actions can be programmed in the “Action” section of synchronized actions:

Table 2-2 Actions in synchronized actions

... DO ...	**Meaning**	**Reference**
Mxx Sxx Hxx	Output of auxiliary functions to PLC	2.4.1
SETAL(no.)	Set alarm, error reactions	2.4.20
$A...= ... $V... = ...	Write real-time variables:	2.4.2
$AA_OFF =	– Overlaid movement	
	– Velocity control:	
$AC_OVR =	Tool path velocity	
$AA_OVR =	Axis velocity	
$AC_VC =	Add. path feedrate override	
$AA_VC =	Add. axis compensation value	
$$SN_SW_CAM_ ...	Change SW cam positions (setting data) and all other SD	2.4.3
$AC_FCT...	Overwrite FCTDEF parameters	2.4.4
	Synchronized action procedures:	
RDISABLE	Activate read-in disable	2.4.8
STOPREOF	End preprocessing stop	2.4.9
DELDTG	Delete distance-to-go	2.4.10
FTOC	Online tool offset	2.4.7
SYNFCT	Polynomial evaluation	2.4.5
ZYKL_T1 (e.g.)	Call of technology cycles	2.5
	Control positioning axes:	
$AA_OVR[x]= 0	Disable an axis motion	2.4.11
AXIS_X (e.g.)	Call an axis program	2.4.12
POS[u]= ...	Position	2.4.12
FA[u]= ...	Define axis feedrate	2.4.13
	Move command axes continuously:	2.4.14
MOV[u]= >0	– Forwards	”
MOV[u] = <0	– Backwards	”
MOV[u] = =0	– Stop	”
	Spindles:	
SPOS	Position	2.4.15
M3, M4, M5, S =	Direction of rotation, stop, speed	
$AA_OVR[S1]= 0	Disable spindle motion	
PRESETON(,)	Preset actual value memory	2.4.16

Table 2-2 Actions in synchronized actions

... DO ...	Meaning	Reference
	Activate/deactivate couplings:	2.4.17
LEADON	Couple slave axis to master axis	
LEADOF	Cancel coupling	
TRAILON	Asynchr. coupled motion ON	
TRAILOF	Asynchr. coupled motion OFF	
MEAWA,	Measurement without deletion of distance-to-go	2.4.18
MEAC	Cyclic measurement	
	Channel synchronization:	2.4.19
SETM	Set a wait marker	
CLEARM	Clear a wait marker	
	Coordination of synchronized actions:	2.5.1
LOCK	– Disable synchronized action/technolog cycle	
UNLOCK	– Enable synchronized action/technology cycle	
RESET	– Reset technology cycle	

2.4.1 Output of M, S and H auxiliary functions to PLC

For general information about auxiliary function outputs, please see:

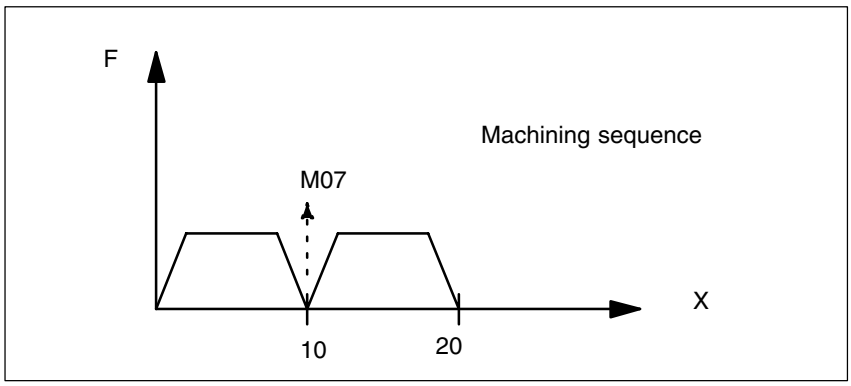
References: /FB/, H2, “Output of Auxiliary Functions to PLC”

Examples

The advantage of implementing auxiliary function outputs in synchronized actions is illustrated by the following example: Switch on coolant at a specific position

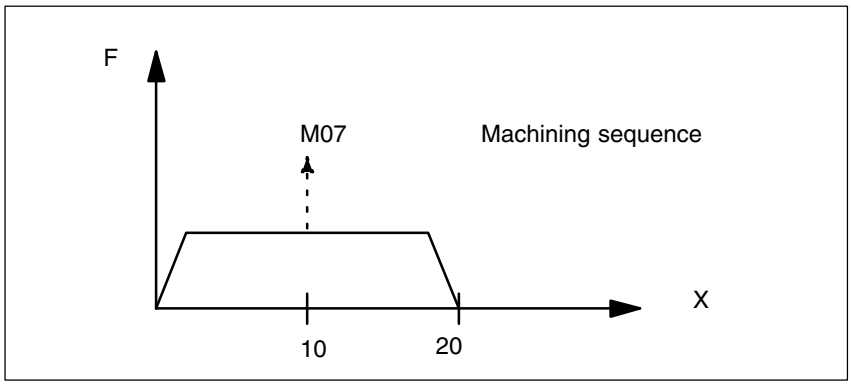
Solution **without** synchronized action: 3 blocks

```
N10 G1 X10 F150
N20 M07
N30 X20
```



Solution **with** synchronized action: 1 block

```
N10 WHEN $AA_IM[X] >= 10 DO M07
N20 G1 X20 F150
```



Auxiliary function output to the PLC

M, S or H auxiliary functions can be output to the PLC as synchronized actions. The output takes place **immediately** (like an interrupt on the PLC) in the interpolation cycle if the condition is fulfilled.
The timing that might be programmed in
MD 11110: AUXFU_GROUP_SPEC (auxiliary function group specification) or
AUXFU_M_SYNC_TYPE (output timing of M functions)/
AUXFU_S_SYNC_TYPE (output timing of S functions)/
AUXFU_H_SYNC_TYPE (output timing of H functions)/
has no effect.

Programming

Auxiliary functions may be programmed with frequency keywords WHEN or EVERY only in synchronized actions.

Example

```
WHEN $AA_IM[X] > 50 DO H15 S3000 M03
; if actual value of X axis is greater than 50, then output H15, set new spindle
speed, new direction of rotation
```

Restrictions

No more than 10 auxiliary functions may be output simultaneously (i.e. in an OB 40 cycle of the PLC). The total number of auxiliary function outputs from part programs and synchronized actions must never exceed 10 at any point in time.
Maximum number of auxiliary functions per synchronized action block or technology cycle block:

- 5 M functions
- 3 S functions
- 3 H functions

Predefined M functions cannot be programmed by means of synchronized actions. They will be rejected by an alarm.

```
WHEN ... DO M0                    ; Alarm
```

However, spindle M functions M3, M4, M5 and M17 may be programmed as the end of a technology cycle.

Acknowledgment

Technology cycle blocks (see Section 2.5) containing auxiliary function outputs are not completely processed until all auxiliary functions in the block have been acknowledged by the PLC. The next block in the technology cycle is not processed until all auxiliary functions in the preceding block have been acknowledged by the PLC.

SW 5

Further methods of acknowledgment have been introduced for SW 5 and later:

- Auxiliary function output without block change delay High-speed auxiliary functions (QUICK) first, as a parallel process in the PLC, then auxiliary function output with anticipated acknowledgment.

The user can choose between INT and REAL as the data type for H auxiliary functions. The PLC user program must interpret the values in accordance with the definition. The INT value range for H auxiliary functions has been increased to: –2 147 483 648 to 2 147 483 647.

References: /FB/, H2, Output of Auxiliary Functions to PLC for SW 5

2.4.2 Setting (writing) and reading of real-time variables

Write

The real-time variables marked with a + sign for access “Write from synchronized actions” in the list in Subsection 2.3.11 can be **written** in actions contained in synchronized actions.

- Machine and setting data, e.g. $$MN_..., $$MC_..., $$MA_... or $$SN_..., $$SC_..., $$SA_...

Note

Machine and setting data that must be written online in the main run must be programmed with **$$**.._...

Activation

Machine data written from synchronized actions must be coded for IMMEDIATE effectiveness. The modified value will not otherwise be available for the remainder of the processing run. Details about the effectiveness of new machine data values after modification can be found in:

References: /LIS/, Lists

Examples:
... DO **$$MN_MD_FILE_STYLE = 3** ; Set machine data
... DO **$$SA_OSCILL_REVERSE_POS1 = 10** ; Set setting data
... DO **$A_OUT[1]=1** ; Set digital output
... DO **$A_OUTA[1]= 25** ; Output analog value

Read

The variables in synchronized actions can be **read-accessed** for assignments to real-time variables, as input quantities for functions and for the purpose of formulating conditions. These variables are indicated by the letter **r** for access “Read from synchronized actions” in the list in Subsection 2.3.11.

- Machine data, setting data, e.g. $$SN_..., $$SC_..., $$SA_...

Note

Machine and setting data whose variables could change during processing must be programmed with **$$**.._... if they need to be addressed online in the main run. In the case of variables whose content remains unchanged, it is sufficient to type a $ sign in front of the identifier.

Examples:
WHEN **$AC_DTEB** < 5 DO ... ; Read distance from end of block in condition
DO $R5= **$A_INA[2]** ; Read value of analog input 2 and assign computing variable

2.4.3 Alteration of SW cam positions and times (setting data)

Introduction

The “Software cams” function allows position-dependent cam signals to be output to the PLC or NCK I/Os.

References: /FB/, N3, Software Cams, Position Switching Signals

Function

Synchronized actions can be programmed to alter cam positions at which signal outputs are set. Existing setting data are written to change these positions. The following setting data can be modified via synchronized actions:

```
$$SN_SW_CAM_MINUS_POS_TAB_1[0..7]  ; Positions of minus cams
$$SN_SW_CAM_MINUS_POS_TAB_2[0..7]  ; Positions of minus cams
$$SN_SW_CAM_PLUS_POS_TAB_1[0..7]   ; Positions of plus cams
$$SN_SW_CAM_PLUS_POS_TAB_2[0..7]   ; Positions of plus cams
```

Example 1

Alteration of a cam position:

```
ID=1 WHEN $AA_IW[x] > 0
                    DO $$SN_SW_CAM_MINUS_POS_TAB_1[0] = 50.0
```

Lead or delay times can be changed via the following setting data:

```
$$SN_SW_CAM_MINUS_TIME_TAB_1[0..7] ; Lead or delay time
                                                  at minus cams
$$SN_SW_CAM_MINUS_TIME_TAB_2[0..7] ; Lead or delay time
                                                  at minus cams
$$SN_SW_CAM_PLUS_TIME_TAB_1[0..7] ; Lead or delay time
                                                  at plus cams
$$SN_SW_CAM_PLUS_TIME_TAB_2[0..7] ; Lead or delay time
                                                  at plus cams
```

Example 2

Alteration of a lead/delay time:

```
ID=1 WHEN $AA_IW[x] > 0
                    DO $$SN_SW_CAM_MINUS_TIME_TAB_1[0] = 1.0
```

Note

Software cams must not be set as a function of velocity via synchronized actions immediately in front of a cam. At least 2–3 interpolation cycles must be available between the setting and the relevant cam position.

2.4.4 FCTDEF

Application

The Online tool offset FTOC and Polynomial evaluation SYNFCT actions described in the following subsections require an interrelationship between an input quantity and an output quantity to be defined in the form of a polynomial. FCTDEF defines polynomials of this type.
For special examples of polynomial application for online dressing of a grinding wheel, please see Subsection 2.4.7. For examples of load-dependent feedrates and clearance control via polynomials, please see Subsection 2.4.5.

Characteristics of polynomials

Polynomials defined by means of FCTDEF have the following characteristics:

- They are generated through an FCTDEF call in the part program.
- The parameters of defined polynomials are real-time variables.
- Individual polynomial parameters can be overwritten using the same method used to write real-time variables. Permissible generally in part program and in action section of synchronized actions. See Subsection 2.4.2.

Note

In SW 4 and later, it is possible to alter validity limits and coefficients of existing polynomials from synchronized actions. Example: WHEN ... DO $AC_FCT1[1]= 0.5

Number of polynomials

In SW 4 and later, the number of polynomials that can be defined simultaneously can be specified in
MD 28252 : MM_NUM_FCTDEF_ELEMENTS.

Block-synchronous polynomial definition

FCTDEF(
Polynomial no.
Lower limit
Upper limit
a0,
a1,
a2,
a3)

The relationship between output quantity y and input quantity x is as follows:
$y= a_0+ a_1x+ a_2x^2 + a_3x^3$

The parameters indicated in the function are stored in system variables as follows:

$AC_FCTLL[n]:	Lower limit, n: Polynomial number
$AC_FCTUL[n]:	Upper limit, n: Polynomial number
$AC_FCT0[n]:	a0 coefficient, n: Polynomial number
$AC_FCT1[n]:	a1 coefficient, n: Polynomial number
$AC_FCT2[n]:	a2 coefficient, n: Polynomial number
$AC_FCT3[n]:	a3 coefficient, n: Polynomial number

On the basis of this relationship, it is also possible to write or modify polynomials directly via the relevant system variables. The validity range of a polynomial is defined via limits $AC_FCTLL[n] and $AC_FCTUL[n].

Call of polynomial evaluation

Stored polynomials can be used in conjunction with the following functions:

- Online tool offset, FTOC()
- Polynomial evaluation, SYNFCT().

References: /PG/, Programming Guide Fundamentals
/PGA/, Programming Guide Advanced
/FB/, W4 “Grinding”

2.4.5 Polynomial evaluation SYNFCT

Application

By applying an evaluation function in the action section of a synchronized action, it is possible to read a variable, evaluate it with a polynomial and write the result to another variable in synchronism with the machining process. This functionality can be used, for example, to perform the following tasks:

- Feedrate as a function of drive load
- Position as a function of a sensor signal
- Laser power as a function of path velocity
  ...

SYNFCT() evaluation function

The function has the following parameters:

SYNFCT(Polynomial number
Real-time variable output
Real-time variable input)

For definition of a polynomial, please see Subsection 2.4.4.

Operating principle of SYNFCT

The polynomial identified by “Polynomial number” is evaluated with the value of the “Real-time variable input”. The result is then limited by maximum and minimum limits and assigned to the “Real-time variable output”.

Example:
FCTDEF(1,0,100,0,0.8,0,0) ; Polynomial 1 is already defined
...
Synchronized action:

ID=1 DO **SYNFCT**(1,$AA_VC[U1], $A_INA[2])
; The additive compensation value of axis U1 is calculated from analog input value 2 on the basis of polynomial 1 in every interpolation cycle

For the ’Real-time variable output’, it is possible to select variables that:

- as an additive control factor (e.g. feedrate)
- as a multiplicative control factor (e.g. override)
- as a position offset or
- directly

into the machining process.

Additive feedrate control

In the case of additive control, the programmed value (F word with respect to Adaptive Control) is compensated by an **additive** factor.
$F_{active} = F_{programmed} + F_{AC}$
The following are examples of “Real-time variable output” settings:
$AC_VC Additive path feedrate override,
$AA_VC[axis] Additive axial feedrate override

Example of additive control of path feedrate

The programmed feedrate (axial or path-related) must be subject to **additive** control by the (positive) X axis current (e.g. infeed torque). The operating point is set to 5 A. The feedrate may be altered by ± 100 mm/min. The magnitude of the axial current deviation may be ± 1 A.

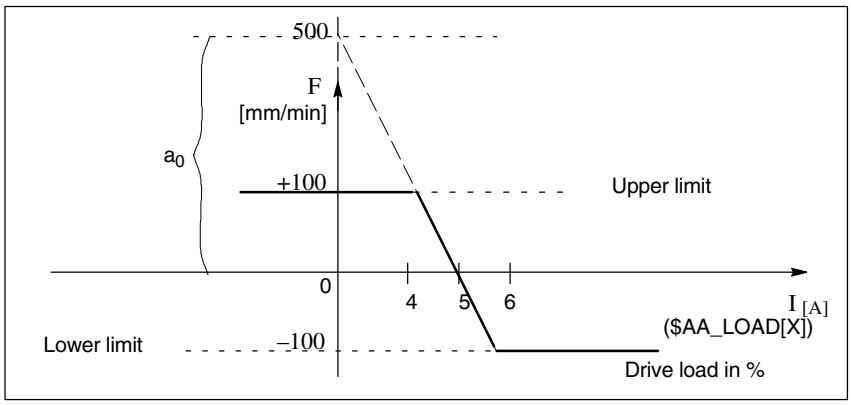


Fig. 2-5 Example of additive control

For definition of coefficients, see also Subsection 2.4.4:

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

$a_1 = -\frac{100\ \text{mm}}{1\ \text{min} \cdot \text{A}}$

$a_1 = -100 \Rightarrow$ control constant

$a_0 = -(-100) \cdot 5 = 500$

$a_2 = 0$ (not a square component)

$a_3 = 0$ (not a cubic component)

Upper limit = 100

Lower limit = –100

The polynomial to be defined (no. 1) is thus as follows:

FCTDEF(1, –100, 100, 500, –100, 0, 0)

The example given in Fig. 2-5 is fully defined with this function.

The *adaptive control* function is activated with the following synchronized action:
ID = 1 DO **SYNFCT**(1, $AC_VC[x], $AA_LOAD[x])
; The additive compensation value for the feedrate of axis x is calculated from the percentage drive load value via polynomial 1 in each interpolation cycle

Multiplicative control

In the case of multiplicative control, the F word is **multiplied** by a factor (override in the case of adaptive control). $F_{active} = F_{programmed} \cdot Factor_{AC}$

Variable $AC_OVR that acts as a *multiplicative* factor on the machining process is used as the real-time variable output.

Example of multiplicative control

The programmed feedrate (axial or path-related) must be subject to **multiplicative** control as a function of drive load. The operating point is set to 100 % at 30 % drive load. The axis(axes) must stop at 80 % drive load. An excessive velocity corresponding to the programmed value +20 % is permissible.

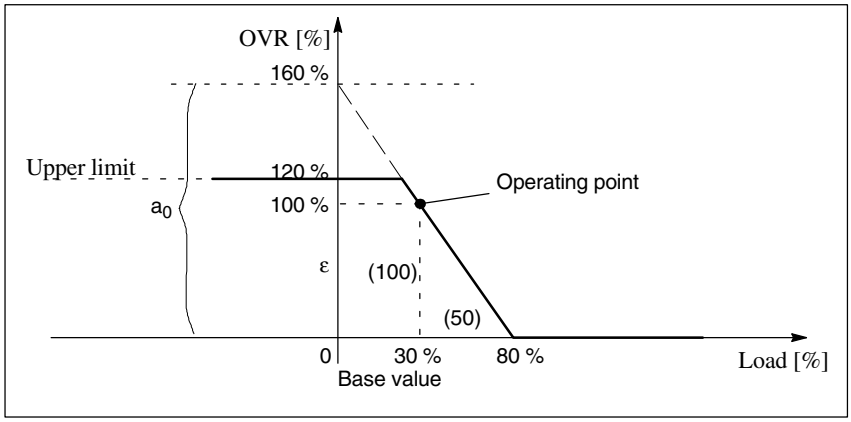


Fig. 2-6 Example of multiplicative control

For definition of coefficients, see also Subsection 2.4.4:

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

$a_1 = -\frac{100\ \%}{(80 - 30)\ \%} = -2$

$a_0 = 100 + (2 \cdot 30) = 160$

$a_2 = 0$ (not a square component)

$a_3 = 0$ (not a cubic component)

Upper limit = 120

Lower limit = 0

The polynomial (no. 2) can therefore be defined as follows:

FCTDEF(2, 0, 120, 160, –2, 0, 0)

The example given in Fig. 2-6 is fully defined with this function.

The associated synchronized action can be programmed as follows:
ID = 1 DO **SYNFCT**(2, $AC_OVR, $AA_LOAD[x])
; The path override is calculated from the percentage drive load for the x axis via polynomial 2 in every interpolation cycle

Position offset with limitation

System variable $AA_OFF controls an axis-specific override that takes immediate effect (basic coordinate system). The mode of override is defined in
MD 36750: $MA_AA_OFF_MODE.
0: Proportional evaluation
1: Integral evaluation
In SW 4 and later, it is possible to limit the value to be compensated absolutely (real-time variable output) to the value stored in setting data
SD 43350 : $SA_AA_OFF_LIMIT.
$AA_OFF_LIMIT[axis] axis-specific system variables can be evaluated in another synchronized action to establish whether the limitation has been reached.
Value –1: Limit of compensation value has been reached in a negative direction.
Value 1: Limit of compensation value has been reached in a positive direction.
Value 0: The compensation value is not within the limit range.

Application:
The SYNFCT function can be used in conjunction with system variable $AA_OFF to implement clearance control in laser machining operations. See below.

Example

Task:
Clearance control as a function of a sensor signal in laser machining operation
The compensation value is limited in the negative Z direction to ensure that the laser head is retracted reliably from finished metal blanks. User reactions such as “Stop axis” (by means of 0 override, see Subsection 2.4.11) or “Set alarm”, see Subsection 2.4.20 can be activated when the limit value is reached.
Boundary conditions:
Integral evaluation of the input quantity of sensor $A_INA[3]. The compensation value is applied in the basic coordinate system, i.e. prior to kinematic transformation. A programmed frame (TOFRAME) has no effect, i.e. the function cannot be used for 3D clearance control in the direction of orientation. The “clearance control” function can be used to implement a clearance control system with high dynamic response or a 3D clearance control system. See

References: /FB/, TE1, “Clearance Control”

References: /PG/, “Programming Guide: Fundamentals”

The interdependency between input quantity and output quantity is assured through the relationship illustrated in the following diagram.

Further examples

Please see Subsection 6.3.1 for an example illustrating dynamic adaptation of a polynomial limit in conjunction with Adaptive Control (clearance control). Please see Subsection 6.3.2 for an example of Adaptive Control applied to path feedrate.

Clearance control

The clearance value is applied integrally via MD 36750: AA_OFF_MODE[V]=1. It works in the basic coordinate system, i.e. before transformation. This means that it can be used for clearance control in the orientation direction (after frame selection with TOFRAME).

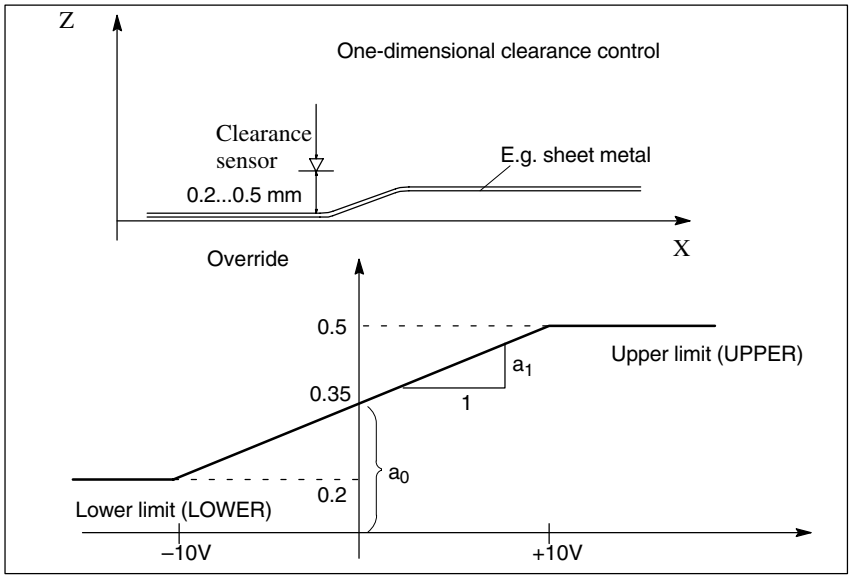


Fig. 2-7 Clearance control

```
%_N_AON_SPF
PROC AON                          ; Subprogram for clearance control ON
FCTDEF(1, 0.2, 0.5, 0.35, 1.5 EX–5) ; Polynomial definition: Compensation is
                                  ;applied in the range 0.2 to 0.5

ID=1 DO SYNFCT(1,$AA_OFF[Z], $A_INA[3])      ; Clearance control active
ID = 2 WHENEVER $AA_OFF_LIMIT[Z]<>0 DO $AA_OVR[X] = 0
                                  ; Disable if limit range X is exceeded.
RET
ENDPROC

%_N_AOFF_SPF
PROC AOFF                         ; Subprogram for clearance control OFF
CANCEL(1)                         ; Delete synchronized action for clearance
                                  ; control
CANCEL(2)                         ; Delete limit range check
RET
ENDPROC

%_N_MAIN_MPF; Main program
                                  ; MD 36750 has been set to 1 for integral
                                  ; evaluation before POWER ON
$SA_AA_OFF_LIMIT[Z]= 1            ; Limit value for compensation
AON                               ; Clearance control ON
...
G1 X100 F1000
AOFF                              ; Clearance control OFF
M30
```

2.4.6 Overlaid movements $AA_OFF settable (SW 6 and later)

Overlaid movements up to SW 5.3

Whatever the current tool and processing level, an overlaid movement is possible for each axis of the channel via the system variable $AA_OFF. The offset is retracted immediately, whether the axis is programmed or not. This allows a clearance control to be implemented.

With axial MD 36750: AA_OFF_MODE, the type of application is defined as follows:

Bit0 = 0: Proportional application (absolute value)
Bit0 = 1: Integral application (incremental value)

$AC_VACTB and $AC_VACTW as input variable for synchronized actions and output are disabled via the options bit (“Feedrate-dependent analog value control” ⇒ laser power control)!

$AA_OFF, position offset as output variable for synchronized actions for clearance control is disabled via the options bit!

Speed limitation with MD 32070: CORR_VELO.

Response of $AA_OFF in SW 6 and later

After RESET, the position offset can still be retained

Previously, during a RESET, the position offset of $AA_OFF was deselected. As, in the case of static synchronized actions IDS = <number> DO $AA_OFF = <value> this response leads to an immediate renewed overlaid movement with the interpolation of a position offset, machine data MD 36750: AA_OFF_MODE can be used to set the RESET response.

Bit1 = 0: $AA_OFF is deselected in the case of a RESET
Bit1 = 1: $AA_OFF is retained beyond the RESET

In JOG mode, an overlaid movement can take place

Also in JOG mode, if $AA_OFF changes, an interpolation of the position offset can be set as an overlaid movement via machine data MD 36750: AA_OFF_MODE.

Bit2 = 0: No overlaid movement on the basis of $AA_OFF
Bit2 = 1: An overlaid movement on the basis of $AA_OFF

If a position offset is interpolated on the basis of $AA_OFF, a mode change can only occur after JOG once the interpolation of the position offset is complete. Otherwise alarm 16907 is signaled.

Activation/ Deactivation

The programmed conditions of the current motion-synchronous actions are recorded in interpolation time, until the conditions are met or the end of the subsequent block is reached with the machine in operation.

In software version 3.2 and later, the introduction of an $$ main variable approved for synchronized actions results in a comparison of the synchronization conditions in interpolation time in the main run.

Restrictions

- **Interrupt routines/asynchronous subroutines**
  When an interrupt routine is activated, modal motion-synchronous actions are retained and are also effective in the asynchronous subroutine. If the subroutine return is not made with REPOS, the modal synchronized actions changed in the asynchronous subroutine continue to be effective in the main program.
- **REPOS**
  In the remainder of the block, the synchronized actions are treated in the same way as in an interruption block. Modifications to modal synchronized actions in the asynchronous subprogram are not effective in the interrupted program. Polynomial coefficients programmed with FCTDEF are not affected by ASUB and REPOS.

  The coefficients from the call program are applied in the asynchronous subprogram. The coefficients from the asynchronous subprogram continue to be applied in the call program.
- **End of program**
  Polynomial coefficients programmed with FCTDEF remain active after the end of program.
- **Block search**
  During block search with calculation, these polynomial coefficients are collected, i.e. written to the setting data.

CORROF SW 6 and later

- The part program command CORROF with DROF is also collected during a block search and output in an action block. In the last block handled by the search run with CORROF or DROF, all the deselected DRF offsets are collected for reasons of compatibility.

  A CORROF with AA_OFF is not collected during a block search and is lost. If a user wishes to continue to use this search run, this is possible by means of block search via “SERUPRO” program testing. More detailed information about these block searches can be found in:

References: /FB1/, K1 “Mode Group, Channel, Program Operation Mode”, Program Testing

- **Axis-specific deselection of DRF offsets with CORROF**
  With CORROF, DRF offsets for individual axes are only possible from the part program.
- **Position offset deselection during active synchronized actions**
  If a synchronized action is active when deselecting the position offset by means of the part program command COROFF(axis,“AA_OFF”), alarm 21660 is signaled. $AA_OFF is deselected simultaneously and not set again. If the synchronized action becomes active later in the block after CORROF, $AA_OFF remains set and a position offset is interpolated.

References: /PG/, “Programming Guide: Fundamentals”

Note

The coordinate system (BCS or WCS) in which a real-time variable is defined determines whether frames will or will not be included.

Distances are always calculated in the set basic system (metric or inch). A change with G70 or G71 has no effect.

DRF offsets, zero offsets external, etc., are only taken into consideration in the case of real-time variables that are defined in the machine coordinate system.

2.4.7 Online tool offset FTOC

Online tool offset

Machining of the workpiece and dressing of the grinding wheel for grinding applications can be implemented either in the same or in different channels (machining and dressing channel).

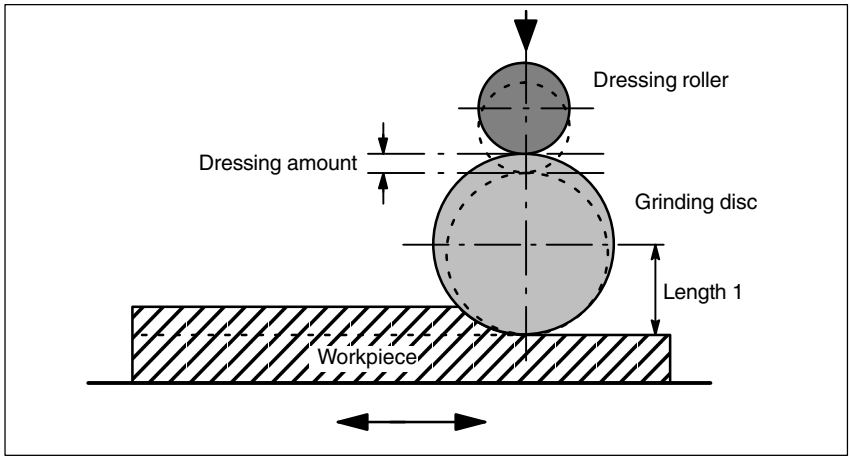


Fig. 2-8 Dressing during machining using a dressing roller

References: /FB/, W4 “Grinding”

Supplementary condition

Synchronized action FTOC is available in SW 3.2 and later.

An online offset allows an overlaid movement to be implemented for a geometry axis according to a polynomial programmed with FCTDEF (see Subsection 2.4.4) as a function of a reference value (e.g. actual value of an axis).

Programming of FTOC

The online offset is specified as follows:

```
FTOC      (Polynomial no.
           Read_real_main_variable      ;Reference value
           Length 1_2_3,
           Channel number
           Spindle number)
```

Parameters

Polynomial no.:	Number of function programmed beforehand with FCTDEF.
Read_real_main_variable:	All main variables listed in Subsection 2.3.11 of the REAL type may be used.
Length 1_2_3:	Wear parameter to which the offset value **is added**.

Channel number:	Target channel in which the offset must be applied. This enables simultaneous dressing from a parallel channel. If the channel number is missing, the offset takes effect in the active channel. Online offset with FTOCON must be activated in the offset target channel.
Spindle number:	The spindle number is programmed if a non-active grinding wheel needs to be dressed. “Constant peripheral speed” or “tool monitoring” must be active for this purpose. If no spindle number is programmed, then the activetool is compensated.

Example

Compensate length of an active grinding wheel

```
%_N_DRESS_MPF

FCTDEF(1,–1000,1000,–$AA_IW[V],1)          ;Definition of function
ID=1 DO FTOC(1,$AA_IW[V],3,1)  ; Select online tool offset:
                               ; Derived from the motion of the V axis,
                               ; length 3 of the active grinding wheel is
                               ; compensated in channel 1.
WAITM (1,1,2)                  ; Synchronization with machining channel
G1 V–0.05 F0.01, G91
G1 V –....
...
CANCEL(1)                      ; deselect online offset
...
```

Note

No frequency keyword nor any condition is programmed in the synchronized action. The FTOC action is therefore active in every interpolation cycle with no dependencies other than $AA_IW[V].

2.4.8 RDISABLE

Programmed read-in disable RDISABLE

An RDISABLE command in the active section causes block processing to be stopped if the relevant condition is fulfilled. Processing of programmed motion-synchronous actions still continues. The read-in disable is canceled again as soon as the condition for the RDISABLE is no longer fulfilled.

An exact stop is initiated at the end of the block containing RDISABLE irrespective of whether or not the read-in disable is still active.
Application:
This method can be used, for example, to start the program in the interpolation cycle as a function of external inputs.

Example of RDISABLE

Programmed read-in disable

```
WHENEVER $A_INA[2]<7000 DO RDISABLE
...
N10 G01 X10      ;RDISABLE takes effect at the end of N10 if the condition
                  is fulfilled while N10 is being processed.
N20 Y20
```

Program processing is halted if the voltage at input 2 drops to below 7 V (assuming that the value 1000 corresponds to 1 V). Example application of this method: Read-in disable until obstruction is removed from path.

2.4.9 STOPREOF

End of preprocessing stop with STOPREOF

A motion-synchronous action containing an STOPREOF command cancels the existing preprocessing stop if the condition is fulfilled. STOPREOF must always be programmed with keyword 'WHEN' and as a non-modal command.
Application: Fast program branch at end of block.

Example of STOPREOF

Program branches

```
WHEN $AC_DTEB<5 DO STOPREOF
G01 X100
IF $A_INA[7]>5000 GOTOF Label 1
```

If the distance to the end of the block is less than 5 mm, end preprocessing stop. If the voltage at input drops below 5V, jump forwards to label 1 (assuming that the value 1000 corresponds to 1 V).

2.4.10 DELDTG

Deletion of distance-to-go

Synchronized actions can be used to activate deletion of distance-to-go for the **path** and for specified **axes** as a function of a condition.

- High-speed prepared deletion of distance-to-go

High-speed, prepared DDTG

High-speed/prepared deletion of distance-to-go is used in time-critical applications, i.e.:

- If the time between deletion of distance-to-go and start of next block needs to be very short
- If there is a high probability that deletion of distance-to-go will be activated

DELDTG

Deletion of distance-to-go is programmed with synchronized action **DELDTG.** After the distance-to-go has been deleted, the remaining path distance is stored in $AC_DELT. Continuous-path mode is thus interrupted at the end of the block with high-speed deletion of distance-to-go.

Restrictions:
Deletion of distance-to-go for the path may only be programmed as a non-modal synchronized action. If tool radius compensation is active, fast deletion of distance-to-go cannot be used.

Commands: MOVE=1: Works for indexing axes with and without Hirth serration MOV=0: Same function for both: approaches the next position. Command: DELDTG. In the case of indexing axes without Hirth tooth system: Axis stops immediately. In the case of indexing axes with Hirth tooth system: Axis traverses to next position.

Example DELDTG

```
... DO DELDTG
N100 G01 X100 Y100 F1000
N110 G01 X...
IF $AC_DELT > 50 ...
```

High-speed, prepared DDTG for axes

High-speed, prepared deletion of distance-to-go for axes must be programmed as a non-modal action.

Application:
A positioning motion programmed in the part program is halted by means of axial deletion of distance-to-go. Several axes can be stopped simultaneously with one command.

... DO **DELDTG(axis1, axis2, ...)**

Examples of DELDTG(axis)

```
WHEN $A_INA[2]>8000 DO DELDTG(X1)
                                  ; If the voltage at input 2 exceeds
                                  ; 8 V, delete distance-to-go
                                  ; for axis X1
POS[X1] = 100                     ; Next position
R10 = $AA_DELT[X 1]               ; Apply axial distance-to-go in R10
```

Once the distance-to-go has been deleted, the variable $AA_DELT[axis] will contain the axial distance-to-go.

(assuming that the value 1000 corresponds to 1 V).

2.4.11 Disabling a programmed axis motion

Task

The axis is programmed within a machining routine and, in particular circumstances, must be not started at the beginning of a block.

Solution

A synchronized action is used to maintain a 0 override until it is time for the axis to be started.

Example:

```
WHENEVER $A_IN[1]==0 DO $AA_OVR[W]=0
G01 X10 Y25 F750 POS[W]=1500FA[W]=1000
                    ; The positioning axis is started asynchronously
                    ; to path machining;
                    ; the enable is set via a digital input
```

Note

Axis motion disable can also be programmed for PLC axes (e.g. magazine axis).

2.4.12 Starting command axes

Introduction

Axes can be positioned, started and stopped completely asynchronously to the part program from synchronized actions. This type of programming is advisable for cyclic sequences or sequences that are strongly dependent on events. Axes started by synchronized actions are called **command axes.**

Control from the PLC

Autonomous individual axis operations (SW 6.3 and later)

A command axis interpolated from the main run (started by static synchronized actions) reacts independently of the NC program in the event of NC Stop, alarm handling, end of program, program control and reset, when control of the command axis has been taken over from the PLC.

Control via the command axis occurs via the axial VDI interface (PLC→NCK) with the “PLC controls axis” interface (DB31, ... DBX28.7) == 1

For more information about the precise sequence of operations of the various steps for transferring control of the command axis to the PLC, please see:
References: /FB/, P2, “Positioning Axes”

Supplementary condition

An axis cannot be moved from the part program and from synchronized actions simultaneously, but may be moved from these two sources successively. Delays may occur if an axis has been moved first from a synchronized action and then programmed again in the part program.

Note

Check and, if necessary, correct MD 30450: IS_CONCURRENT_POS_AX indicates whether the axis is primarily intended as a command axis or for programming by the part program:
0: Not a competing axis
1: Competing axis (command axis)

Example 1

... ID=1 EVERY $A_IN[1]==1 DO **POS[X]=100** ...

Example 2

An axis motion can be initiated in the form of a technology cycle (see Section 2.5). Main program:

```
...
ID=2 EVERY $A_IN[1]==1 DO AXIS_X
...
```

Axis program:

```
AXIS_X:
        M100
        POS[X]=100
        M17
```

Programming

Positioning axis motions are programmed in synchronized actions as they are from the part program:

ID = 1 EVERY $AA_IM[B] > 75 DO **POS[U]=100**

The programmed position is evaluated in inches or in the metric system depending on whether setting G70 or G71 is active in the current part program block.

G70/G71 and G700/G710 can also be programmed directly in synchronized actions in SW 5 and later.

This allows the inch/metric evaluation of a command axis movement to be defined independent of programming in the part program.

ID = 1 WHENEVER $A_OUT[1] ==1 DO **G710** POS[X]=10

ID = 2 EVERY **G710** $AA_IM[Z] >100 DO **G700** POS[Z2]=10

Note

Only **G70, G71, G700, G710** can be programmed in synchronized actions! See Section 2.1.

G functions, which are programmed in the synchronized action block, are only effective for the synchronized action or within the technology cycle. They have no effect on subsequent blocks in the part program.

References: /PG/ Chapter 3 “Positional Parameters”

Absolute/ incremental end position

The end position can be programmed either absolutely or incrementally. The position is approached absolutely or incrementally depending on whether G90 or G91 is active in the main program block currently being processed. It is possible to explicitly program whether the value must be interpreted as an absolute or incremental setting:

IC: Incremental
AC: Absolute
DC: Direct, i.e. position rotary axis via shortest route
ACN: Position modulo rotary axis absolutely in negative direction of motion
ACP: Position modulo rotary axis absolutely in positive direction of motion
CAC: Traverse axis to coded position absolutely
CIC: Traverse axis to coded position incrementally
CDC: Traverse rotary axis to coded position via shortest route
CACN: Traverse modulo rotary axis to coded position in negative direction
CACP: Traverse modulo rotary axis to coded position in positive direction

Coded positions are settings stored in machine data.

Example 1 fixed value

ID = 1 EVERY $AA_IM[B] > 75 DO **POS[U]=IC(10)**

; If event occurs, advance U axis by 10

Example 2 current value

The traversing path is generated in real time from a real-time variable:

ID = 1 EVERY $AA_IM[B] > 75 DO **POS[U]=$AA_MW[V]–$AA_IM[W] + 13.5**

Axial frames

The response of synchronized actions and axial frames is explained below:

Effect

When positioning motions are executed from synchronized actions, the axial offsets, scaling and mirroring functions of the programmable and settable frames (G54 etc.) as well as tool length compensations are all operative.

Whichever frame is operative in the current block takes effect. If a rotation is active in the current block, then an alarm is output to reject a positioning motion initiated from a positioning motion.

Example:

```
TRANS X20
IDS= 1 EVERY $A_IN==1 DO POS[X]=40
G1 Y100          ; If the input is set, X is positioned at 60
...
TRANS X–10
G1 Y10           ; If the input is set, X is positioned at 30
```

Suppression

The effect of frames and tool lengths can be suppressed by means of

MD 32074: FRAME_OR_CORRPOS_NOTALLOWED

Suppressing axial frames

Axial frames that travel incrementally to indexing positions have no effect on a command axis. Therefore,

bit 9 = 1 is set in MD 32074: FRAME_OR_CORRPOS_NOTALLOWED[AX4]

and the command axis is positioned with JOG.

Example:

```
RANS A=0.001
POS[A]=CAC(2)                    ; Axis travels to position 180.001 degrees

; The axial frame has no effect on the command axis
; MD 32074: FRAME_OR_CORRPOS_NOTALLOWED[AX4] = 'H0020'

WHEN TRUE DO POS[A]=CIC(–1) ; Axis travels to position 180.000 degrees.
```

Note

If a command axis travels to indexing positions incrementally, axial frames usually have **no** effect on this command axis.

2.4.13 Axial feedrate from synchronized actions

Feedrates

An axial feedrate can be programmed in addition to the end position:

ID = 1 EVERY $AA_IM[B] > 75 DO POS[U]=100 **FA[U]=990**

The axial feedrate for command axes is modal. It is programmed under address FA. The default value is set via axial machine data
MD 32060: POS_AX_VELO.

The feedrate value is either preset to a fixed quantity or generated in real time from real-time variables:

Example of calculated feedrate

ID = 1 EVERY $AA_IM[B] > 75 DO POS[U]=100 **FA[U]=$AA_VACTM[W]+100**

The feedrate value is programmed either as a linear or a rotational feedrate:
The feedrate type is determined by setting data:
SD 43300: $SA_ASSIGN_FEED_PER_REV_SOURCE.
This data can be altered by an operator input, from the PLC or from the part program. In synchronism with the part program context, the feedrate type can be switched over using the NC commands FPRAON, FPRAOF.
See also:

References: /FB/, V1 “Feedrates”

Note

The axial feedrate from motion-synchronous actions is not output as an auxiliary function to the PLC. Parallel axial technology cycles would otherwise block one another.

2.4.14 Starting/Stopping axes from synchronized actions

Starting/stopping

Command axes can be stopped from synchronized actions even when no end position has been specified. In this case, the axis is traversed in the programmed direction until another motion is set by means of a new motion or positioning command or until the axis is halted by a stop command. This method can be used, for example, to program an endlessly turning rotary axis.

Starting and stopping are programmed using the same method as positioning motions.

MOV[axis]=Value

The value is an INT data type.
The sign of the value determines the direction of motion:
> 0: Axis motion in positive direction
<0: Axis motion in negative direction
==0: Stop axis motion

If a moving indexing axis is halted by command MOV[axis]=0, then the next indexing position is approached in the same way as in JOG mode.

The **feedrate** for the motion can be programmed with **FA[axis]=value** (see above). If no axial feedrate is programmed, the feedrate value is derived from an axis motion that may already be activated from synchronized actions or from the axis velocity set via MD 32060: POS_AX_VELO.

Example

```
... DO MOV[u]=0        ; Stop axis motion as soon as condition has been fulfilled
```

2.4.15 Spindle motions from synchronized actions

General

Analogously to positioning axes, it is also possible to start, position and stop **spindles** from synchronized actions. Spindle movements can be started at defined points in time by blocking a spindle motion programmed in the part program or by controlling the axis motion from synchronized actions.

Starting/stopping

The use of these functions is recommended for cyclic operations or for operations that are predominantly event-controlled.

Stop until event occurs

Application:
A spindle is programmed within a machining routine, but must not be started at the beginning of the block in particular circumstances. A synchronized action is used to maintain a 0 override until the spindle is to start.
Example:

```
ID=1 WHENEVER $A_IN[1]==0 DO $AA_OVR[S1]=0
G01 X100 F1000 M3  S1=1000
                   ; The spindle is started asynchronously to path machining;
                   ; the spindle is started via a digital input
```

Auxiliary functions, speed, position

These functions are programmed in the action section of the synchronized action by exactly the same method as used in the part program.
Commands: S= ..., M3, M4, M5, SPOS= ...
Example:
ID = 1 EVERY $A_IN[1]==1 DO M3 S1000
ID = 2 EVERY $A_IN[2]==1 DO SPOS=270

If there is no numeric extension, each of the commands applies to the master spindle. By specifying a numeric extension, it is possible to activate each spindle individually:

ID = 1 EVERY $A_IN[1]==1 DO M1=3 S1=1000 SPOS[2]=90

The same rules are used for programming the type of positioning as for positioning axes (see above).

If **concurrent commands** are input via simultaneously active synchronized actions for an axis/spindle, then the commands are applied in the **chronological sequence** in which they are programmed.

Example:
ID=1 EVERY $A_IN[1]==1 DO M3 S300 ; rotational direction and speed
ID = 2 EVERY $A_IN[2]==1 DO M4 S500; rotational direction and speed
ID=3 EVERY $A_IN[3]==1 DO S1000 ; new speed setting
; for active spindle rotation
ID=4 EVERY ($A_IN[4]==1) AND ($A_IN[1]==0) DO SPOS=0
; position spindle

Feedrate

The feedrate for “Position spindles” can be programmed from a synchronized action with command:

FA[Sn]= ...

:

Note

Only a modal data item is available for the feedrate of synchronized actions for spindle mode and axis mode. FA[S] and FA[C] are supplied in the same way.

SW limit switches, working area limitations

The restrictions imposed by SW limit switches and working area limitations also apply to axis/spindle movements activated from synchronized actions.

Influence of limitations on movements from synchronized actions

Working area limitations programmed by G25/G26 are taken into account as a function of setting data: SD 43400: WORKAREA_PLUS_ENABLE.

Activation and deactivation of working area limitations by G functions WALIMON/WALIMOF **in the part program** does not affect command axes.

Correcting acceleration

ACC[axis]=0..200 can be used to change the acceleration preset in MD 32300: MAX_AX_ACCEL in a range from 0% to 200% with:
ACC[axis]=<Value> (effective in the part program and in synchronized actions).

Axis coordination

If a positioning command (POS, MOV) is started from synchronized actions for an axis that is already operating as a path or PLC axis, then processing is aborted with an alarm.

Axis movement by PP and SA alternately

In typical cases, an axis is either moved from the part program (PP) in motion blocks or as a positioning axis from a synchronized action (SA). However, if the same axis must be traversed alternately from the part program as a path axis or positioning axis and from synchronized actions, then a coordinated transfer takes place between both axis motions.

Example

; Traverse X axis alternately from part program and from synchronized actions

```
N10 G01 X100 Y200 F1000          ; X axis programmed in part program
...
N20 ID=1 WHEN $A_IN[1]==1 DO POS[X]=100 FA[X]=200
                                 ; Start positioning from synchronized action,
                                 ; if digital input set
...
CANCEL(1)                        ; Deselect synchronized action
...


N100 G01 X100 Y200 F1000         ; X: Path axis
                                 ; Delay before movement if digital
                                 ; input was 1 and X was therefore
                                 ; positioned by a synchronized action
```

On-the-fly transitions

Transitions can be made between command axes and spindles.

Initial situation

Since several synchronized actions can be active simultaneously, the situation may arise where an axis motion is started when the axis is already active.

Procedure

In this case, the **most recently activated** motion is applicable. POS and MOV motions can be activated alternately. When a reversal in the direction of motion is forced in this manner, the axis is first decelerated and then positioned in the opposite direction.

Examples:
ID=1 EVERY $AC_TIMER[1] >= 5 DO POS[V]=100 FA[V]=560
ID=2 EVERY $AC_TIMER[1] >= 7 DO POS[V]=$AA_IM[V] + 2 FA[V]=790
; Due to the programming of $AC_TIMER[1], the synchronized action with ID=2 is the most recently activated action. Its commands are applied in place of the commands in ID=1.

The end position and feedrate for a command axis can therefore be adjusted while the axis is in motion.

Example: Activation by signal

```
ID=1   EVERY $A_IN[1]==1 DO POS[U]=$AA_IM[U]+$AA_IM[V]*.5
                            FA[U]=$AA_VACTM[U]+10
```

Legal transitions Transitions marked with x are legal:

from ↓ to →	POS	MOV=1 MOV= – 1	MOV=0	SPOS	M3 M4	M5	LEADON	TRAIL ON
Axis stationary								
Axis mode	x	x	x	x	x	x	x	x
Position-controlled spindle	x	x	x	x	x	x		
Speed-controlled spindle				x	x	x		
Axis in motion								
Axis mode	x	x	x				x	x
Position-controlled spindle								
Speed-controlled spindle				x	x	x		

Transitions not marked with an x are rejected with an alarm.

Example: Legal transition

```
N10 WHEN $AA_IM[Y] >= 5 DO MOV[Y]=–1      ; In position +5, start
                                          ; axis in negative
                                          ; direction
N20 WHEN TRUE DO POS[Y]=20 FA[Y]=500      ; Start Y axis when
                                          ; block is loaded
```

On-the-fly transitions for axis couplings

Positioning axis motions and movements resulting from axis couplings programmed via synchronized actions can be activated alternately.

– See Subsection 2.4.17 and

References: /M3/, Coupled Motions and Master Value Couplings

Legal transitions in master value couplings are marked by LEADON in the above table. Legal transitions in coupled motions are marked by TRAILON.

2.4.16 Setting actual values from synchronized actions

Application

The PRESETON function can be used to redefine the control zero in the machine coordinate system.

Function

When Preset is applied, the axis is not moved. A new position value is merely entered for the current axis position.

Programming

The value for **one** axis can be programmed in each synchronized action.
For example:

WHEN $AA_IM[a] >= 89.5 DO PRESETON(a, 10.5)

With PRESETON(axis,value)

Axis:	Axis whose control zero must be altered
Value:	Amount by which control zero must be altered.

Permissible applications

PRESETON from synchronized actions can be programmed for

- Modulo rotary axes that have been started from the part program and
- All command axes that have been started from a synchronized action

Restrictions

PRESETON cannot be programmed for axes, which are involved in a transformation.

Example

Please see Subsection 6.7.3 for an example of how to use PRESETON in conjunction with an “On-the-fly parting” application.

Note

The “PRESETON” preset actual value memory may only be programmed using the keywords “WHEN” or “EVERY”.

2.4.17 Coupled motions and activating/deactivating couplings

Introduction

The following functions are described in detail in:

References: /FB/, M3, Coupled Motions

The following functions are described in detail:

- Coupled motion slave axis (axes) is (are) linked to a master axis via a coupling factor.
- Curve tables Curve tables represent a (complex) relationship between the master and slave values. The following may be applied as master values:
  - Setpoints generated by the control
  - Actual values measured by encoders
  - Externally specified quantities

    Situations where a slave axis is linked to a master axis by means of a curve table are particularly relevant with respect to synchronized actions.
- Master value coupling of the following master value couplings, which may be implemented for part programs:
  - Axis master value coupling
  - Path master value coupling

    only axis master value couplings are available for use in synchronized actions.

Coupled motion

From a synchronized action it is possible to define and simultaneously activate the assignment between a slave axis and a master axis using a coupling factor:

... DO **TRAILON**(FA, LA, Kf)

Where:

FA	Slave axis
LA	Master axis
Kf	Coupling factor

The commands for separating the coupled-axis grouping are as follows:

... DO **TRAILOF**(FA, LA, LA2)

Where: FA	Slave axis
LA	Master axis
LA2	Master axis 2, optional

Curve tables

The relationship between a master quantity and a slave quantity stored in curve tables can be utilized in synchronized actions in the same way as other REAL functions (e.g. SIN, COS):

Calculate slave value

The slave value calculated from a master value on the basis of curve table n must be assigned to an arithmetic variable.
Example:

... DO $R17=**CTAB**(LW, n, deg)

Where:

MV	Master value
n	Number of curve table
deg	Gradient parameter, result (2 additional opt. parameters for scaling: – Slave axis – Master axis)

Example:
DEF REAL GRADIENT
...
WHEN $A_IN[1] == 1 DO $R17 = CTAB(75.0, 2, GRADIENT)

Calculate master value

From a synchronized action it is possible to calculate a concrete master value for a slave value on the basis of a curve table.
Example:

... DO $R18=**CTABINV**(SV, aprMV, n, deg)

Where:

SV	Slave value
aprMV	Master value being approached, which can be used to determine a unique MV if the reverse function in the curve table is ambiguous
n	Number of the curve table
deg	Gradient parameter, result (2 additional opt. parameters for scaling: – Slave axis – Master axis)

The CTAB and CTABINV functions can be programmed both in conditions and in the action section of synchronized actions.

Axis master value coupling

The coupling between slave axis FA and naster axis LA based on the stored curve table with number NR is called in the action section of synchronized actions as follows:

... DO **LEADON**(FA; LA, NUM)
Where:

FA	Slave axis
LA	Master axis
NUM	Number of the curve table

Deactivate axis coupling from synchronized action

If the axis master value coupling must be canceled again on the fulfillment of another condition, the action must be programmed as follows:

... DO **LEADOF**(FA, LA)

System variables

The system variables of the master value coupling as specified in the list of system variables can be read/written from the part program and synchronized actions.

See Subsection 2.3.11.

Detection of synchronism

System variable $AA_SYNC[ax] can be read from the part program and synchronized action and indicates whether and in what manner slave axis FA is synchronized:

0: Not synchronized

1: Coarse synchronism (acc. to MD 37200: COUPLE_POS_TOL_COARSE)

2: Fine synchronism (acc. to MD 37210: COUPLE_POS_TOL_FINE)

Definition of application

Couplings directly activated in the part program are activated at block limits. With the additional option of activating couplings from synchronized actions, it is possible to implement event-controlled, differential activation, e.g.

- From block beginning for specific axis path
- Up to block end for specific distance-to-go
- Appearance of digital input signals or
- Combinations of these

Page 2.1, Conditions

For more information about programming coupling functions and curve tables, please see:

Reference: /PGA/, Programming Guide Advanced

Note

Axes, which might be in any given motional state at the instant they are coupled via synchronized actions, are synchronized by the control system. For more details, please see Description of Functions M3.

Examples

Please see Subsection 6.7.3 for an example illustrating an axis coupling implemented by means of a curve table.

2.4.18 Measurements from synchronized actions

Introduction

There are the following measuring functions provided for part programs:
MEAS, MEAW, MEASA, MEAWA, MEAC

References: /PGA/, Programming Guide Advanced
/FB/, M5, “Measurements”

Only the following may be used in synchronized actions:

- MEAWA Axial measurement without deletion of distance-to-go
- MEAC Axial, continuous measurement

While measuring functions are limited to one block at a time in part program motion blocks, they can be activated and deactivated any number of times from synchronized actions:

Note

With static synchronized actions, measurements are also available in JOG mode.

Programming

MEAWA[axis]=(mode, trigger_event_1, trigger_event_2,
trigger_event_3, trigger_event_4)
; Activate axial measurement without deletion of
; distance-to-go

MEAC[axis]=(mode, measurement memory, trigger_event_1, trigger_event_2,
trigger_event_3, trigger_event_4)
; Activate axial, continuous measurement

Axis: Axis for which measurement is taken

Table 2-3 Mode meanings:

Tens decade	Units decade	Meaning
	0	Abort measurement job
	1	Up to 4 trigger events can be activated **simultaneously**
	2	Up to 4 trigger events can be activated **consecutively**
	3	Up to 4 trigger events can be activated **consecutively**, but with no monitoring of trigger event 1 on START
0		Active meas. system
1		1st measuring system
2		2nd measuring system
3		Both measuring systems

Trigger_event_1 to trigger_event_4:

1:	Rising edge probe 1	
–1:	Falling edge probe 1	*optional*
2:	Rising edge probe 2	*optional*
–2:	Falling edge probe 2	*optional*

Measurement memory: Number of a FIFO variable

Measured values are supplied exclusively for the **machine** coordinate system.

MEAWA

... DO **MEAWA**[axis]=(, , , ,) ;Axial measurement without deletion of distance-to-go

Deletion of distance-to-go can be called explicitly in the synchronized action, see Subsection 2.4.10 and example below.

GEO axes and axes involved in transformations can be programmed individually.

Programming:
The programming method is identical to that used in the part program

---

Note

System variable $AC_MEA does not supply any useful information about the validity of a measurement called from a synchronized action. Only one measurement job at a time may be active for an axis.

System variables:

$AA_MEAACT[axis]	Supplies the instantaneous measuring status of an axis.
1	Measurement active
0	Measurement not active
$A_PROBE[probe]	Supplies the instantaneous status of the probe.
1	Probe switched, high signal
0	Probe not switched, low signal

Measured values in machine coordinate system with 2 probes (encoders):

$AA_MM1[axis]	Trigger event 1, encoder 1
$AA_MM2[axis]	Trigger event 1, encoder 2
$AA_MM3[axis]	Trigger event 2, encoder 1
$AA_MM4[axis]	Trigger event 2, encoder 2

MEAC

... DO **MEAC**[axis]=(mode, No_FIFO, trigger events)

$AC_FIFO variables (see Subsection 2.3.6) are provided for the purpose of storing measured values from cyclic measuring processes. See above for mode and trigger events.

Examples: Two FIFOs have been set up in machine data for the following examples.

Machine data

```
MD 28050: MM_NUM_R_PARAM = 300
MD 28258: MM_NUM_AC_TIMER = 1
MD 28260: NUM_AC_FIFO = 2          ; 2 FIFOs
MD 28262: START_AC_FIFO = 100      ; First FIFO starts at R100
MD 28264: LEN_AC_FIFO = 22         ; Each FIFO can store 22 values
MD 28266: MODE_AC_FIFO = 0                ; No summation
```

Example 1.
All rising edges of probe 1 must be recorded on a path between X0 and X100. It is assumed that no more than 22 edges will occur.

Program 1:

```
DEF INT ANZAHL
DEF INT INDEX_R
N0    G0 X0
N1    MEAC[X]=( 1, 1, 1) POS[X]=100
                                   ; Mode = 1, simultaneous
                                   ; No_FIFO      = 1
                                   ; Trigger event 1= Rising edge, encoder 1
N2    STOPRE                       ; Stop preprocessing
N3    MEAC[X]=( 0)                 ; Cancel continuous measurement
N4    NUMBER= $AC_FIFO1[4]         ; Number of measured values stored in FIFO variables
N5    NUMBER= NUMBER – 1
N6    FOR INDEX_R= 0 TO NUMBER
N7    R[INDEX_R]= $AC_FIFO1[0]     ; Enter FIFO content in R0 – ...
N8    ENDFOR                       ; Following a read operation, the FIFO variable is empty
```

Example 2.
All rising and falling edges of probe 1 on a path between X0 and X100 must be recorded. The number of trigger events that may occur is unknown. This means that the measured values must be fetched and stored in ascending order in R1 as a parallel operation in one synchronized action. The number of stored measured values is entered in R0.

Program 2:

```
N0    G0 X0                        ; Rapid traverse to starting point
N1    $AC_MARKER[1]=1              ; Marker 1 as index for computing variables R[..]
N2    ID=1 WHENEVER $AC_FIFO1[4]>=1
      DO $R[$AC_MARKER[1]]= $AC_FIFO1[0] $AC_MARKER[1]=$AC_MARKER[1]+1
                                   ; Synchronized action as test:
                                   ; If 1 or more measured values are stored in the FIFO variable,
                                   ; read the oldest value from the FIFO and save it to the current R[ ..],
                                   ; increment index for R by 1
```

```
N3   MEAC[X]=( 1, 1, 1, –1)  POS[X]=100   ; Activate continuous measurement, motion
                                          ; after X = 100
                                          ; Mode = 1, simultaneous
                                          ; No_FIFO = 1
                                          ; Trigger event 1= 1, rising edge encoder 1
                                          ; Trigger event 2= –1, falling edge encoder 1
N4   MEAC[X]=(0)                          ; Deselect measurement
N5   STOPRE                               ; Stop preprocessing
N6   R0= $AC_MARKER[1]                    ; Number of values stored in R0
```

Example 3 Continuous measurement with explicit deletion of distance-to-go after 10 measurements

Program 3:

```
N1   WHEN $AC_FIFO1[4]>=10
     DO MEAC[X]=(0) DELDTG(X)             ; End condition as synchronized action:
                                          ; If the FIFO variables contain 10 or more measured
                                          ; values
                                          ; deselect continuous measurement and
                                          ; delete distance-to-go
N2   MEAC[X]=( 1,1,1,–1) G01 X100 F500    ; Continuous measurement activated in part program
                                          ; Mode = 1, simultaneous
                                          ; Nr_FIFO = 1, FIFO variable 1
                                          ; Trigger event 1= 1, rising edge encoder 1
                                          ; Trigger event 2= –1, falling edge encoder 1
N3   MEAC[X]=(0)                          ; Deselect continuous measurement
N4   R0= $AC_FIFO1[4]                     ; Actual number of measured values
```

Priority with more than one measurement

Only one measurement job can be active for an axis at any given time.

If a measurement job for the same axis is started, the trigger events are re activated and the measurement results reset. The system does not react in any special way if Deactivate measurement job (mode 0) is programmed when no measurement job has been activated beforehand. Measurement jobs started from the part program cannot be influenced from synchronized actions. An alarm is generated if a measurement job is started for an axis from a synchronized action when a measurement job from the part program is already active for the same axis. If a measurement job is already in progress from a synchronized action, a measurement job from the part program cannot be started at the same time.

Measurement jobs and status changes

When a measurement job has been executed from a synchronized action, the control system responds in the following way:

State	Procedure
Mode change	A measurement job activated by means of a modal synchronized action is not affected by a change in operating mode. It remains active beyond block limits.
RESET	Measurement job is aborted

State	Procedure
Block search	Measurement jobs are collected, but not activated until the programmed condition is fulfilled.
REPOS	Activated measurement jobs are not affected.
End of program	Measurement jobs started from static synchronized actions remain active.

2.4.19 Setting and deletion of wait markers for channel synchronization

Introduction

Coordination of operational sequences in channels is described in:

References: /FB/, K1, Mode Group, Channel, Program Operation Mode

The following of the functions described in this document, may be legally used in synchronized actions:

Set wait marker

Command **SETM** (marker number) can be programmed in the part program and the action section of a synchronized action. It sets the marker (marker number) for the channel in which the command is applied (own channel).

Delete wait marker

Command **CLEARM** (marker number) can be programmed in the part program and the action section of a synchronized action. It deletes the marker (marker number) for the channel in which the command is applied (own channel).

2.4.20 Setting alarm/error reactions

Error situations

"Set alarm" is one way of reacting to error states.

Application:
The SETAL command can be programmed to set cycle alarms from synchronized actions.

The following reactions can also be programmed as a response to errors:

- Stop axis See Subsection 2.4.11
- Set output See Subsection 2.4.2
- Other actions described in Section 2.4

Example set alarm

```
ID=67 WHENEVER $AA_IM[X1] – $AA_IM[X2] < 4.567 DO SETAL(61000)

; Set alarm if distance (actual value of axis X1 – actual value of axis X2)
; exceeds the critical value 4.567.
```

Cycles and cycle alarms

For information about cycles and cycle alarms, please see

References: /PGC/, Programming Guide Cycles

2.4.21 Evaluating data for machine maintenance

Function

Machine operators are able to use system variables in part programs, synchronized actions and via the OPI interface (even from a PLC or HMI) to access information about the use of the machine.

Maintenance measures can then be taken directly or requested on the basis of the values read out.

Saving

The system variables for machine maintenance are stored in SRAM. This means that they are retained after POWER ON.

Note

In contrast, the lubricant signal is only ever set if an axis path stored in a machine data has been exceeded since POWER ON. See Description of Functions – Basic Machine: Interface signals from Axis/Spindle.

Availability

The values for machine maintenance are available if the global NCK machine data MD ...: MM_MAINTENACE_MON is set and axis-specific machine data has been used to indicate which data should be provided for each axis involved.

MD: MM_MAINTENACE_MON is used to activate the function of and prepare the memory for the values indicated in the axis-specific MD. Changes to MD ...: MM_MAINTENACE_MON take effect at POWER ON.

Axis-specific values:

The following information can be entered in bit-coded format in MD ...: MAINTENACE_DATA:

Bit 0:	Total travel distance, total travel time and travel count of the axis
Bit 1:	Total travel distance, total travel time and travel count at **high axis speeds** High speeds are >= 80% of the maximum axis speed
Bit 2:	Total axis jerk, travel time with jerk and travel count with jerk
Bits 3 – 15:	Reserved

Configuration example

$MN_MM_MAINTENANCE_MON = TRUE

$MA_MAINTENANCE_DATA[0]=1
$MA_MAINTENANCE_DATA[1]=1
$MA_MAINTENANCE_DATA[2]=1

Activates the system variables for total travel distance, total travel time and travel count for the first 3 axes

System variables

The following system variables

$AA_TRAVEL_DIST	Total travel distance in mm or degrees
$AA_TRAVEL_TIME	Total travel time in seconds
$AA_TRAVEL_COUNT	Total travel count
$AA_TRAVEL_DIST_HS	Total travel distance at high speeds in mm or degrees
$AA_TRAVEL_TIME_HS	Total travel time in seconds at high speeds
$AA_TRAVEL_COUNT_HS	Total travel count at high speeds
$AA_JERK_TOT	Total axis jerk in m/s^3
$AA_JERK_TIME	Axis travel time with jerk in seconds
$AA_JERK_COUNT	Axis travel count with jerk

can be read from the part program and from synchronized actions.

Example: Distance during part program processing

Repeat read-outs can be used for example to determine the total travel distance of an axis within an area of a part program.

```
; Start of processing area in part program
R1 = $AA_TRAVEL_DIST[X]
...
...                 ; End of processing area
R2 = $AA_TRAVEL_DIST[X]
R3 = R2 –R1         ; Total distance traveled by X axis during
                    ; processing of the processing area in the part program
```

2.5 Calling technology cycles

Definition

A technology cycle is a sequence of actions that are executed sequentially in the interpolation cycle. The actions described in Section 2.4 can be combined to form programs. From the user's point of view, these programs are subprograms without parameters.

Parallel processing in channel

Several technology cycles or actions can be processed simultaneously in the same channel. These cycles and actions are processed in parallel in the channel in one interpolation cycle.

Various processing methods

With respect to processing sequence, the user must select the most suitable method from the following options:

- Several actions in one synchronized action:
  All actions are executed simultaneously in the interpolation cycle in which the condition is fulfilled.
- Actions are joined to form a technology cycle:
  The actions in the technology cycle are processed sequentially in the interpolation cycle. One block is processed in each interpolation cycle. A distinction must be made between single-cycle and multi-cycle actions. A technology cycle is ended when its last action has been executed (generally after several interpolation cycles have passed).

Commands such as variable assignments in technology cycles are processed in **one** interpolation cycle. Other commands (e.g. movement of command axis, see Subsection 2.4.12) take **several** interpolation cycles to complete. If the function is complete (e.g. exact stop on positioning of axis), the next block is executed in the following interpolation cycle.

Each block requires at least one interpolation cycle. If a block contains several single-cycle actions, then these are all processed in one interpolation cycle. Fig. 2-9 provides examples to indicate which actions are single-cycle and which are multi-cycle.

Application

One possible application of technology cycles is to move each axis using a separate axis program.

Programming

A technology cycle can be activated as a function of a condition in a modal/ static synchronized action.

End of program is programmed with M02/M17/M30/RET.

Search path

The call search path is the same as for subprograms and cycles.

Example:

```
...
ID=1 EVERY $AA_IM[Y]>=10 DO AX_X                ; AX_X subprogram
                                                ; name for axis program for X axis
```

```
AX_X:                ; Axis program
POS[X]=$R[7] FA[X]=377
$A_OUT[1]=1
POS[X]=R10
POS[X]=–90
M30
```

Note

If the condition is fulfilled again while the technology cycle is being processed, the cycle is not restarted. If a technology cycle has been activated from a synchronized action of the WHENEVER type and the relevant condition is still fulfilled at the end of the cycle, then it will be restarted.

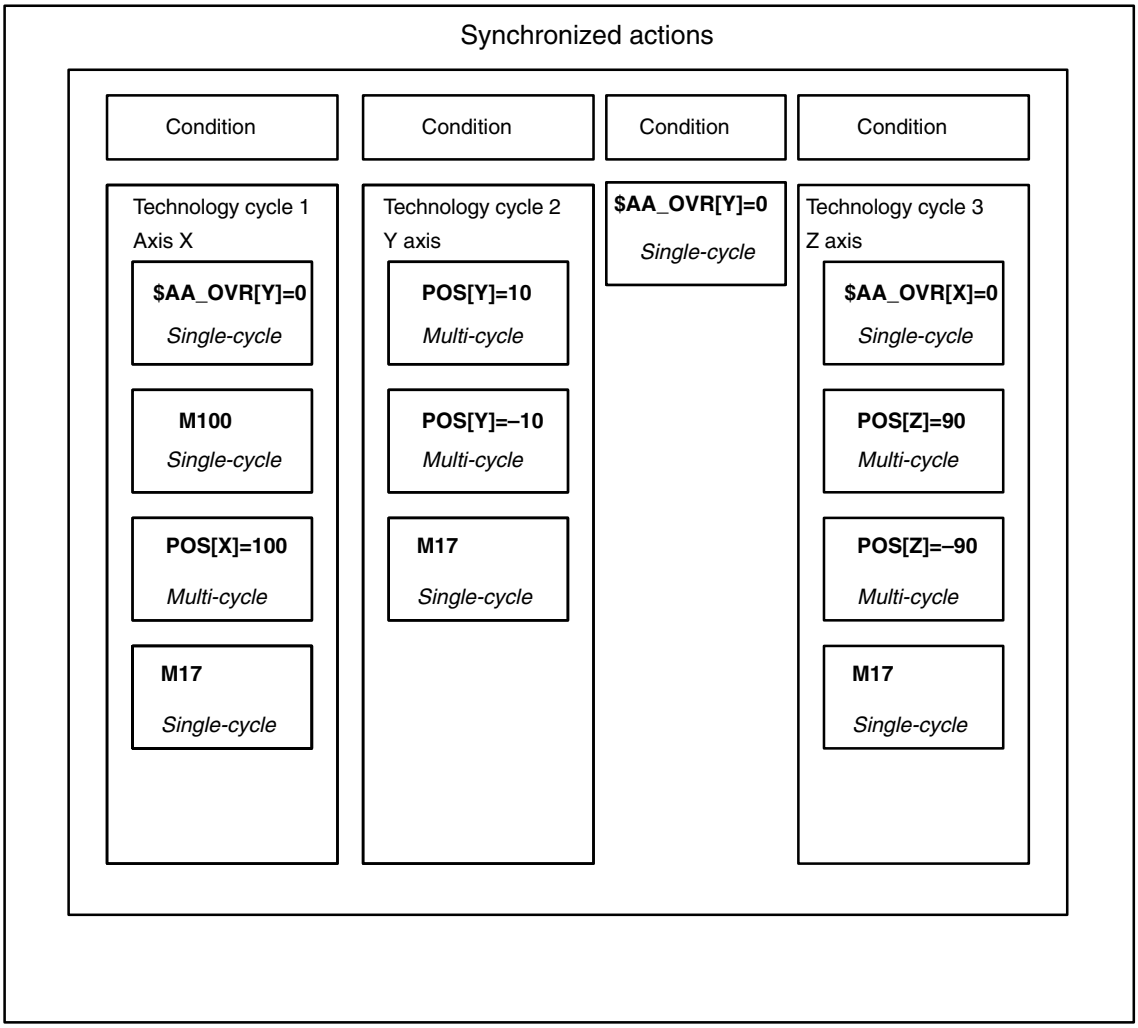


Fig. 2-9 Several technology cycles

Example (2) for coordinated axis motions:
Different axis programs can be started by setting digital NC inputs.

Main program:

```
...
ID=1 WHEN $A_IN[1]==1 DO ACHSE_X
ID=2 WHEN $A_IN[2]==1 DO ACHSE_Y
ID=3 WHEN $A_IN[3]==1 DO AA_OVR[Y]=0
ID=4 WHEN $A_IN[4]==1 DO ACHSE_Z
M30
```

Axis programs:

AXIS_X:

```
$AA_OVR[Y]=0
M100
POS[X]=100
M17
```

AXIS_Y:

```
POS[Y]=10
POS[Y]=–10
M17
```

AXIS_Z:

```
$AA_OVR[X]=0
POS[Z]=90
POS[Z]=–90
M17
```

2.5.1 Coordination of synchronized actions, technology cycles, part program (and PLC)

Control of technology cycles

Technology cycles/synchronized actions are controlled via the identification number of the synchronized action in which they are programmed as an action:

Means of coordination

Keyword	Meaning	PP	SA
	Call legal in part program Call legal in synchr. action/technology cycle	+	+
LOCK(ID)	Disable technology cycle. An active action is interrupted.		+
UNLOCK(ID)	UNLOCK continues the technology cycle at the point of interruption. An interrupted positioning operation is continued.		+
RESET(ID)	Abort technology cycle. Active positioning operations are aborted. If the technology cycle is restarted, then it is processed from the 1st block in the cycle. Depending on the type of synchronized action, actions are executed once more when the condition is fulfilled again. Completed synchronized actions of the WHEN type are not processed again after RESET.		+
CANCEL(ID)	Synchronized action is deleted.	+	

- LOCK(ID), UNLOCK(ID) by PLC see Subsection 2.6.1

Note

A synchronized action contains a technology cycle call. No further actions may be programmed in the same block in order to ensure that the assignment between ID number and relevant technology cycle is unambiguous.

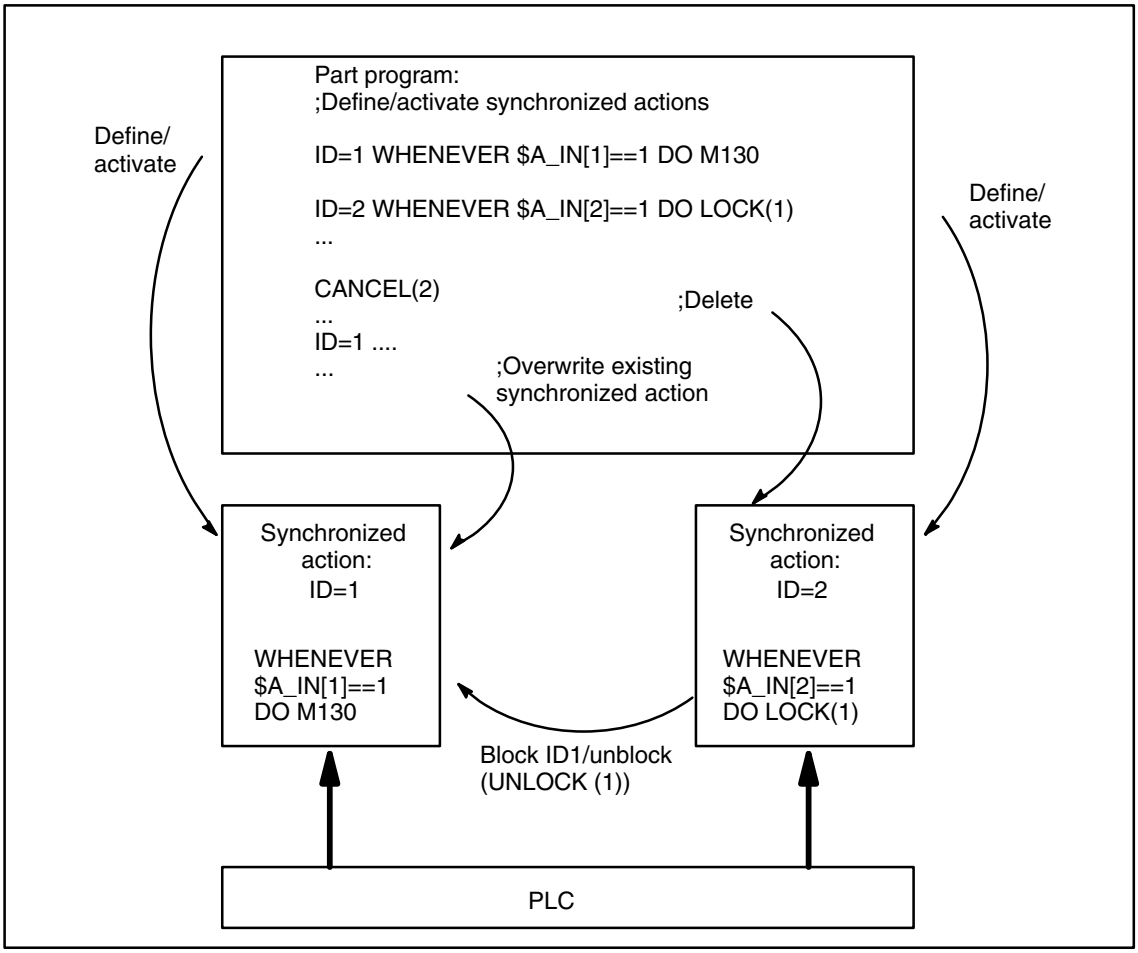


Fig. 2-10 Setting up/locking modal synchronized actions/deleting

2.6 Control and protection of synchronized actions

2.6.1 Control via PLC

Function

Modal synchronized actions (ID, IDS) can be locked or enabled from the PLC.

- Disabling of all modal synchronized actions
- Selective disabling of individual synchronized actions

Control scope

The PLC can control up to the first 64 modal synchronized actions by applying disables (ID, IDS 1–64). The synchronized actions, which **can be disabled** by the PLC are stored in a 64-bit array of the interface:
DB21–30, DBB308–315
and are tagged with a 1 by the NC. Protected synchronized actions are never tagged as being possible to disable. See Subsection 2.6.2.

Disable all synchronized actions

The PLC application program can set DB 21–30, DBB1 bit 2 to disable (lock against activation) all modal synchronized actions that are already defined in the NC and stored against activation. In this case, protected synchronized actions are an exception. Please see Subsection 2.6.2.
Setting DB 21–30, DBB1 bit 2 to 0 cancels the general disable by the PLC again.

Application of selective disabling

One bit is reserved for each of first 64 IDs (1–64) in the PLC interface (DB 21–30, DBB 300 bit 0 to DB21–30 DBB 307 bit 7).

These functions are enabled by default (bits = 0). When the allocated bit is set, evaluation of the condition and execution of the associated function are disabled in the NCK.

Cancellation of selective disabling

Setting the bits corresponding to the ID, IDS number to 0 in DB 21–30, DBB 300, bit 0 to DB 21–30, DBB 307 bit 7 causes the PLC to enable a previously disabled synchronized action.

Updating the selective disabling

If the PLC user program has made changes in the range DB 21–30, DBB 300 bit 0 to DB 21–30, DBB 307 bit 7, the changes must be activated with DB 21–30 DBX280.1.

Selective disabling status signal

If selective disabling was activated by the NCK, this is indicated in DB 21–30 DBX.281.1.

References: /LIS/, Lists, Interface Signals

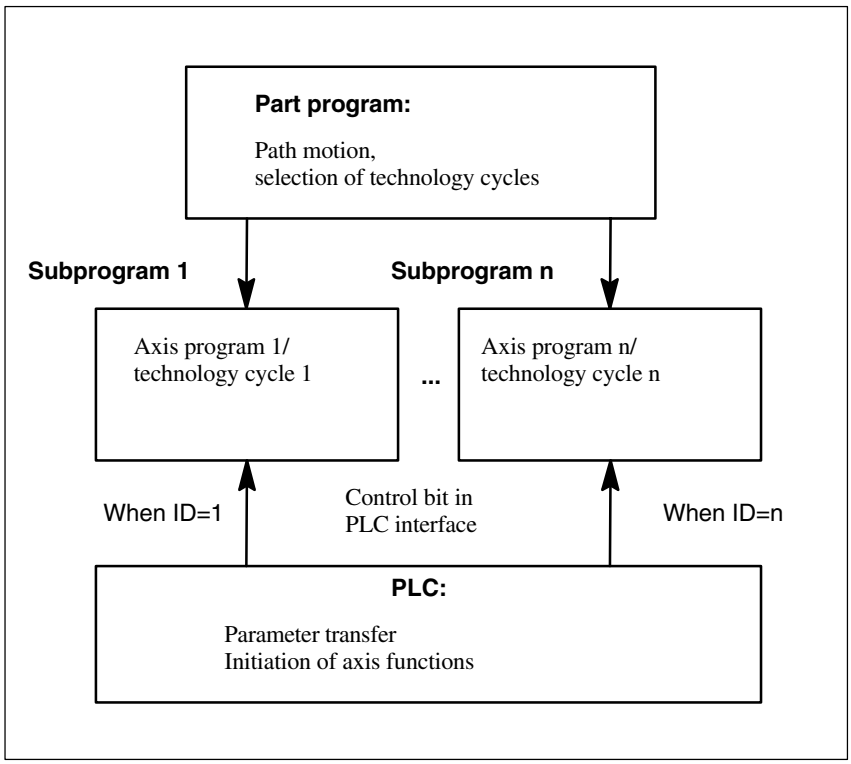


Fig. 2-11 Axis programs/technology cycles

Reading/writing of PLC data

In SW version 4 and later, PLC data can be read and written from the part program by transferring parameters between the NCK and PLC via the VDI interface.
This is an option: PLC variables

References: /FB/, P3, “Basic PLC Program”

Parameters can also be accessed from synchronized actions, thus allowing PLC data to be transferred to the NCK for parameterization before an axis function is initiated. The system variables to be addressed can be found in Subsection 2.3.11.

2.6.2 Protected synchronized actions

Global protection

Function

Machine data
MD 11500: PREVENT_SYNACT_LOCK
can be programmed to define an area of protected synchronized actions. Synchronized actions with ID numbers within the protected area can **no longer be**:

- Overwritten
- Deleted (CANCEL) or
- Disabled (LOCK)

once they have been defined. Protected synchronized actions cannot be disabled via the PLC either. They are indicated to the PLC as non-lockable in the interface. See Subsection 2.6.1.

Note

The functionality is also used for Safety Integrated systems.

Applications

The end customer must be prevented from modifying reactions to certain states defined by the machine manufacturer.

The machine is commissioned by the manufacturer without protection. This enables the gating logic to be defined and tested. However, the manufacturer declares the range of synchronized actions he has used as protected before the system is delivered to the end customer, thus preventing the end customer from defining his own synchronized actions within this protected area.

Notation of MD 11500

```
$MN_PREVENT_SYNACT_LOCK[0]= i  ; i Number of the first disable ID
$MN_PREVENT_SYNACT_LOCK[1]= j  ; j Number of the last disable ID
```

i and j can also be inverted.
If i = 0 and j = 0, no synchronized actions are protected.

Channel-specific protection

Function

The operator can use the channel-specific machine data
MD 21240 : PREVENT_SYNACT_LOCK_CHAN
to define an area of protected synchronized actions for the channel. Synchronized actions with ID numbers within the protected area can **no longer be**:

- Overwritten
- Deleted (CANCEL) or
- Disabled (LOCK)

once they have been defined. Protected synchronized actions cannot be disabled via the PLC either. They are indicated to the PLC as non-lockable in the interface. See Subsection 2.6.1.

Application

See above

Notation of MD 21240

```
CHANDATA(C)    ; With C channel number
$MC_PREVENT_SYNACT_LOCK_CHAN[0]= k
                ; k Number of the first ID to be disabled for the channel
$MC_PREVENT_SYNACT_LOCK_CHAN[1]= l
                ; l Number of the last ID to be disabled for the channel
```

k and l can also be inverted.
If k = 0 and l = 0, no synchronized actions are protected.

k = –1 and l = –1 indicates that the global area of protected synchronized actions programmed for the channel with MD 11500: PREVENT_SYNACT_LOCK applies.

Note

Protection for synchronized actions must be canceled while protected static synchronized actions are being defined, otherwise POWER ON will have to be executed for every alteration to allow redefinition of the logic.

The effect of the disable is identical, whether it is programmed as:
a global disable or a
channel-specific disable.

Example

In a system with 2 channels, synchronized actions should be protected as follows:
IDs 20 to 30 should be protected in the first channel and
IDs 25 to 35 in the second. Global and channel-specific protection may be mixed.

```
$MN_PREVENT_SYNACT_LOCK[0] = 25          ; Global protection

$MN_PREVENT_SYNACT_LOCK[1] = 35          ; Global protection

CHANDATA(1)

$MC_PREVENT_SYNACT_LOCK_CHAN[0] = 20
; Only the channel-specific MD (first ID number to be protected) is effective in
; the first channel

$MC_PREVENT_SYNACT_LOCK_CHAN1] = 30
; Only the channel-specific MD (last ID number to be protected) is effective in
; the first channel

CHANDATA(2)

$MC_PREVENT_SYNACT_LOCK_CHAN[0] = –1
                ; The global machine data
                ; $MN_PREVENT_SYNACT_LOCK is effective in the
                ; second channel!

$MC_PREVENT_SYNACT_LOCK_CHAN[1] = –1
...
```

2.7 Control system response for synchronized actions in specific operational states

2.7.1 POWER ON

No synchronized actions are active during POWER ON. Static synchronized actions that are required to be active immediately after POWER ON must be activated within an ASUB started by the PLC.

References: /FB/, P3, Basic PLC Program
/FB/, K1, Mode Group, Channel, Program Operation Mode

This arrangement can be used only on condition that SW 4 with ASUBs in all operating modes functionality is installed.

Examples:

- Adaptive control
- Safety Integrated, gating logic formulated by means of synchronized actions

2.7.2 RESET

Positioning axis motions

All positioning motions initiated from synchronized actions are aborted on NC reset. Active technology cycles are reset.

ID

Synchronized actions programmed locally (i.e. with ID=...) are deselected on NC reset.

IDS

Static synchronized actions (programmed with IDS = ...) remain active after NC reset. Motions can be restarted from static actions after NC reset.

Other reactions, dependent on actions

RESET continued

Synchronized action/ technology cycle	Modal and non-modal Active action is aborted, synchronized actions are canceled	Static (IDS) Active action is aborted, technology cycle is reset
Axis/positioning spindle	Motion is aborted	Motion is aborted
Speed-controlled spindle	$MA_SPIND_ACTIVE_AFTER_RESET== TRUE: Spindle remains active $MA_SPIND_ACTIVE_AFTER_RESET==FALSE: Spindle stops.	$MA_SPIND_ACTIVE_AFTER_RESET== TRUE: Spindle remains active $MA_SPIND_ACTIVE_AFTER_RESET==FALSE: Spindle stops.
Master value coupling	$MC_RESET_MODE_MASK, bit13 == 1: Master value coupling remains active $MC_RESET_MODE_MASK, bit13 == 0: Master value coupling is separated	$MC_RESET_MODE_MASK, bit13 == 1: Master value coupling remains active $MC_RESET_MODE_MASK, bit13 == 0: Master value coupling is separated
Measuring operations	Measuring operations started from synchronized actions are aborted	Measuring operations started from static synchronized actions are aborted

2.7.3 NC STOP

Motion start from static

Motions that have been started from static synchronized actions remain active in spite of an NC STOP.

Response of a command axis in SW 6.3 and later:

Note

In SW 6.3 and later, it is possible to convert a command axis started by a static synchronized action to a PLC-controlled axis. (VDI interface with “PLC controlling axis” IS (DB31, ... DBX28.7). This type of access is now stopped with an axial STOP **rather than** an NC STOP.

Motion start from non-modal and modal

Axis motions started from non-modal and modal actions are interrupted and then restarted by NC START. Speed-controlled spindles remain active.

Synchronized actions programmed in the current block remain active.
Example:
Set output: ... DO $A_OUT[1] = 1

2.7.4 Mode change

The response differs depending on whether the relevant synchronized action is static or programmed locally.
Synchronized actions activated by keyword **IDS** remain active after a change in operating mode. All other synchronized actions are deactivated in response to a mode change and reactivated on switchover to AUTO mode for repositioning.

Example:

```
N10    WHEN $A_IN[1] == 1 DO DELDTG
N20    G1      X10 Y 200 F150 POS[U]=350
```

Block N20 contains a STOP command. The operating mode is switched to JOG. If deletion of distance-to-go was not active prior to the interruption, then the synchronized action programmed in block N10 is reactivated when AUTO mode is selected again and the program continued.

2.7.5 End of program

Static synchronized actions remain active after the end of program. Modal and non-modal synchronized actions are aborted.
Static and modal synchronized actions programmed in M30 blocks remain active. They can be aborted with CANCEL before the M30 block. Polynomial coefficients programmed with FCTDEF remain active after the end of program.

2.7.6 Response of active synchronized actions to end of program and change in operating mode

See Subsections 2.7.4 and 2.7.5.

Synchronized action/ technology cycle	**Modal and non-modal actions** are aborted	**Static actions (IDS)** remain active
Axis/ positioning spindle	M30 is delayed until the axis/spindle is stationary.	Motion continues
Speed-controlled spindle	End of program: $MA_SPIND_ACTIVE_AFTER_RESET== TRUE: Spindle remains active $MA_SPIND_ACTIVE_AFTER_RESET==FALSE: Spindle stops The spindle remains active if the operating mode changes	Spindle remains active
Master value coupling	$MC_RESET_MODE_MASK, bit13 == 1: Master value coupling remains active $MC_RESET_MODE_MASK, bit13 == 0: Master value coupling is separated	A coupling started from a static synchronized action remains active
Measuring operations	Measuring operations started from synchronized actions are aborted	Measuring operations started from static synchronized actions remain active

2.7.7 Block search

General

Synchronized actions in the program, which have been interpreted during the block search, are collected, but their conditions are not evaluated. No actions are executed. Processing of synchronized actions does not commence until NC Start.

IDS

Synchronized actions that are programmed with keyword IDS and are already active remain operative during the block search.

Polynomial coefficients

Polynomial coefficients programmed with FCTDEF are collected **with calculation** during a block search, i.e. they are written to system variables.

2.7.8 Program interruption by ASUB

ASUB start

Modal and static motion-synchronous actions remain active and are also operative in the asynchronous subprogram (ASUB).

ASUB end

If the asynchronous subprogram is not continued with REPOS, then modal and static motion-synchronous actions modified in the subprogram remain operative in the main program.
Positioning motions started from synchronized actions respond in the same way as to operating mode switchover:
Motions started from non-modal and modal actions are stopped and continued with REPOS (if programmed). Motions started from static synchronized actions continue uninterrupted.

2.7.9 REPOS

In the remainder of the block, the synchronized actions are treated in the same way as in an interruption block.
Modifications to modal synchronized actions in the asynchronous subprogram are not effective in the interrupted program.
Polynomial coefficients programmed with FCTDEF are not affected by ASUB and REPOS.
The coefficients from the call program are applied in the asynchronous subprogram. The coefficients from the asynchronous subprogram continue to be applied in the call program.

If positioning motions started from synchronized actions are interrupted by the operating mode switchover or start of the interrupt routine, then they are continued in response to REPOS.

2.7.10 Response to alarms

Axis and spindle motions started by means of synchronized actions are decelerated in response to an alarm involving a motion stop instruction. All other actions (such as Set output) continue to be executed.

If an alarm is activated by a synchronized action, then the action is no longer processed in the next interpolation cycle, i.e. the alarm is output only once.
Alarms that respond with an interpreter stop only take effect once the precoded blocks have been processed.

Processing of all other actions continues as normal.
If a technology cycle generates an alarm with motion stop, then processing of the relevant cycle ceases.

2.8 Configuration

2.8.1 Configurability

Number of synchronized action elements

The number of programmable synchronized action blocks depends entirely on the configurable number of synchronized action elements. The number of storage elements for motion-synchronous actions (synchronized action elements) is defined in machine data

MD 28250: MM_NUM_SYNC_ELEMENTS.

This data can be set irrespective of the number of blocks available in the control system, thus enabling the complexity of expressions evaluated in real time as well as the number of actions to be set flexibly.

Use of elements

One synchronized action element is required for each of the following:

- A comparison expression in a condition
- An elementary action
- The synchronized action block

Example:
A total of four elements is needed for the synchronized action block below.

```
WHENEVER ($AA_IM[x] > 10.5) OR ($A_IN[1]==1) DO
|________| |__________________|   |________________|
Element 1        Element 2              Element 3

$AC_PARAM[0]=$AA_in[y]+1
|_____________________________|
Element 4
```

The default value of MD 28250: $MC_MM_NUM_SYNC_ELEMENTS is selected such that it is possible to activate the maximum presetting for SW 3 and earlier of 16 synchronized actions.

Note

If the user does not wish to program any synchronized actions, then he can reset the value in MD 28250: MM_NUM_SYNC_ELEMENTS to 0 so as to save approximately 16 KB of DRAM memory.

Display

The status display for synchronized actions (see Section 2.9) indicates how much of the memory provided for synchronized actions is still available. This status can also be read from synchronized actions in variable $AC_NUM_SYNC_ELEM.

Alarm

An alarm is generated if all available elements are used up during program execution. The user can respond by increasing the number of synchronized action elements or by modifying his program accordingly.

Number of FCTDEF functions

The number of programmable FCTDEF functions for each block can be configured via machine data
MD 28252: MM_NUM_FCTDEF_ELEMENTS.

The default value for all types of control is 3.
The control-specific maximum value can be found in

References: /LIS/, Lists.

Interpolation cycle

The time required on the interpolation level increases with the number of synchronized actions programmed. It may be necessary for the start-up engineer to lengthen the interpolation cycle accordingly.

Guide values for lengthening interpolation cycle

As a guide, individual times required to perform operations within synchronized actions (measured on an 840D with NCU 573.x) are given below: Times may be different for other control types.

NC language	**Time required**	
	Total	**Text in bold print**
Basic load for a synchronized action, if condition is not fulfilled: **WHENEVER FALSE** DO $AC_MARKER[0]=0	10 µs	**~10 µs**
Read variable: WHENEVER **$AA_IM[Y]>10** DO $AC_MARKER[0]=1	11 µs	**~1 µs**
Write variable: DO ***$R2=1***	11–12 µs	**~1–2 µs**
Read/write setting data: DO **$$SN_SW_CAM_MINUS_POS_TAB_1[0]=20**	24 µs	**~14 µs**
Basic arithmetic operations, e.g. multiplication: DO $R2=**$R2*2**	22 µs	**~12 µs**
Trigonometric functions (e.g. cos): DO $R2=**COS($R2)**	23 µs	**~13 µs**
Start positioning axis motion: WHEN TRUE DO **POS[z]=10**	83 µs	**~73 µs**

2.9 Diagnostics (with MMC 102/MMC 103 only)

Diagnostic functionality

The following special test tools are provided for diagnosing synchronized actions:

- Status display
- The current values of all synchronized action variables can be displayed (display real-time variables)
- Characteristics of variables can be recorded in the interpolation cycle grid (log real-time variables)

This functionality is structured in the operator interface in the following way:

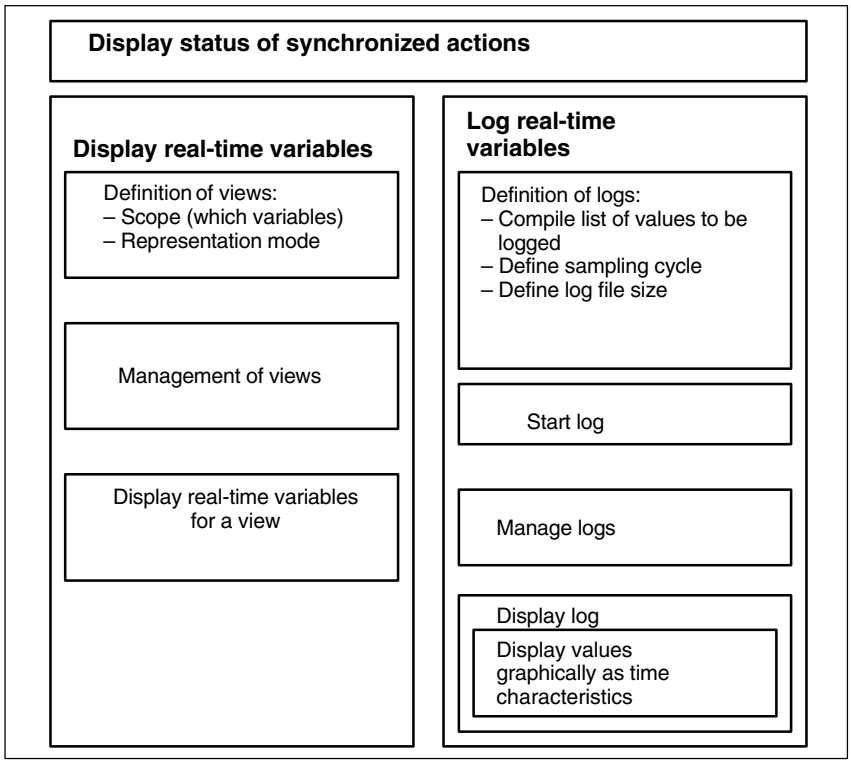


Fig. 2-12 Functionality of test tools for synchronized actions

For a description of how to use these functions, please see:

References: /BA/, Operator's Guide

2.9.1 Display status of synchronized actions

Status display

The status display contains the following information:

- Current extract of selected program

All programmed synchronized actions according to:

- Line number
- Code denoting synchronized action type
- ID number of synchronized action (for modal actions)
- Status

Synchronized action type

Synchronized actions are categorized as follows:

– ID Modal synchronized action

– IDS Static modal synchronized action

– Non-modal synchronized action for next executable block (in AUTOMATIC mode only)

Status

The following status conditions might be displayed:

- No status: The condition is checked in the interpolation cycle
- Disabled LOCK has been set for the synchronized action
- Active Action currently being executed If the action consists of a technology cycle, the current line number in the cycle is also displayed.

Complete synchronized actions

A search function can be used to display the originally programmed line in NC language for each displayed synchronized action.

2.9.2 Display real-time variables

System variables can be monitored for the purpose of monitoring synchronized actions. Variables, which may be used in this way are listed for selection by the user.
A complete list of individual system variables with ID code W for write access and R for read access for synchronized actions can be found in:

References: /PGA/, Programming Guide Advanced, Appendix

Views

“Views” are provided to allow the user to define the values, which are relevant for a specific machining situation and to determine how (in lines and columns, with what text) these values must be displayed. Several views can be arranged in groups and stored in correspondingly named files.

Managing views

A view defined by the user can be stored under a name of his choice and then called again. Variables included in a view can still be modified (Edit View).

Displaying real-time variables for a view

The values assigned to a view are displayed by calling the corresponding user-defined view.

2.9.3 Log real-time variables

Initial situation

To be able to trace events in synchronized actions, it is necessary to monitor the action status in the interpolation cycle.

Method

The values selected in a log definition are written to a log file of defined size in the specified cycle. Special functions for displaying the contents of log files are provided.

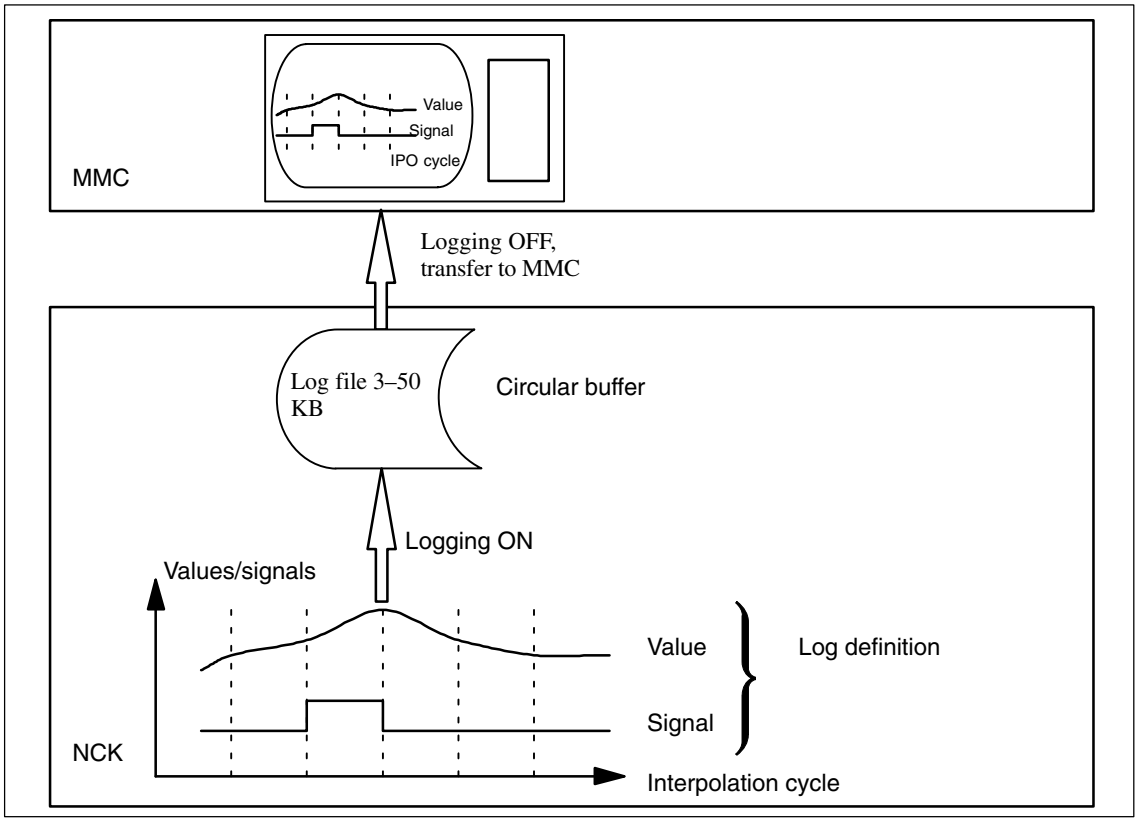


Fig. 2-13 Schematic representation of Log real-time variables process

Operation

For information about operating the logging function, please see:

References: /BA/, Operator's Guide

Log definition

The log definition can contain up to 6 specified variables. The values of these variables are written to the log file in the specified cycle. A list of variables, which may be selected for logging purposes, is displayed. The cycle can be selected in multiples of the interpolation cycle. The file size can be selected in KB. A log definition must be initialized before it can be activated on the NCK for the purpose of acquiring the necessary values.

Log file size

Values between 3 KB (minimum) and 50 KB (maximum) can be selected as the log file size.

Storage method

When the effective log file size has been exceeded, the oldest entries are overwritten, i.e. the file works on the circular buffer principle.

Starting logging

Logging according to one of the initialized log definitions is started by:

- Operation
- Setting system variable $A_PROTO=1 from the part program

The starting instant must be selected such that the variables to be logged are not altered until operations on the machine have been activated. The start point refers to the last log definition to be initialized.

Stopping logging

This function terminates the acquisition of log data in the NCK. The file containing the logged data is made available on the MMC for storage and evaluation (graphic log). Logging can be stopped by:

- Operation
- Setting system variable $A_PROTO=0 from the part program

Graphic log function

The measured values (up to 6) of a log are represented graphically as a function of the sampling time. The names of variables are specified in descending sequence according to the characteristics of their values. The screen display is arranged automatically. Selected areas of the graphic can be zoomed.

Note

Graphic log representations are also available as text files on the MMC 102. An editor can be used to read the exact values of a sampling instant (values with identical count index) numerically.

Managing logs

Several log definitions can be stored under names of the user's choice. They can be called later for initialization and start of recording or for modification and deletion.

■

3 Supplementary Conditions

Availability/scope of performance

The scope of performance provided by the “Synchronized actions” function package depends on the following:

- The type of SINUMERIK control system
  - HW
  - SW (export/standard versions)
- The availability of functions that can be initiated by “Actions”:
  - Standard functions
  - Functions that are available as options

The performance of control systems and their variants as well as functions supplied as options are described in catalogs specific to the SW version:

References: /BU/, Ordering Information, **Catalog NC60.1** and in /LIS/, Lists

The functions associated with synchronized actions are also dependent on:

- The list of system variables that can be read/written from synchronized actions including machine and setting data.
  - The number of available system variables depends on the SW version installed

System variables that may be used in conjunction with specific SW versions are described in:

References: /PGA/, Programming Guide Advanced, Appendix (for the relevant SW version)

Extensions in SW 4

The following extensions have been introduced with SW 4:

- Diagnostic facilities for synchronized actions
- Availability of additional real-time variables
- Complex conditions in synchronized actions
  - Basic arithmetic operations
  - Functions
  - Indexing with real-time variables
  - Access to setting and machine data
  - Logic operators
- Configurability
  - Number of simultaneously active synchronized actions
  - Number of special variables for synchronized actions
- Activate command axes/axis programs/technology cycles from synchronized actions

- PRESET from synchronized actions
- Couplings and coupled motions from synchronized actions
  - Switch on
  - Switch off
  - Parameterization
- Use of measuring functions from synchronized actions
- SW cams
  - Redefinition of position
  - Redefinition of lead times
- Deletion of distance-to-go without preprocessing stop
- Static synchronized actions (modes other than AUTO possible)
- Synchronized actions:
  - Protection against overwriting and deletion
  - Stopping, continuing, deleting
  - Resetting technology cycles
  - Parameterizing, enabling and disabling from PLC
- Overlaid movement/optimized clearance control
- Coordinating channels from synchronized actions
- Starting ASUBs from synchronized actions
- Non-modal auxiliary function outputs
- All necessary functions for Safety Integrated for formulation of requisite safety-oriented logic operations, protected against changes.
- 16 synchronized actions are included in the basic version

Extensions in SW 5

The following extensions have been introduced with SW 5:

- Synchronized actions, which can be tagged for the PLC
- Availability of additional real-time variables
- Access to PLC I/O (option)
- 255 parallel synchronized actions per channel are possible with the option “Synchronized actions step 2”.
- Static synchronized actions IDS that are active beyond the program end and are effective in all operating mode are possible using the option “Inter-mode group actions, ASUBs and synchronized actions”.

■

4 Data Descriptions (MD, SD)

4.1 General machine data

??? Classify MD number	**MM_MAINTENANCE_MON** Activate recording of maintenance data		
Default setting: 0	Minimum input limit: 0	Maximum input limit: 1	
Changes effective after POWER ON	Protection level: 2 / 7	Unit: –	
Data type: BOOLEAN	Applies from SW 6.5, 7.1		
Meaning:	This machine data is used to activate the recording of data for machine maintenance. Data to be recorded must be set with the axis-specific MD ... : MAINTENANCE_DATA.		

11500 MD number	**PREVENT_SYNACT_LOCK** Protected synchronized actions		
Default setting: 0, 0	Minimum input limit: 0	Maximum input limit: 255	
Changes effective after POWER ON	Protection level: 2 / 7	Unit: –	
Data type: DWORD	Applies from SW 4.1		
Meaning:	First and last ID of a protected synchronized action area. Synchronized actions with IDs within this area cannot be overwritten or disabled in the program (NC: CANCEL, LOCK). Neither can protected synchronized actions be disabled (LOCK) by the PLC. Typical application: The machine manufacturer defines safety logic in an asynchronous subprogram. This logic is started by the PLC during POWER ON. The range of IDs used is locked out via this machine data, thus preventing the end customer from modifying or deactivating the safety logic integrated by the machine manufacturer. Note: Protection for synchronized actions must be canceled while actions to be protected are being defined, otherwise POWER ON will have to be executed for every alteration to allow redefinition of the logic. A setting of 0.0 means that no synchronized actions are protected, i.e. the function is not switched on. The values are read as absolute values. Upper and lower values can be specified in any sequence. If necessary, the configuration can be reconfigured using the channel-specific MD 21240: PREVENT_SYNACT_LOCK_CHAN.		
Related to	MD 21240: PREVENT_SYNACT_LOCK_CHAN		

11510 MD number	**IPO_MAX_LOAD** Maximum permissible IPO load		
Default setting: 0	Minimum input limit: 0	Maximum input limit: 100	
Changes effective after POWER ON	Protection level: 2 / 7	Unit: %	

11510 MD number	**IPO_MAX_LOAD** Maximum permissible IPO load	
Data type:		Applies from SW
Meaning:	Activate capacity evaluation using synchronized actions. This MD can be used to specify the IPO computing time (as a % of the IPO cycle) at and above which the system variable $AN_IPO_LOAD_LIMIT should be set to TRUE. If, once the value has exceeded the limit, it falls back below it, the variable is reset to FALSE. If the machine data = 0, this diagnostic function is deactivated.	

4.2 Channelspecific machine data

21240 MD number	**PREVENT_SYNACT_LOCK_CHAN** Protected synchronized actions for channel		
Default setting: –1, –1	Minimum input limit: –1	Maximum input limit: 255	
Changes effective after POWER ON	Protection level: 2 / 7	Unit: –	
Data type: DWORD	Applies from SW 6.4		
Meaning:	First and last ID of a protected synchronized action area. Synchronized actions with IDs within this area cannot be overwritten or disabled in the program (NC: CANCEL, LOCK). Neither can protected synchronized actions be disabled (LOCK) by the PLC. The range of IDs used is locked out via this machine data, thus preventing the end customer from modifying or deactivating the safety logic integrated by the machine manufacturer. Note: Protection for synchronized actions must be canceled while actions to be protected are being defined, otherwise POWER ON will have to be executed for every alteration to allow redefinition of the logic. A setting of 0.0 means that no synchronized actions are protected, i.e. the function is not switched on. The values are read as absolute values. Upper and lower values can be specified in any sequence. –1, –1 indicates that the ID numbers programmed for the channel with MD 11500: PREVENT_SYNACT_LOCK are valid.		
Related to	MD 11500: PREVENT_SYNACT_LOCK		

28250 MD number	**MM_NUM_SYNC_ELEMENTS** Number of elements for expressions in synchronized actions		
Default setting: 159	Minimum input limit: 0	Maximum input limit: 2000	
Changes effective after POWER ON	Protection level: 2 / 7	Unit: –	
Data type: DWORD	Applies from SW 4.1		
Meaning:	The components of synchronized actions are stored in elements for storage in the control system. An action requires a minimum of 4 elements. The assignments are as follows: – Every operand in the condition uses 1 element – Every action >= 1 element – Every assignment 2 elements – Every additional operand in complex expressions 1 element. One element uses approximately 64 bytes of memory.		
References	Programming Guide Advanced		

28252 MD number	**MM_NUM_FCTDEF_ELEMENTS** Number of FCTDEF elements		
Default setting: 3	Minimum input limit: 0	Maximum input limit: 100	
Changes effective after POWER ON	Protection level: 2 / 7	Unit: –	
Data type: DWORD	Applies from SW 4.1		
Meaning:	Storage elements are required to store functions in the control system for use by synchronized actions. This MD determines the number of these elements.		

28254 MD number	**MM_NUM_AC_PARAM** Number of $AC_PARAM parameters	
Default setting: 50	Minimum input limit: 0	Maximum input limit: 10000, SW 6.3 and later: 20000
Changes effective after POWER ON	Protection level: 2 / 7	Unit: –
Data type: DWORD	Applies from SW 4.1	
Meaning:	Number of channel-specific $AC_PARAM parameters for synchronized actions	

28255 MD number	**MM_BUFFERED_AC_PARAM** Storage location for $AC_PARAM	
Default setting: 0	Minimum input limit: 0	Maximum input limit: 1
Changes effective after POWER ON	Protection level: 2 / 7	Unit: –
Data type: DWORD	Applies from SW 6.3	
Meaning:	The $AC_PARAM system variables can be saved either: 0: In dynamic DRAM (default) 1: In static SRAM System variables saved in SRAM retain their current values after RESET and POWER ON. They can be included in the data backup.	
Related to	MM_NUM_AC_PARAM	
References	/IAD/, Installation and StartUp Guide	

28256 MD number	**MM_NUM_AC_MARKER** Number of $AC_MARKER markers	
Default setting: 8	Minimum input limit: 0	Maximum input limit: 10000, SW 6.3 and later: 20000
Change effective after POWER ON	Protection level: 2 / 7	Unit: –
Data type: DWORD	Applies from SW 4.1	
Meaning:	Number of channel-specific $AC_MARKER markers for synchronized actions	

28257 MD number	**MM_BUFFERED_AC_MARKER** Storage location for $AC_MARKER	
Default setting: 0	Minimum input limit: 0	Maximum input limit: 1
Changes effective after POWER ON	Protection level: 2 / 7	Unit: –
Data type:	Applies from SW 6.3	
Meaning:	You can save the system variables $AC_MARKER either: 0: In dynamic DRAM (default) 1: In static SRAM System variables saved in SRAM retain their current values after RESET and POWER ON. They can be included in the data backup.	
Related to	MM_NUM_MARKER	
References	/IAD/, Installation and Start-Up Guide	

28258 MD number	**MM_NUM_AC_TIMER** Number of $AC_TIMER time variables		
Default setting: 0	Minimum input limit: 0	Maximum input limit: 10000	
Changes effective after POWER ON	Protection level: 2 / 7	Unit: –	
Data type: DWORD	Applies from SW 4.1		
Meaning:	Number of channel-specific $AC_TIMER time variables for synchronized actions		

28260 MD number	**NUM_AC_FIFO** Number of $AC_FIFO1, $AC_FIFO2, ... variables	
Default setting: 0	Minimum input limit: 0	Maximum input limit: 10
Changes effective after POWER ON	Protection level: 2 / /	Unit: –
Data type: DWORD	Applies from SW 4.1	
Meaning:	Number of FIFO variables, $AC_FIFO1 to $AC_FIFO10, for synchronized actions.	
Application example(s)	FIFO variables can be used, for example, to track products: Information (e.g. product length) can be buffered for each part on a conveyor belt in a separate FIFO variable.	
Related to	MD 28262: START_AC_FIFO	

28262 MD number	**START_AC_FIFO** Store FIFO variables from R parameter	
Default setting: 0	Minimum input limit: 0	Maximum input limit: 10000
Changes effective after POWER ON	Protection level: 2 / 7	Unit: –
Data type: DWORD	Applies from SW 4.1	
Meaning:	Number of R parameter at start of FIFO variable storage area. All R parameters with low numbers can be used as required in the part program. R parameters above the FIFO range cannot be written from the part program. The number of R parameters must be set in machine data MD 28050: $MC_MM_NUM_R_PARAM such that there is space to store all FIFO variables from the R parameter at the start of the FIFO area: $MC_MM_NUM_R_PARAM=$MC_START_FIFO + $MC_NUM_AC_FIFO*($MC_LEN_AC_FIFO+6) The FIFO variable names are $AC_FIFO1 to $AC_FIFOn. They have been set up as fields. Indices 0 – 5 have special meanings: n= 0: When a variable is written with index 0, a new value is stored in the FIFO When a variable is read with index 0, the oldest element is deleted from the FIFO n=1: Access to first element to be read in n=2: Access to last element to be read in n=3: Sum of all FIFO elements n=4: Number of elements available in FIFO n=5: Current write index relative to beginning of FIFO	
Related to	MD 28260: NUM_AC_FIFO	

28264 MD number	**LEN_AC_FIFO** Length of $AC_FIFO ... FIFO variables	
Default setting: 0	Minimum input limit: 0	Maximum input limit: 10000
Changes effective after POWER ON	Protection level: 2 / 7	Unit: –
Data type: DWORD	Applies from SW 4.1	
Meaning:	Length of FIFO variables $AC_FIFO1 to $AC_FIFO10. All FIFO variables in one channel are the same length.	
Related to	MD 28262, MD 28260	

28266 MD number	**MODE_AC_FIFO** FIFO processing mode		
Default setting: 0	Minimum input limit: 0	Maximum input limit: ***	
Changes effective after POWER ON	Protection level: 2 / 7	Unit: –	
Data type: BYTE	Applies from SW 4.1		
Meaning:	FIFO processing mode: Bit 0 = 1: The sum of all FIFO contents is regenerated on every write access operation. Bit 0 = 0: No summation		
Related to	MD 28260: NUM_AC_FIFO		

4.3 Axis/spindlespecific machine data

??? Classify! MD number	**MAINTENANCE_DATA** Setting for maintenance data to be recorded		
Default setting: 1	Minimum input limit: 0	Maximum input limit: 3	
Changes effective after RESET	Protection level: 2 / 7	Unit: –	
Data type:	Applies from SW		
Meaning:	This machine data is used to configure the recording of data for machine maintenance. It enables data to be recorded to be selected on an axis-specific basis. This can be done as follows: Bit 0: Total travel distance, total travel time and travel count Bit 1: Total travel distance, total travel time and travel count at high axis speeds Bit 2: Total jerk, travel time and travel count with jerk Bits 3 to 15 are reserved.		

30450 MD number	**IS_CONCURRENT_POS_AX** Competing positioning axis		
Default setting: 0	Minimum input limit: 0	Maximum input limit: 1	
Changes effective after POWER ON	Protection level: 2 / 7	Unit: 1	
Data type: Boolean	Applies from SW 1		
Meaning:	This axis is a competing positioning axis. From SW4.3 (not FM-NC): If FALSE: At RESET a neutral axis becomes channel axis again. If TRUE: At RESET a neutral axis remains in the neutral axis state, and a channel axis becomes neutral axis.		
References	Starting the command axes see Subsection 2.4.12		

32070 MD number	**CORR_VELO** Axis speed for handwheel, ext. ZO, cont. dressing, clearance control		
Default setting: 100	Minimum input limit: 0	Maximum input limit: Plus	
Changes effective after POWER ON	Protection level: 2 / 7	Unit: %	
Data type: DWORD	Applies from SW 3.2		
Meaning:	Limitation of axis velocity for handwheel override, external zero offset, continuous dressing, clearance control $AA_OFF via synchronized actions referenced to JOG velocity MD: JOG_VELO, MD: JOG_VELO_RAPID, MD: JOG_REV_VELO, MD: JOG_REV_VELO_RAPID. The maximum permissible velocity corresponds to the maximum velocity setting in MD: MAX_AX_VELO. The limitation is applied at this value. An alarm is generated if this maximum setting is exceeded. Conversion to linear or rotary axis velocity is carried out in accordance with MD: IS_ROT_AX.		
Application example(s)	Limitation of velocity for traversal of overlaid movements.		

32074 MD number	FRAME_OR_CORRPOS_NOTALLOWED Effectiveness of frames and tool length compensation		
Default setting: 0	Minimum input limit: 0	Maximum input limit: 0xFF	
Changes effective after POWER ON	Protection level: 2 / 7	Unit: –	
Data type: DWORD	Applies from SW 4.2		
Meaning:	The effectiveness of frames and tool length compensations with respect to indexing axes, PLC axes and command axes started from synchronized actions is programmed in this machine data. Bit == 0: Frame or compensation values are permitted Bit assignment: Bit 0 == 1: Programmable zero offset (TRANS) prohibited for indexing axis. Bit 1 == 1: Scale modification (SCALE) prohibited for indexing axis Bit 2 == 1: Reversal of direction of movement (MIRROR) prohibited for indexing axis Bit 3 == 1: DRF offset prohibited for axis Bit 4 == 1: Zero offset external prohibited for axis Bit 5 == 1: Online tool offset prohibited for axis Bit 6 == 1: Synchronized action offset prohibited for axis Bit 7 == 1: Compile cycles offset prohibited for axis Bit 8 == 1: Axial frames are taken into account for PLC axes Bit 8 == 0: Axial frames are NOT taken into account for PLC axes (bit evaluation for reasons of compatibility) Bit 9 == 1: Axial frames are NOT taken into account for command axes Bit 9 == 0: Axial frames are taken into account for command axes		

32920 MD number	AC_FILTER_TIME Filter smoothing constant for adaptive control		
Default setting: 0.0	Minimum input limit: 0.0	Maximum input limit: Plus	
Changes effective after POWER ON	Protection level: 2/7	Unit: s	
Data type: DOUBLE	Applies from SW 2.1		
Meaning:	The following actual drive values can be acquired by means of main run variables $AA_LOAD, $AA_POWER, $AA_TORQUE and $AA_CURR: – Drive capacity utilization – Drive active power – Drive torque setpoint – Current actual value of axis or spindle To compensate for peaks, the measured values can be smoothed using a PT1 filter. The filter time constant is defined in MD: AC_FILTER_TIME (filter smoothing time constant for Adaptive Control). The PT1 filter acts in addition to the filters integrated in the 611–D with respect to the drive torque setpoint or actual current value. The two filters are connected in series if both heavily and weakly smoothed values are required in the system. An input of a 0 second smoothing time deactivates the filter.		
MD irrelevant for	FM-NC with 611A		
Application example(s)	Smoothing of actual current value for AC Control.		

36750 MD number	AA_OFF_MODE Effect of value assignment for axial override with synchronized actions		
Default setting: 0	Minimum input limit: 0	Maximum input limit: 7	
Changes effective after POWER ON	Protection level: 2/7	Unit: –	
Data type: BYTE	Applies from SW 3.2 (bits 1 and 2 in SW 6 and later)		
Meaning:	Main run variable $AA_OFF allows an overlaid movement for the programmed axis to be implemented within a synchronized action. In axial MD: AA_OFF_MODE, the type of application is defined as follows: Bit0: Effect of tool assignment within a synchronized variable: In software version 3.2 and later Bit0 = 0: Absolute value Bit0 = 1: Incremental value (integrator) Bit1: Response of $AA_OFF in the case of a reset Bit1 = 0: $AA_OFF is deselected in the case of a reset Bit1 = 1: $AA_OFF is retained beyond the reset (SW 6 and later) Bit2: $AA_OFF in JOG mode Bit2 = 0: No overlaid movement on the basis of $AA_OFF Bit2 = 1: Overlaid movement is interpolated on the basis of $AA_OFF (SW 6 and later)		
Application example(s)	• Clearance control for laser machining (integral) • Joystick-controlled axis traversal (proportional)		

4.4 Setting data

43350 MD number	AA_OFF_LIMIT Upper limit of compensation value for $AA_OFF clearance control		
Default setting: 1.0 Ex+8	Minimum input limit: 0	Maximum input limit: ***	
Changes effective IMMEDIATELY	Protection level: 2 / 7	Unit: mm/degrees	
Data type: DOUBLE	Applies from SW 4.2		
Meaning:	Upper limit of compensation value that can be preset from synchronized actions by means of variable $AA_OFF. The limit value is applied to the effective absolute amount of compensation. Application for clearance control in laser machining operations: The compensation value is limited to prevent the laser head from becoming trapped in metal blanks. System variable $AA_OFF_LIMIT can be scanned to determine whether the compensation value is within the limit range.		

■

Notes

5 Signal Descriptions

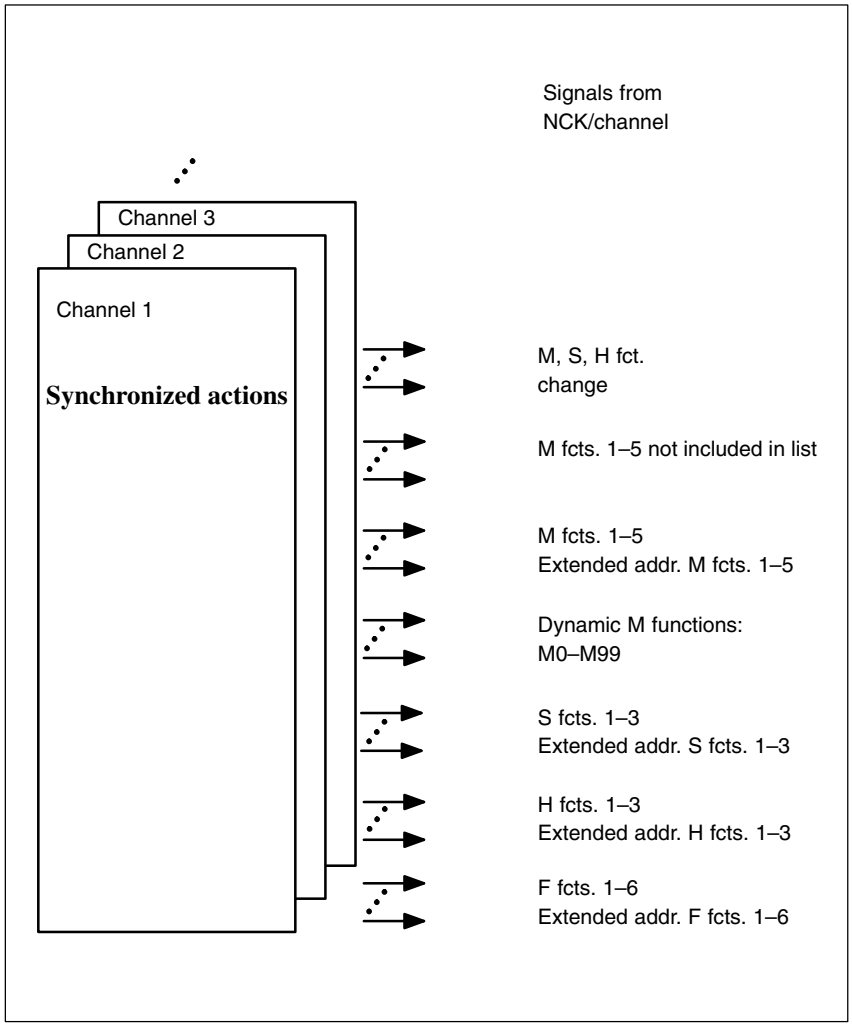


Fig. 5-1 PLC interface signals for synchronized actions

The signals generated as synchronized actions using auxiliary function output correspond to the signals described in

References: /FB/, H2, Output of Auxiliary Functions to PLC.

Signals to channel

The PLC user program uses signals
DB 21 – 30 DBB 300 bit 0 to
DB 21 – 30 DBB 307 bit 7
to request that the assigned synchronized actions be disabled, where:
DBB 300 bit 0 corresponds to the first modal synchronized action (ID=1/IDS=1) and DBB 307 bit 7 to the 64th modal synchronized action (ID=64/IDS=64).

Note

Only the instance (NCK or PLC), which initiated a disable can cancel the disable again.

Signals from channel

The channel uses signals DB 21 – 30 DBB 308 bit 0 to
DB 21 – 30 DBB 315 bit 7
to indicate to the PLC user program which synchronized actions can be disabled by the PLC, where:
DBB 308 bit 0 corresponds to the first modal synchronized action (ID=1/IDS=1) and DBB 315 bit 7 to the 64th modal synchronized action (ID=64/IDS=64).

Disable all synchronized actions

The global signal

DB21–30 DBB1 bit 2

disables **all** modal/static synchronized actions, which are not protected.

Disable selected synchronized actions

DB 21 – 30 DBX280.1
From the synchronized actions that are marked as possible to disable in DB 21 – 30 DBB 308 bit 0 to DB 21 – 30 DBB 315 bit 7, disable the ones in DB 21 – 30 DBB 300 bit 0 to DB 21 – 30 DBB 307 bit 7 that have been marked to be disabled by means of the set bit.

Synchronized actions disabled

DB 21 – 30 DBX281.1
The NCK confirms that the requested synchronized actions have been disabled.

■

6 Examples

6.1 Examples of conditions in synchronized actions

Path distance from end of block

Axial distance from block end: 10 mm or less (workpiece coordinate system):
... WHEN $AC_DTEW <= 10 DO ...
G1 X10 Y20

Axis distance from end of path

... WHEN $AA_DTEW[X]<= 10 DO ...
POS[X]= 10

Path distance from start of block

Path 20 mm or more after start of block in basic coordinate system:
...WHEN $AC_PLTBB >= 20 DO ...

Condition with function in comparison

Actual value for axis Y in MCS greater than 10 x sine of value in R10:
... WHEN $AA_IM[y] > 10*SIN(R10) DO ...

Step-by-step positioning

Every time input 1 is set, the axis position is advanced by one step. The input must be reset again to allow cold restarting of the system.
G91
EVERY $A_IN[1]==1 DO POS[X]= 10

OVR in every interpolation cycle

In order to selectively disable a path motion until a programmed signal arrives, $AC_OVR must be set to zero in every interpolation cycle (keyword WHENEVER).
WHENEVER $A_IN[1]==0 DO $AC_OVR= 0

Other options

The list of system variables that can be read in synchronized actions, which appears in

References: /PGA/, Programming Guide, Advanced and Subsection 2.3.11.

describes the full range of quantities that can be evaluated in the conditions of synchronized actions.

6.2 Reading and writing of SD/MD from synchronized actions

Infeed and oscillation for grinding operations

Setting data whose values remain unchanged during machining are addressed in the part program by their usual names.

Example: Oscillation from synchronized actions

NC language	Remarks
N610 ID=1 WHENEVER $AA_IM[Z]>**$SA_OSCILL_REVERSE_POS1[Z]**	
DO $AC_MARKER[1]=0	
	;Whenever the current position of the reciprocating
	;axis in the machine coordinate system is
	;less than the start of reversal area 2,
	; set the axial override of the
	; infeed axis to 0
N620 ID=2 WHENEVER $AA_IM[Z]<**$SA_OSCILL_REVERSE_POS2[Z]**-6	
DO $AA_OVR[X]=0 $AC_MARKER[0]=0	
	;Whenever the current position of the reciprocating
	;axis in the machine coordinate system
	;is the same as reverse position 1,
	; set the axial override of the
	; reciprocating axis to 0
	;and set the axial override of the
	; infeed axis to 100% (canceling the
	; previous synchronized action!)
N630 ID=3 WHENEVER $AA_IM[Z]==**$SA_OSCILL_REVERSE_POS1[Z]**	
DO $AA_OVR[Z]=0 $AA_OVR[X]=100	
	;Whenever the distance-to-go of the part infeed is
	;equal to 0,
	;set the axial override of the reciprocating
	; axis to 100% (canceling the previous
	; synchronized action!)
N640 ID=4 WHENEVER $AA_DTEPW[X]==0	
DO $AA_OVR[Z]=100 $AC_MARKER[0]=1 $AC_MARKER[1]=1	
N650 ID=5 WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0	
N660 ID=6 WHENEVER $AC_MARKER[1]==1 DO $AA_OVR[X]=0	
	;Whenever the current position of the reciprocating
	;axis in the workpiece coordinate system
	; is the same as reverse position 1,
	; set the axial override of the
	; reciprocating axis to 100%
	;and set the axial override of the
	; infeed axis to 0 (canceling the
	; second synchronized action once!)

N670 ID=7 WHEN $AA_IM[Z]==**$SA_OSCILL_REVERSE_POS1**[Z]
DO $AA_OVR[Z]=100 $AA_OVR[X]=0

Setting data whose value may change during machining (e.g. through an operator input or synchronized action) must be programmed with **$$S**...:

Example: Oscillation from synchronized actions with alteration of oscillation position via operator interface

N610 ID=1 WHENEVER $AA_IM[Z]>**$$SA_OSCILL_REVERSE_POS1[Z]** DO $AC_MARKER[1]=0

;Whenever the current position of the reciprocating
;axis in the machine coordinate system is
; less than the start of reversal area 2,
; set the axial override of the
; infeed axis to 0

N620 ID=2 WHENEVER $AA_IM[Z]<**$$SA_OSCILL_REVERSE_POS2[Z]**-6
DO $AA_OVR[X]=0 $AC_MARKER[0]=0

;Whenever the current position of the reciprocating
;axis in the machine coordinate system
;is the same as reverse position 1,
; set the axial override of the
; reciprocating axis to 0
;and set the axial override of the
; infeed axis to 100% (canceling the
; previous synchronized action)

N630 ID=3 WHENEVER $AA_IM[Z]==**$$SA_OSCILL_REVERSE_POS1[Z]**
DO $AA_OVR[Z]=0 $AA_OVR[X]=100

;Whenever the distance-to-go of the part infeed is
;equal to 0,
;set the axial override of the reciprocating
; axis to 100% (canceling the previous
; synchronized action)

N640 ID=4 WHENEVER $AA_DTEPW[X]==0
DO $AA_OVR[Z]=100 $AC_MARKER[0]=1 $AC_MARKER[1]=1
N650 ID=5 WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0
N660 ID=6 WHENEVER $AC_MARKER[1]==1 DO $AA_OVR[X]=0

;Whenever
; the current position of the reciprocating
; axis in the workpiece coordinate system
; is the same as reverse position 1,
; set the axial override of the
; reciprocating axis to 100%
;and set the axial override of the
; infeed axis to 0 (canceling the
; second synchronized action
; once)

N670 ID=7 WHEN $AA_IM[Z]==**$$SA_OSCILL_REVERSE_POS1[Z]**
DO $AA_OVR[Z]=100 $AA_OVR[X]=0

6.3 Examples of adaptive control

General procedure

The following examples use the polynomial evaluation function SYNFCT().

1. Representation of relationship between input value and output value (real-time variables in each case)
2. Definition of this relationship as polynomial with limitations
3. With position offset: Setting of MD and SD
   – MD 36750: $AA_OFF_MODE
   – SA 43350: $SA_AA_OFF_LIMIT (optional)
4. Activation of the control in a synchronized action

6.3.1 Clearance control with variable upper limit

Example of polynomial with dyn. upper limit

For the purpose of clearance control, the upper limit of the output ($AA_OFF, override value in axis V) is varied as a function of the spindle override (analog input 1). The upper limit for polynomial 1 is varied dynamically as a function of analog input 2.
Polynomial 1 is defined directly via system variables:

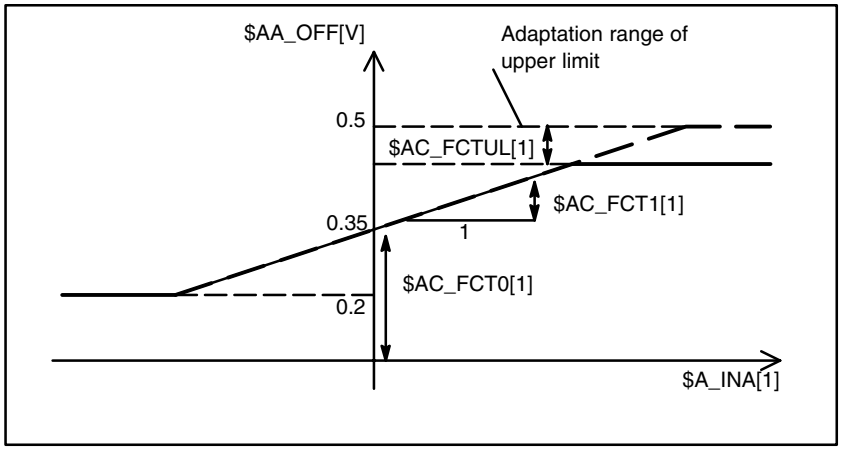


Fig. 6-1 Clearance control with variable upper limit

```
$AC_FCTLL[1]=0.2                    ; Lower limit
$AC_FCTUL[1]=0.5                    ; Init. value upper limit
$AC_FCT0[1]=0.35                    ; Zero crossover a0
$AC_FCT1[1]=1.5 EX-5                ; Lead a1
STOPRE                              ; See note below
...
ID=1 DO $AC_FCTUL[1]=$A_INA[2]*0.1+0.35    ;Adapt upper limit
                                    ; dynamically using
                                    ; analog input 2, no condition
ID=2 DOSYNFCT(1, $AA_OFF[V], $A_INA[1])
                                    ; Clearance control using override,
                                    ; no condition
...
```

Note

When system variables are applied in the part program, STOPRE must be programmed to ensure block-synchronous writing. The following is an equivalent notation for polynomial definition:
FCTDEF(1, 0.2, 0.5, 0.35, 1.5EX-5).

6.3.2 Feedrate control

Example of adaptive control with an analog input voltage

A process quantity (measured via $A_INA[1]) must be regulated to 2 V through an additive control factor implemented by a path (or axial) feedrate override. Feedrate override shall be performed within the range of ±100 [mm/min].

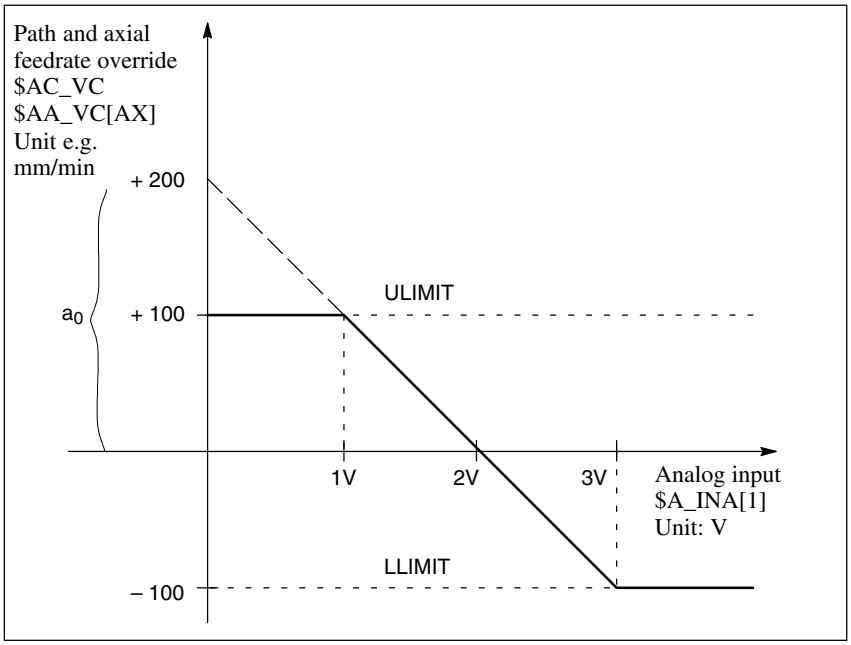


Fig. 6-2 Diagram illustrating adaptive control

Determination of coefficients:

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

$a_1 = -\frac{100\ mm}{1\ min \cdot 1\ V}$

$a_1 = -100 \Rightarrow$ control constant, lead

$a_0 = -(-100) \cdot 2 = 200$

$a_2 = 0$ (not a square component)

$a_3 = 0$ (not a cubic component)

Upper limit = 100

Lower limit = –100

```
FCTDEF(     Polynomial no.
            LLIMIT
            ULIMIT
            a0,             ; y for x = 0
            a1,             ; Lead
            a2,             ; Square component
            a3 )            ; Cubic component
```

With the values determined above, the polynomial is defined as follows:

```
FCTDEF(1, –100, 100, 200, –100, 0, 0)
```

The following synchronized actions can be used to activate the adaptive control function for the axis feedrate:

```
ID = 1 DO SYNFCT(1, $AA_VC[X], $A_INA[1])
```

or for the path feedrate:

```
ID = 2 DO SYNFCT(1, $AC_VC, $A_INA[1])
```

6.3.3 Control velocity as a function of normalized path

Multiplicative adaptation

The normalized path is applied as an input quantity: $AC_PATHN.
0: At block beginning
1: At block end
Variation quantity $AC_OVR must be controlled as a function of $AC_PATHN according to a 3rd order polynomial. The override must be reduced from 100 to 1% during the motion.

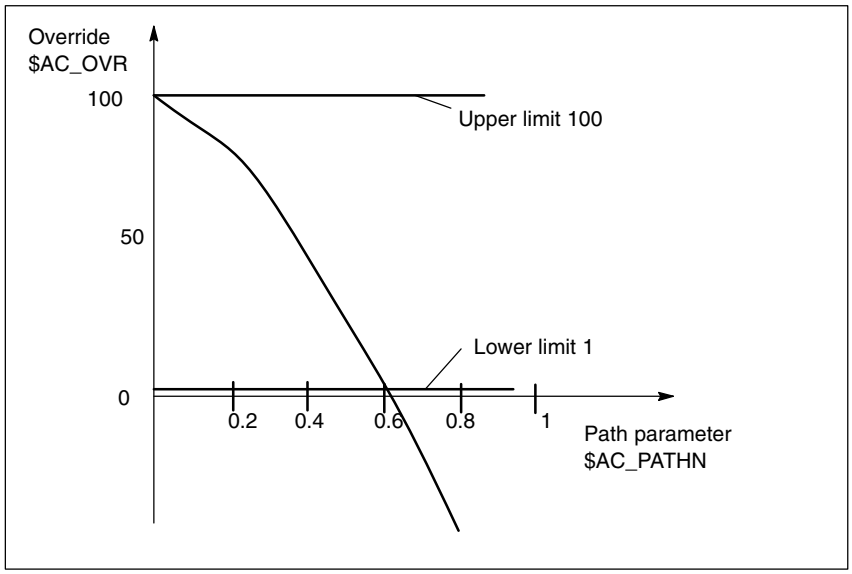


Fig. 6-3 Regulate velocity continuously

Polynomial 2:
Lower limit: 1
Upper limit: 100
a_0: 100
a_1: –100
a_2: –100
a_3: Reserved

With these values, the polynomial definition is as follows:

```
FCTDEF(2, 1, 100, 100, –100, –100)
; Activation of variable override as a function of path:

ID= 1 DO SYNFCT(2, $AC_OVR, $AC_PATHN)
G01 X100 Y100 F1000
```

6.4 Monitoring of a safety clearance between two axes

Task

Axes X1 and X2 operate two independently controlled transport devices used to load and unload workpieces.
To prevent the axes from colliding, a safety clearance must be maintained between them.
If the safety clearance is violated, then axis X2 is decelerated. This interlock is applied until axis X1 leaves the safety clearance area again.
If axis X1 continues to move towards axis X2, thereby crossing a closer safety barrier, then it is traversed into a safe position.

NC language	**Remarks**
ID=1 WHENEVER $AA_IM[X2] – $AA_IM[X1] < 30 DO $AA_OVR[X2]=0	; Safety barrier
ID=2 EVERY $AA_IM[X2] – $AA_IM[X1] < 15 DO POS[X1]=0	; Safe position

6.5 Store execution times in R parameters

Task

Store the execution time for part program blocks starting at R parameter 10.

Program	**Remarks**
	; The example is as follows **without** symbolic ; programming:
IDS=1 EVERY $AC_TIMEC==0 DO $AC_MARKER[0] = $AC_MARKER[0] + 1	
	; Advance R parameter pointer on block change
IDS=2 DO $R[10+$AC_MARKER[0]] = $AC_TIME	
	; Write current time of block start in each case to R ; parameter
	; The example is as follows **with** symbolic ; programming:
DEFINE INDEX AS $AC_MARKER[0]	; Agreements for symbolic programming
IDS=1 EVERY $AC_TIMEC==0 DO INDEX = INDEX + 1	; Advance R parameter pointer on block change
IDS=2 DO $R[10+INDEX] = $AC_TIME	
	; Write current time of block start in each case ; to R parameter

6.6 “Centering” with continuous measurement

Introduction

The gaps between gear teeth are measured sequentially. The gap dimension is calculated from the sum of all gaps and the number of teeth. The center position sought for continuation of machining is the position of the first measuring point plus 1/2 the average gap size. The speed for measurement is selected in order to enable one measured value to be reliably acquired in each interpolation cycle.

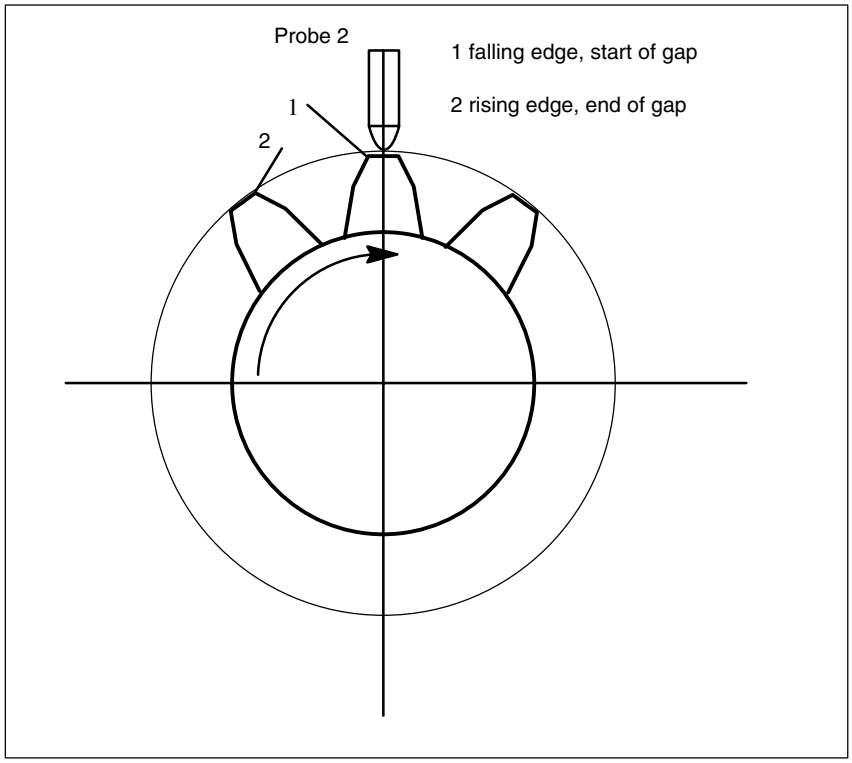


Fig. 6-4 Diagrammatic representation of measurement of gaps between gear teeth

```
%_N_MEAC_MITTEN_MPF

;Measure using rotary axis B (BACH) with display of difference between measured values


;****** Define local user variables ***
N1 DEF INT TOOTH COUNT              ; Enter number of gear wheel teeth
N5 DEF REAL HYS_POS_EDGE            ; Hysteresis positive edge probe
N6 DEF REAL HYS_NEG_EDGE            ; Hysteresis negative edge probe

********* Define code name for synchronization marker ***********
define M_TEETH as $AC_MARKER[1]     ; ID marker for computing: Neg/pos edge per tooth
define Z_MW as $AC_MARKER[2]        ; ID counter MW Read out FIFO
define Z_RW as $AC_MARKER[3]        ; ID counter MW Calculate tooth gaps
```

```
;****** Input values for GEAR WHEEL MEASUREMENT *******
N50 NO. OF TEETH=26                    ; Enter number of gear wheel teeth to be measured
N70 HYS_POS_EDGE = 0.160               ; Hysteresis positive edge probe
N80 HYS_NEG_EDGE = 0.140               ; Hysteresis negative edge probe

start:                                 ;******* Assign variables ************************
R1=0                                   ; ID2 Gap dimension calculated
R2=0                                   ; ID2 Total of all gaps
R3=0                                   ; Content of the first element read in
R4=0                                   ; R4 Corresponds to a tooth clearance
R5=0                                   ; Gap position calculated, end result
R6=1                                   ; ID 3 Activate BACH with MOV
R7=1                                   ; ID 5 Activate MEAC
M_TEETH=NO_OF_TEETH*2                  ; ID Calculate neg./pos. edge per tooth
Z_MW=0                                 ; ID counter MW Read out FIFO up to tooth count
Z_RW=2                                 ; ID counter Calculate tooth gap difference
R13=HYS_POS_EDGE                       ; Hysteresis in calc. register
R14=HYS_NEG_EDGE                       ; Hysteresis in calc. register

;******* Move, measure, calculate axis **********
N100 MEAC[BACH]=(0)                    ; Reset measurement job
;Reset FIFO1[4] variables and ensure a defined measurement trace
N105 $AC_FIFO1[4]=0                    ; Reset FIFO1
STOPRE

****** Read out FIFO until no. of teeth reached *****
; If FIFO1 is not yet empty and not all teeth have been measured, relocate measured value from FIFO variable
; to synchronized action parameter and increase measured value counter

ID=1 WHENEVER ($AC_FIFO1[4]>=1) AND (Z_MW<M_TEETH)
        DO $AC_PARAM[0+Z_MW]=$AC_FIFO1[0] Z_MW=Z_MW+1


; If 2 measured values are available, start to calculate. Calculate gap dimension ONLY and total gap
; increment calculated value counter by 2

ID=2 WHENEVER (Z_MW>=Z_RW) AND (Z_RW<M_TEETH)
        DO $R1=($AC_PARAM[–1+Z_RW]–$R13)–($AC_PARAM[–2+Z_RW]–$R14) Z_RW=Z_RW+2 $R2=$R2+$R1

; ***** Activate axis BACH as endlessly turning rotary axis with MOV *********
WAITP(BACH)
ID=3 EVERY $R6==1 DO MOV[BACH]=1 FA[BACH]=1000          ; Activate
ID=4 EVERY $R6==0 und ($AA_STAT[BACH]==1) DO MOV[BACH]=0 ; Deactivate

; Measure in succession, store in FIFO 1, MT2 neg, MT2 pos edge
; the distance between 2 teeth falling edge ... rising edge, probe 2, is measured
N310 ID=5 WHEN $R7==1 DO MEAC[BACH]=(2, 1, –2, 2)
N320 ID=6 WHEN (Z_MW>=M_TEETH) DO MEAC[BACH]=(0)        ; Abort measurement
M00
STOPRE


; ******* Fetch FIFO values and store ***
N400 R3=$AC_PARAM[0]                   ; Content of first element to be read in
                                       ; Reset FIFO1[4] variable and ensure a defined
                                       ; measurement trace for next measurement job

N500 $AC_FIFO1[4]=0


; ****** Calculate difference between individual teeth
N510 R4=R2/(NO.TEETH)/1000             ; R4 corresponds to an average distance between teeth
                                       ; “/1000” division is omitted in later SW versions
```

```
; ******* Calculate center position **********
N520 R3=R3/1000 ; First measurement position converted to degrees
N530 R3=R3 MOD 360 ; First measurement point modulo
N540 R5=(R3–R14)+(R4/2) ; Calculate gap position
M00
stopre
R6=0                                ; Deactivate rotation of BACH axis
gotob start
M30
```

6.7 Axis couplings via synchronized actions

6.7.1 Coupling to leading axis

Task assignment A cyclic curve table is defined by means of polynomial segments. Controlled by means of arithmetic variables, the movement of the master axis and the coupling process between master and slave axes is activated/deactivated.

```
%_N_KOP_SINUS_MPF

N5 R1=1                          ; ID 1, 2 activate/deactivate coupling: LEADON (CACB, BACH)
N6 R2=1                          ; ID 3, 4 Master axis movement on/off: MOV BACH
N7 R5=36000                      ; BACH feedrate/min
N8 STOPRE

;**** Define periodic table no. 4 using polynomial segments ****
N10 CTABDEF (YGEO,XGEO,4,1)
N16 G1 F1200 XGEO=0.000 YGEO=0.000    ; Approach initial positions
N17 POLY PO[XGEO]=(79.944,3.420,0.210) PO[YGEO]=(24.634,0.871,–9.670)
N18 PO[XGEO]=(116.059,0.749,–0.656) PO[YGEO]=(22.429,–5.201,0.345)
N19 PO[XGEO]=(243.941,–17.234,11.489) PO[YGEO]=(–22.429,–58.844,39.229)
N20 PO[XGEO]=(280.056,1.220,–0.656) PO[YGEO]=(–24.634,4.165,0.345)
N21 PO[XGEO]=(360.000,–4.050,0.210) PO[YGEO]=(0.000,28.139,–9.670)
N22 CTABEND    ; **** End of table definition*****

; Traverse master axis and coupled axis in rapid mode to initial position
N80 G0 BACH=0 CACH=0             ; Channel axis names
N50 LEADOF(CACH,BACH)            ; Existing coupling OFF if necessary

N235 ;******* Activation of coupled motion for axis CACH *******
N240 WAITP(CACH)                  ; Synchronize axis with channel
N245 ID=1 EVERY $R1==1 DO LEADON(CACH, BACH, 4)      ; Couple using table 4
N250 ID=2 EVERY $R1==0 DO LEADOF(CACH, BACH)         ; Deactivate coupling

N265 WAITP(BACH)

N270 ID=3 EVERY $R2==1 DO MOV[BACH]=1 FA[BACH]=R5   ; Turn master axis endlessly at feedrate in R5
N275 ID=4 EVERY $R2==0 DO MOV[BACH]=0               ; Stop master axis

N280 M00
N285 STOPRE
N290 R1=0                                ; Deactivate coupling condition
N295 R2=0                                ; Deactivate condition for master axis rotation
N300 R5=180                              ; New feedrate for BACH
N305 M30
```

6.7.2 Non-circular grinding via master value coupling

Task assignment

A non-circular workpiece that is rotating on axis CACH must be machined by grinding. The distance between the grinding wheel and workpiece is controlled by axis XACH and depends on the angle of rotation of the workpiece. The inter-relationship between angles of rotation and assigned movements is defined in curve table 2. The workpiece must move at velocities that are determined by the workpiece contour defined in curve table 1.

Solution

CACH is designated as the leading axis in a coupling. It controls:
– The compensatory motion of axis XACH via table 2 and
– Software axis CASW via table 1.

The axis override of axis CACH is determined by the actual values of axis CASW, thus providing the required contour-dependent velocity of axis CACH.

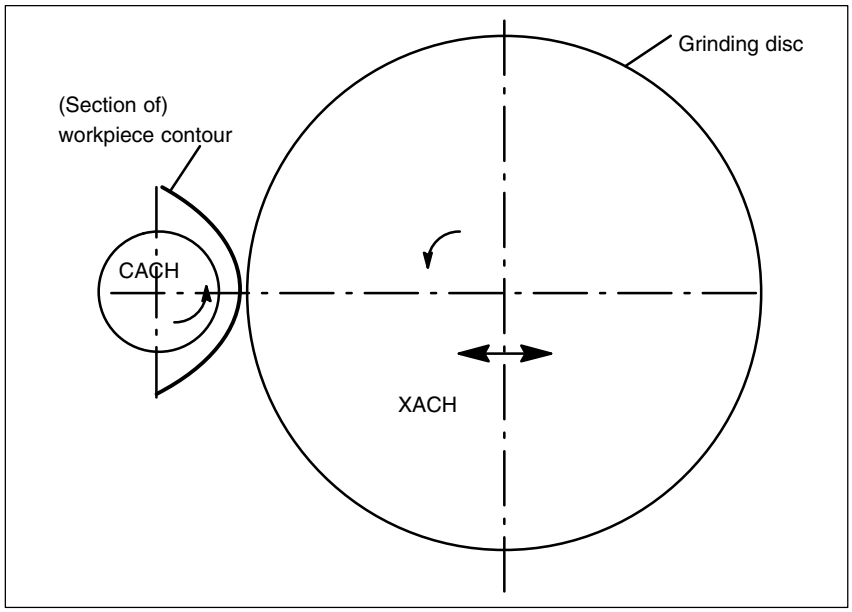


Fig. 6-5 Diagrammatic representation of non-circular contour grinding

```
%_N_CURV_TABS_SPF
PROC CURV_TABS
N160   ; ********** Table 1 Define override ******
N165 CTABDEF(CASW,CACH,1,1)              ; Periodic table 1
N170 CACH=0 CASW=10
N175 CACH=90 CASW=10
N180 CACH=180 CASW=100
N185 CACH=350 CASW=10
N190 CACH=359.999 CASW=10
N195 CTABEND
```

```
N160   ; **** Table 2 Define linear compensatory motion of XACH ******
CTABDEF(YGEO,XGEO,2,1)                    ; Periodic table 2
N16 XGEO=0.000 YGEO=0.000
N16 XGEO=0.001 YGEO=0.000
N17 POLY PO[XGEO]=(116.000,0.024,0.012) PO[YGEO]=(4.251,0.067,–0.828)
N18 PO[XGEO]=(244.000,0.072,–0.048) PO[YGEO]=(4.251,–2.937)
N19 PO[XGEO]=(359.999,–0.060,0.012) PO[YGEO]=(0.000,–2.415,0.828)
N16 XGEO=360.000 YGEO=0.000
N20 CTABEND

M17


%_N_UNRUND_MPF
; Coupled axis grouping for non-circular machining

; XACH is the infeed axis of the grinding wheel
; CACH is the workpiece axis as a rotary axis and master axis
; Application: Grind non-circular contour
; Table 1 mirrors the override for axis CACH as a function of the position of CACH
; Override of XGEO axis with handwheel infeed for scratching

N100 DRFOF                          ; Deselect handwheel override
N200 MSG("select DRF, (handwheel 1 active) and select INCREMENT.== handwheel override ACTIVE")
N300 M00
N500 MSG()                          ; Reset message
N600 R2=1                           ; LEADON Table 2, activate with ID=3/4 CACH to XACH
N700 R3=1                           ; LEADON Table 1, activate with ID=5/6 CACH to CASW, override
N800 R4=1                           ; Endlessly turning rotary axis CACH, start with ID=7/8
N900 R5=36000                       ; FA[CACH] Endlessly turning rotary speed

N1100 STOPRE
N1200                               ; ********* Set axes and master axis to slave axis *******
                                    ; Traverse master and slave axes to initial positions
N1300 G0 XGEO=0 CASW=10 CACH=0
N1400 LEADOF(XACH,CACH)             ; Coupling OFF XACH compensatory movement
N1500 LEADOF(CASW,CACH)             ; Coupling OFF CASW override table

N1600 CURV_TABS                     ; Subprogram with definition of tables

N1700                               ; ******* Activate LEADON compensatory motion XACH *******
N1800 WAITP(XGEO)                   ; Synchronize axis with channel
N1900 ID=3 EVERY $R2==1 DO LEADON(XACH,CACH,2)
N2000 ID=4 EVERY $R2==0 DO LEADOF(XACH,CACH)

N2100                               ; ************ Activate LEADON CASW override table ****
N2200 WAITP(CASW)
N2300 ID=5 EVERY $R3==1 DO LEADON(CASW,CACH,1)      ; CTAB coupling ON master axis CACH
N2400 ID=6 EVERY $R3==0 DO LEADOF(CASW,CACH)        ; CTAB coupling OFF master axis CACH

N2500                               ; ** Control CASH override from position CASW with ID 10 *
N2700 ID=11 DO $$AA_OVR[CACH]=$AA_IM[CASW]          ; Assign "axis position" CASW to OVR CACH

N2900 WAITP(CACH)

N3000 ID=7 EVERY $R4==1 DO MOV[CACH]=1 FA[CACH]=R5  ; Start as endlessly turning rotary axis
N3100 ID=8 EVERY $R4==0 DO MOV[CACH]=0              ; Stop as endlessly turning rotary axis

N3200 STOPRE
N3300 R90=$AA_COUP_ACT[CASW]                        ; Status of coupling for CASW for checking
N3400 MSG("activate CASW override table with LEADON"<<R90<<", go to END with NC START")
```

```
N3500 M00                ; ********** NC STOP **************
N3600 MSG()
N3700 STOPRE             ; Preprocessing stop
N3800 R1=0               ; Stop with ID=2 CASW axis as endlessly turning rotary axis
N3900 R2=0               ; LEADOF with ID=6 FA XACH and leading axis CACH
N4000 R3=0               ; LEADOF TAB1 CASW with ID=7/8 CACH to CASW override table
N4100 R4=0               ; Stop axis as endlessly turning rotary axis, ID=4 CACH
N4200 M30
```

Expansion options

The example above can be expanded by the following components:

- Introduction of a Z axis to move the grinding wheel or workpiece from one non-circular operation to the next on the same shaft (cam shaft).
- Switchover between tables if the cams have different contours, e.g. for inlet and outlet.
  ID = ... <condition> DO LEADOF(XACH, CACH) LEADON(XACH, CACH, <new table number>)
- Dressing of grinding wheel by means of online tool offset acc. to Subsection 2.4.7.

6.7.3 On-the-fly parting

Task assignment

An extruded material which passes continuously through the operating area of a cutting tool must be cut into parts of equal length.

X axis: Axis in which the extruded material moves. WCS
X1 axis: Machine axis of extruded material, MCS
Y axis: Axis in which cutting tool “tracks” the extruded material

For the purpose of this example, it is assumed that the cutting tool infeed is controlled via the PLC. The signals at the PLC interface can be evaluated to determine whether the extruded material and cutting tool are synchronized.

Actions

Activate coupling, LEADON
Deactivate coupling, LEADOF
Set actual values, PRESETON

NC program	Remarks
%_N_SCHERE1_MPF	
;$PATH=/_N_WKS_DIR/_N_DEMOFBE_WPD	
N100 R3=1500	; Length of a part to be cut off
N200 R2=100000 R13=R2/300	
N300 R4=100000	
N400 R6=30	; Start position Y axis
N500 R1=1	; Start condition for belt axis
N600 LEADOF(Y,X)	; Delete any existing coupling
N700 CTABDEF(Y,X,1,0)	; Table definition
N800 X=30 Y=30	; Value pairs
N900 X=R13 Y=R13	
N1000 X=2*R13 Y=30	
N1100 CTABEND	; End of table definition
N1200 PRESETON(X1,0)	; PRESET at beginning
N1300 Y=R6 G0	; Start pos. Y axis
	; Axis is linear
N1400 ID=1 EVERY $AA_IW[X]>$R3 DO PRESETON(X1,0)	; PRESET after length R3, PRESTON only permitted ; with WHEN and EVERY
	; New start after material parting
N1500 WAITP(Y)	
N1800 ID=6 EVERY $AA_IM[X]<10 DO LEADON(Y,X,1)	; Couple Y to X via Table 1 when X < 10
N1900 ID=10 EVERY $AA_IM[X]>$R3–30 DO LEADOF(Y,X)	; decouple when X > 30 from start of cutting ; length 30
N2000 WAITP(X)	
N2100 ID=7 WHEN $R1==1 DO MOV[X]=1 FA[X]=$R4	; Set extruded material axis continuously in motion
N2200 M30	

6.8 Technology cycles position spindle

Application

Interacting with the PLC program, the spindle which initiates a tool change must be:

- Traversed to an initial position or
- Positioned at a specific point at which the tool to be inserted is also located.

See Subsections 2.4.12, 2.6.1.

Coordination

The PLC and NCK are coordinated by means of the common data that are provided in SW version 4 and later (see Subsection 2.3.11)

- $A_DBB[0] 1 traverse to initial position
- $A_DBB[1] 1 traverse to target position
- $A_DBW[1] Position value + / –, PLC calculates the shortest route.

Synchronized actions

%_N_MAIN_MPF

```
...
IDS=1 EVERY $A_DBB[0]==1 DO NULL_POS          ; If $A_DBB[0] is set by PLC, traverse to initial position
IDS=2 EVERY $A_DBB[1]==1 DO ZIEL_POS          ; If $A_DBB[1] is set by PLC, traverse spindle to
                                              ; position stored in $A_DBW[1]
...
```

Technology cycle NULL_POS

%_N_NULL_POS_SPF

```
PROC NULL_POS
SPOS=0                                        ; Move drive for tool change into initial position
$A_DBB[0]=0                                   ; Initial position executed in NCK
```

Technology cycle ZIEL_POS

%_N_ZIEL_POS_SPF

```
PROC TARGET_POS
SPOS=IC($A_DBW[1])                            ; Traverse spindle to position value that has been stored
                                              ; in $A_DBW[1] by the PLC, incremental dimension
$A_DBB[1]=0                                   ; Target position executed in NCK
```

6.9 Synchronized actions in the TCC/MC area

Introduction

The following figure shows the schematic structure of a tool-changing cycle.

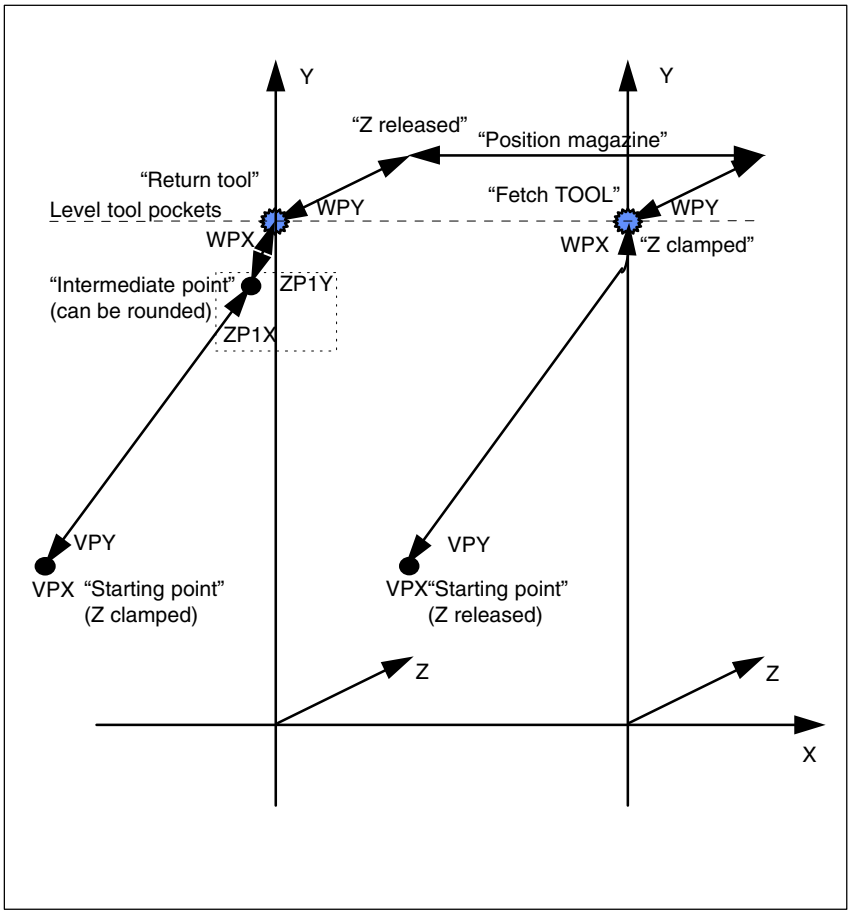


Fig. 6-6 Schematic sequence for tool-changing cycle

Flowchart

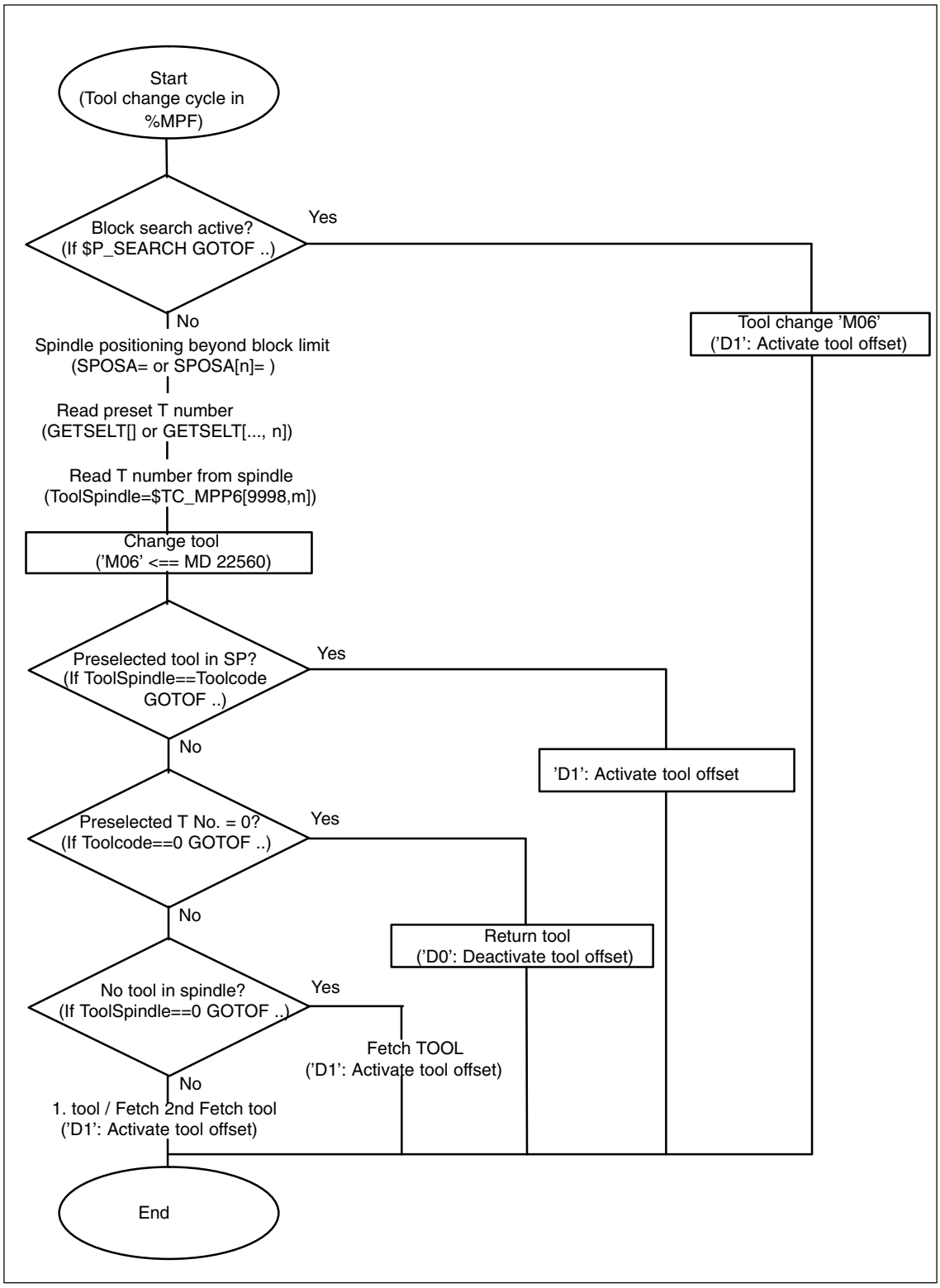


Fig. 6-7 Flowchart for tool-changing cycle

NC program **Remarks**

%_N_WZW_SPF

```
;$PATH=/_N_SPF_DIR
N10 DEF INT ToolCode, ToolSpindle
N15 WHEN $AC_PATHN<10 DO $AC_MARKER[0]=0 $AC_MARKER[1]=0 $AC_MARKER[2]=0
N20 ID=3 WHENEVER $A_IN[9]==TRUE DO $AC_MARKER[1]=1          ; Marker to = 1 when MagAxis traversed
N25 ID=4 WHENEVER $A_IN[10]==TRUE DO $AC_MARKER[2]=1         ; Marker to = 1 when MagAxis traversed
N30 IF $P_SEARCH GOTOF tc_preprocessing                      ; Block search active? –>
N35 SPOSA=0 D0
N40 GETSELT(ToolCode)                                        ; Read preselected T No.
N45 ToolSpindle=$TC_MPP6[9998,1]                             ; Read tool in spindle
N50 M06
N55 IF ToolSpindle==ToolCode GOTOF tool_in_spindle IF ToolCode==0 GOTOF return1 IF ToolSpindle==0 GO-
TOF fetch1

;*****Fetch and return tool*****
return1fetch1:
N65 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[1]=1             ;If MagAxis traverses marker = 1
N70 G01 G40 G53 G64 G90 X=magazine1VPX Y=magazine1VPY Z=magazine1Zclamped F70000 M=QU(120)
M=QU(123) M=QU(9)
N75 WHENEVER $AA_STAT[S1]<>4 DO $AC_OVR=0                    ; Spindle in position
N80 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[1]=1             ; MagAxis traversing request
N85 WHENEVER $AC_MARKER[1]==0 DO $AC_OVR=0                   ; Override=0 if axis not traversed
N90 WHENEVER $AA_STAT[C2]<>4 DO $AC_OVR=0          ; Override=0 if MagAxis not in pos. fine
N95 WHENEVER $AA_DTEB[C2]>0 DO $AC_OVR=0                     ; Override=0 if distance-to-go MagAxis > 0
N100 G53 G64 X=magazine1ZP1X Y=magazine1ZP1Y F60000
N105 G53 G64 X=magazine1WPX Y=magazine1WPY F60000
N110 M20                                                     ; Release tool
N115 G53 G64 Z=MR_magazine1Zreleased F40000
N120 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[2]=1;
N125 WHENEVER $AC_MARKER[2]==0 DO $AC_OVR=0
N130 WHENEVER $AA_STAT[C2]<>4 DO $AC_OVR=0
N135 WHENEVER $AA_DTEB[C2]>0 DO $AC_OVR=0
N140 G53 G64 Z=magazine1Zclamped F40000
N145 M18                                                     ; Clamp tool
N150 WHEN $AC_PATHN<10 DO M=QU(150) M=QU(121)                ; Condition always satisfied
N155 G53 G64 X=magazine1VPX Y=magazine1VPY F60000 D1 M17

;*****Return tool*****
return1:
N160 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[1]=1
N165 G01 G40 G53 G64 G90 X=magazine1VPX Y=magazine1VPY Z=magazine1Zclamped F70000 M=QU(120)
M=QU(123) M=QU(9)
N170 WHENEVER $AA_STAT[S1]<>4 DO $AC_OVR=0
N175 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[1]=1
N180 WHENEVER $AC_MARKER[1]==0 DO $AC_OVR=0
N185 WHENEVER $AA_STAT[C2]<>4 DO $AC_OVR=0 N190 WHENEVER $AA_DTEB[C2]>0 DO $AC_OVR=0
N195 G53 G64 X=magazine1ZP1X Y=magazine1ZP1Y F60000
N200 G53 G64 X=magazine1WPX Y=magazine1WPY F60000
N205 M20                                                     ; Release tool
N210 G53 G64 Z=magazine1Zreleased F40000
N215 G53 G64 X=magazine1VPX Y=magazine1VPY F60000 M=QU(150) M=QU(121) D0 M17

;*****Fetch tool*****
fetch1:
N220 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[2]=1
N225 G01 G40 G53 G64 G90 X=magazine1VPX Y=magazine1VPY Z=magazine1Zreleased F70000 M=QU(120)
M=QU(123) M=QU(9)
N230 G53 G64 X=magazine1WPX Y=magazine1WPY F60000
N235 WHENEVER $AA_STAT[S1]<>4 DO $AC_OVR=0
N240 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[2]=1
N245 WHENEVER $AC_MARKER[2]==0 DO $AC_OVR=0
N250 WHENEVER $AA_STAT[C2]<>4 DO $AC_OVR=0
```

```
N255 WHENEVER $AA_DTEB[C2]>0 DO $AC_OVR=0
N260 G53 G64 Z=magazine1Zclamped F40000
N265 M18                                                                    ; Clamp tool
N270 G53 G64 X=magazine1VPX Y=magazine1VPY F60000 M=QU(150) M=QU(121) D1 M17

;*****Tool in spindle *****
tool_in_spindle:
N275 M=QU(121) D1 M17

;*****Block search*****
tc_block_search:
N280 STOPRE
N285 D0
N290 M06
N295 D1 M17
```

■

Notes

7 Data Fields, Lists

7.1 Interface signals

DB number	Bit, byte	Name	Reference
Channelspecific			
21–30	280.1	Disable modal synchronized actions acc. to DBX 300.0–307.7	
21–30	300.0 –	Disable modal synchronized actions acc. to DBX 300.0–307.7, acknowledgment from NCK	
21–30	300.0 –	Disable modal synchronized actions ID or IDS 1 –	
21–30	307.7	64 disable. Request to NCK channel	
21–30	308.0 –	Modal synchronized actions ID or IDS 1 –	
21–30	315.7	64 can be disabled. Message from NCK.	

7.2 Machine data

Number	Name of identifier	Name	Reference
General ($MN_ ...)			
11110	AUXFU_GROUP_SPEC	Auxiliary function group specification	H2
11500	PREVENT_SYNACT_LOCK	Protected synchronized actions	
???	MM_MAINTENANCE_MON	Activate recording of maintenance data	
Channelspecific ($MC_ ...)			
21240	PREVENT_SYNACT_LOCK_CHAN	Protected synchronized actions for channel	
28250	MM_NUM_SYNC_ELEMENTS	Number of elements for expressions in synchronized actions	
28252	MM_NUM_FCTDEF_ELEMENTS	Number of FCTDEF elements	
28254	MM_NUM_AC_PARAM	Number of $AC_PARAM parameters	
28255	MM_BUFFERED_AC_PARAM	Storage location for $AC_PARAM (SW 6.3 and later)	
28256	MM_NUM_AC_MARKER	Number of $AC_MARKER markers	
28257	MM_BUFFERED_AC_MARKER	Storage location for $AC_MARKER (SW 6.3 and later)	
28258	MM_NUM_AC_TIMER	Number of $AC_TIMER time variables	
28260	NUM_AC_FIFO	Number of $AC_FIFO1, $AC_FIFO2, ... variables	
28262	START_AC_FIFO	Store FIFO variables from R parameter	
28264	LEN_AC_FIFO	Length of $AC_FIFO ... FIFO variables	

ChannelSpecific ($MC_ ...)			
28266	MODE_AC_FIFO	FIFO processing mode	

Axisspecific ($MA_ ...)			
30450	IS_CONCURRENT_POS_AX	Competing positioning axis	P2
32060	POS_AX_VELO	Initial setting for positioning axis velocity	P2
32070	CORR_VELO	Axis speed for handwheel, ext. ZO, cont. dressing, clearance control (SW 3 and later)	H1
32074	FRAME_OR_CORRPOS_NOTALLOWED	Effectiveness of frames and tool length compensation	
32920	AC_FILTER_TIME	Filter smoothing time constant for Adaptive Control (SW2 and later)	
36750	AA_OFF_MODE	Effect of value assignment for axial override for synchronized actions (SW3 and later)	
37200	COUPLE_POS_TOL_COARSE	Threshold value for “Coarse synchronism”	S3
37210	COUPLE_POS_TOL_FINE	Threshold value for “Fine synchronism”	S3
???	MAINTENANCE_DATA	Setting for maintenance data to be recorded	
Setting data ($SA_ ...)			
43300	ASSIGN_FEED_PER_REV_SOURCE	Rotational feedrate for positioning axes/spindles	V1
43350	AA_OFF_LIMIT	Upper limit of offset value for $AA_OFF clearance control	
43400	WORKAREA_PLUS_ENABLE	Working area limitation in pos. direction	A3

7.3 Alarms

For more detailed information about the alarms, see
References: /DA/, “Diagnostics Guide”
or, for systems with MMC 101/102, the online help.

■

Notes

References

A

General Documentation

/BU/ SINUMERIK & SIMODRIVE, Automation Systems for Machine Tools
Catalog NC 60
Order number: E86060–K4460–A101–A9–7600

/IKPI/ Industrial Communication and Field Devices
Catalog IK PI
Order number: E86060–K6710–A101–B2–7600

/ST7/ SIMATIC
Products for Totally Integrated Automation and Micro Automation
Catalog ST 70
Order number: E86060–K4670–A111–A8–7600

/Z/ MOTION–CONNECT
Connections & System Components for SIMATIC, SINUMERIK, MASTERDRIVES, and SIMOTION
Catalog NC Z
Order number: E86060–K4490–A001–B1–7600

Safety Integrated
Application Manual
The safety program for the industries of the world
Order number: 6ZB5000–0AA02–0BA0

Electronic Documentation

/CD1/ The SINUMERIK System (03.04 Edition)
DOC ON CD
(includes all SINUMERIK 840D/840Di/810D/802 and SIMODRIVE publications)
Order number: 6FC5298–7CA00–0BG0

User Documentation

/AUK/	SINUMERIK 840D/810D **AutoTurn** Short Operating Guide Order number: 6FC5298–4AA30–0BP2	(09.99 Edition)
/AUP/	SINUMERIK 840D/810D **AutoTurn Graphic Programming System** Programming/Setup Order number: 6FC5298–4AA40–0BP3	(02.02 Edition)
/BA/	SINUMERIK 840D/810D **MMC** Operator's Guide Order number: 6FC5298–6AA00–0BP0	(10.00 Edition)
/BAD/	SINUMERIK 840D/840Di/810D Operator's Guide **HMI Advanced** Order number: 6FC5298–6AF00–0BP3	(03.04 Edition)
/BAH/	SINUMERIK 840D/840Di/810D Operator's Guide **HT 6** Order number: 6FC5298–0AD60–0BP3	(03.04 Edition)
/BAK/	SINUMERIK 840D/840Di/810D **Short Operating Guide** Order number: 6FC5298–6AA10–0BP0	(02.01 Edition)
/BAM/	SINUMERIK 810D/840D **ManualTurn** Operator's Guide Order number: 6FC5298–6AD00–0BP0	(08.02 Edition)
/BAS/	SINUMERIK 840D/840Di/810D Operator's Guide **ShopMill** Order number: 6FC5298–6AD10–0BP2	(11.03 Edition)
/BAT/	SINUMERIK 840D/810D Operator's Guide **ShopTurn** Order number: 6FC5298–6AD50–0BP2	(06.03 Edition)
/BEM/	SINUMERIK 840D/810D Operator's Guide **HMI Embedded** Order number: 6FC5298–6AC00–0BP3	(03.04 Edition)

/BNM/	SINUMERIK 840D840Di//810D Operator's Guide **Measuring Cycles** Order number: 6FC5298–6AA70–0BP3	(03.04 Edition)
/BTDI/	SINUMERIK 840D840Di//810D Motion Control Information System (MCIS) User's Guide **Tool Data Information** Order number: 6FC5297–6AE01–0BP0	(04.03 Edition)
/CAD/	SINUMERIK 840D/840Di/810D Operator's Guide **CAD Reader** Order number: (part of the online help)	(03.02 Edition)
/DA/	SINUMERIK 840D/840Di/810D **Diagnostics Guide** Order number: 6FC5298–7AA20–0BP0	(03.04 Edition)
/KAM/	SINUMERIK 840D/810D Short Guide **ManualTurn** Order number: 6FC5298–5AD40–0BP0	(04.01 Edition)
/KAS/	SINUMERIK 840D/810D Short Guide **ShopMill** Order number: 6FC5298–5AD30–0BP0	(04.01 Edition)
/KAT/	SINUMERIK 840D/810D Short Guide **ShopTurn** Order number: 6FC5298–6AF20–0BP0	(07.01 Edition)
/PG/	SINUMERIK 840D/840Di/810D Programming Guide **Fundamentals** Order number: 6FC5298–7AB00–0BP0	(03.04 Edition)
/PGA/	SINUMERIK 840D/840Di/810D Programming Guide **Advanced** Order number: 6FC5298–7AB10–0BP0	(03.04 Edition)
/PGA1/	SINUMERIK 840D/840Di/810D Lists Manual **System Variables** Order number: 6FC5298–7AE10–0BP0	(03.04 Edition)
/PGK/	SINUMERIK 840D/840Di/810D Short Guide **Programming** Order number: 6FC5298–6AB30–0BP0	(10.00 Edition)

/PGM/ SINUMERIK 840D/840Di/810D (11.02 Edition)
Programming Guide **ISO Milling**
Order number: 6FC5298–6AC20–0BP2

/PGT/ SINUMERIK 840D/840Di/810D (11.02 Edition)
Programming Guide **ISO Turning**
Order number: 6FC5298–6AC10–0BP2

/PGZ/ SINUMERIK 840D840Di/810D (03.04 Edition)
Programming Guide **Cycles**
Order number: 6FC5298–7AB40–0BP0

/PI/ **PCIN 4.4**
Software for Data Transfer to/from MMC Modules
Order number: 6FX2060–4AA00–4XB0 (English, German, French)
Order from: WK Fürth

/SYI/ SINUMERIK 840Di (02.01 Edition)
System Overview
Order number: 6FC5298–6AE40–0BP0

Manufacturer/Service Documentation

a) Lists

/LIS/ SINUMERIK 840D/840Di/810D (03.04 Edition)
SIMODRIVE 611D
Lists
Order number: 6FC5297–7AB70–0BP0

b) Hardware

/ASAL/ SIMODRIVE (10.03 Edition)
Planning Guide **General Part for Asynchronous Motors**
Order number: 6SN1197–0AC62–0BP0

/APH2/ SIMODRIVE (10.03 Edition)
Planning Guide **Asynchronous Motors 1PH2**
Order number: 6SN1197–0AC63–0BP0

/APH4/ SIMODRIVE (10.03 Edition)
Planning Guide **Asynchronous Motors 1PH4**
Order number: 6SN1197–0AC64–0BP0

/APH7/ SIMODRIVE (12.03 Edition)
Planning Guide **Asynchronous Motors 1PH7**
Order number: 6SN1197–0AC65–0BP0

/APL6/ MASTERDRIVES VC/MC (12.03 Edition)
Planning Guide **Asynchronous Motors 1PL6**
Order number: 6SN1197–0AC66–0BP0

/BH/ SINUMERIK 840D840Di//810D (11.03 Edition)
Operator Components Manual
Order number: 6FC5297–6AA50–0BP3

/BHA/ SIMODRIVE **Sensor** (03.03 Edition)
Absolute Encoder with Profibus DP User's Guide (HW)
Order number: 6SN1197–0AB10–0YP2

/EMV/ SINUMERIK, SIROTEC, SIMODRIVE (06.99 Edition)
Planning Guide **EMC Installation Guide**
Order number: 6FC5297–0AD30–0BP1

The up-to-date declaration of conformity can be viewed on the Internet at
http://www4.ad.siemens.de

Please enter the ID No.: 15257461 in the "Find"/"Search" field (upper right) and click "go".

/GHA/ SINUMERIK/ SIMOTION (02.03 Edition)
ADI4 – Analog Drive Interface for 4 Axes
Manual
Order number: 6FC5297–0BA01–0BP1

/PFK6/ SIMODRIVE 611, SIMOVERT MASTERDRIVES MC (05.03 Edition)
Planning Guide **Synchronous Servo Motors 1FK6**
Order number: 6SN1197–0AD05–0BP0

/PFK7/ SIMODRIVE 611, SIMOVERT MASTERDRIVES MC (01.03 Edition)
Planning Guide **Synchronous Servo Motors 1FK7**
Order number: 6SN1197–0AD06–0BP0

/PFS6/ SIMODRIVE 611, SIMOVERT MASTERDRIVES MC (04.04 Edition)
Planning Guide **Synchronous Servo Motors 1FS6**
Order number: 6SN1197–0AD08–0BP1

/PFT5/ SIMODRIVE (05.03 Edition)
Planning Guide **Synchronous Servo Motors 1FT5**
Order number: 6SN1197–0AD01–0BP0

/PFT6/ SIMODRIVE (12.03 Edition)
Planning Guide **Synchronous Servo Motors 1FT6**
Order number: 6SN1197–0AD02–0BP0

/PFU/ SINAMICS, SIMOVERT MASTERDRIVES (09.03 Edition)
MICROMASTER
SIEMOSYN Motors 1FU8
Order number: 6SN1197–0AC80–0BP0

/PHC/ SINUMERIK 810D (11.02 Edition)
Configuring Manual (HW)
Order number: 6FC5297–6AD10–0BP1

/PHD/ SINUMERIK 840D (11.03 Edition)
Configuring Manual (HW)
Order number: 6FC5297–6AC10–0BP3

/PJAL/ SIMODRIVE 611/MASTERDRIVES MC (01.03 Edition)
Planning Guide **Synchronous Servo Motors**
General Part for 1FT/1FK Motors
Order number: 6SN1197–0AD07–0BP0

/PJAS/ SIMODRIVE (07.03 Edition)
Planning Guide for **Asynchronous Motors**
Order number: 6SN1197–0AC61–0BP0

/PJFE/ SIMODRIVE (02.03 Edition)
Planning Guide **Built–In Synchronous Motors 1FE1**
Three–Phase AC Motors for Main Spindle Drives
Order number: 6SN1197–0AC00–0BP4

/PJF1/ SIMODRIVE (12.02 Edition)
Installation Guide **Built–In Synchronous Motors 1FE1 051.–1FE1 147**.
Three–Phase AC Motors for Main Spindle Drives
Order number: 610.43000.02

/PJLM/ SIMODRIVE (06.02 Edition)
Planning Guide **Linear Motors 1FN1, 1FN3**

ALL	General Information on the Linear Motor
1FN1	Three–Phase AC Linear Motor 1FN1
1FN3	Three–Phase AC Linear Motor 1FN3
CON	Connections

Order number: 6SN1197–0AB70–0BP3

/PJM/ SIMODRIVE (11.00 Edition)
Planning Guide **Servo Motors**
Three–Phase Motors for Feed and Main Spindle Drives
Order number: 6SN1197–0AC20–0BP4

/PJM2/ SIMODRIVE (07.03 Edition)
Planning Guide for **Synchronous Servo Motors**
Content: General Part, 1FT5, 1FT6, 1FK6, 1FK7, 1FS6
Order number: 6SN1197–0AA20–0BP0

/PJTM/ SIMODRIVE (05.03 Edition)
Planning Guide **Built–In Torque Motors 1FW6**
Order number: 6SN1197–0AD00–0BP1

/PJU/ SIMODRIVE 611 (02.03 Edition)
Planning Guide **Converters**
Order number: 6SN1197–0AA00–0BP6

/PKTM/ SIMODRIVE (09.03 Edition)
Planning Guide **Complete Torque Motors 1FW3**
Order number: 6SN1197–0AC70–0BP0

/PMH/ SIMODRIVE **Sensor** (07.02 Edition)
Planning/Installation Guide
Hollow–Shaft Measuring System SIMAG H
Order number: 6SN1197–0AB30–0BP1

/PMHS/ SIMODRIVE (12.00 Edition)
Installation Guide **Measuring System for Main Spindle Drives**
Gear Wheel Encoder SIZAG2
Order number: 6SN1197–0AB00–0YP3

/PMS/ SIMODRIVE (02.03 Edition)
Planning Guide **ECO Motor Spindle for Main Spindle Drives**
Order number: 6SN1197–0AD04–0BP1

/PPH/ SIMODRIVE (12.01 Edition)
Planning Guide **1PH2 / 1PH4 / 1PH7 Motors**
Three–Phase Asynchronous Motors for Main Spindle Drives
Order number: 6SN1197–0AC60–0BP0

/PPM/ SIMODRIVE (11.01 Edition)
Planning Guide
Hollow–Shaft Motors for Main Spindle Drives **1PM4 and 1PM6**
Order number: 6SN1197–0AD03–0BP0

c) Software

/FB1/ SINUMERIK 840D/840Di/810D (03.04 Edition)
Description of Functions **Basic Machine (Part 1)**
(the various manuals are listed below)
Order number: 6FC5297–7AC20–0BP0

A2	Various Interface Signals
A3	Axis Monitoring, Protection Zones
B1	Continuous Path Mode, Exact Stop and Look Ahead
B2	Acceleration
D1	Diagnostic Tools
D2	Interactive Programming
F1	Travel to Fixed Stop
G2	Velocites, Setpoint/Actual Value Systems, Closed-Loop Control
H2	Output of Auxiliary Functions to PLC
K1	Mode Group, Channels, Program Operation Mode
K2	Axis Types, Coordinate Systems, Frames, Actual-Value System for Workpiece, External Zero Offset
K4	Communication
N2	EMERGENCY STOP
P1	Traverse Axes
P3	Basic PLC Program
R1	Reference Point Approach
S1	Spindles
V1	Feeds
W1	Tool Compensation

/FB2/ SINUMERIK 840D/840Di/810D (03.04 Edition)
Description of Functions **Extended Functions (Part 2)**
including FM-NC: turning, stepper motor
(contents listed below)
Order no.: 6FC5297–7AC30–0BP0

A4	Digital and Analog NCK I/Os
B3	Several Operator Panels and NCUs
B4	Operation via PG/PC
F3	Remote Diagnosis
H1	Jog with/without Handwheel
K3	Compensations
K5	Mode Groups, Channels, Axis Replacement
L1	FM-NC Local Bus
M1	Kinematic Transformation
M5	Measurements
N3	Software Cams, Position Switching Signals
N4	Punching and Nibbling
P2	Positioning Axes
P5	Oscillation
R2	Rotary Axes
S3	Synchronous Spindles
S5	Synchronized Actions (SW 3 and lower, higher /FBSY/)
S6	Stepper Motor Control
S7	Memory Configuration
T1	Indexing Axes
W3	Tool Change
W4	Grinding

/FB3/ SINUMERIK 840D/840Di/810D (03.04 Edition)
Description of Functions **Special Functions (Part 3)**
(the various manuals are listed below)
Order number: 6FC5297–7AC80–0BP0

F2	3-Axis to 5-Axis Transformation
G1	Gantry Axes
G3	Cycle Times
K6	Contour Tunnel Monitoring
M3	Coupled Axes and ESR
S8	Constant Workpiece Speed for Centerless Grinding
S9	Setpoint Switching (S9)
T3	Tangential Control
TE0	Installation and Activation of Compile Cycles
TE1	Clearance Control
TE2	Analog Axis
TE3	Speed/Torque Coupling Master-Slave
TE4	Transformation Package Handling
TE5	Setpoint Exchange
TE6	MCS Coupling
TE7	Retrace Support
TE8	Unclocked Path-Synchronous Switching Signal Output
V2	Preprocessing
W5	3D Tool Radius Compensation

/FBA/ SIMODRIVE 611D/SINUMERIK 840D/810D (03.04 Edition)
Description of Functions **Drive Functions**
(the various manuals are listed below)
Order number: 6SN1197–0AA80–1BP1

DB1	Operational Messages/Alarm Reactions
DD1	Diagnostic Functions
DD2	Speed Control Loop
DE1	Extended Drive Functions
DF1	Enable Commands
DG1	Encoder Parameterization
DL1	Linear Motor MD
DM1	Calculation of Motor/Power Section Parameters and Controller Data
DS1	Current Control Loop
DÜ1	Monitors/Limitations

/FBAN/ SINUMERIK 840D/SIMODRIVE 611 DIGITAL (02.00 Edition)
Description of Functions **ANA MODULE**
Order number: 6SN1197–0AB80–0BP0

/FBD/ SINUMERIK 840D (07.99 Edition)
Description of Functions **Digitizing**
Order number: 6FC5297–4AC50–0BP0

DI1	Start-Up
DI2	Scanning with Tactile Sensors (scancad scan)
DI3	Scanning with Lasers (scancad laser)
DI4	Milling Program Generation (scancad mill)

/FBDM/ SINUMERIK 840D/840Di/810D (09.03 Edition)
Description of Functions NC Program Management
DNC Machines
Order number: 6FC5297–1AE81–0BP0

/FBDN/ SINUMERIK 840D/840Di/810D (03.03 Edition)
Motion Control Information System (MCIS)
Description of Functions **NC Program Management DNC**
Order number: 6FC5297–1AE80–0BP0

DN1	DNC Plant / DNC Cell
DN2	DNC IFC SINUMERIK, NC Data Transfer via Network

/FBFA/ SINUMERIK 840D/840Di/810D (11.02 Edition)
Description of Functions **ISO Dialects for SINUMERIK**
Order number: 6FC5297–6AE10–0BP3

/FBFE/ SINUMERIK 840D/810D (04.03 Edition)
Motion Control Information System (MCIS)
Description of Functions **Remote Diagnosis**
Order number: 6FC5297–0AF00–0BP2

FE1	Remote Diagnosis (ReachOut)
FE3	RCS Host / RCS Viewer (pcAnywhere)

/FBH/ SINUMERIK 840D/840Di/810D
HMI Programming Package (11.02 Edition)
Order number: (included with the software)

Part 1	Operator's Guide
Part 2	Description of Functions

/FBH1/ SINUMERIK 840D/840Di/810D
HMI Programming Package (03.03 Edition)
ProTool/Pro Option SINUMERIK
Order number: (included with the software)

/FBHL/ SINUMERIK 840D/SIMODRIVE 611 digital (10.03 Edition)
Description of Functions **HLA Module**
Order number: 6SN1197–0AB60–0BP3

/FBIC/ SINUMERIK 840D/840Di/810D (06.03 Edition)
Motion Control Information System (MCIS)
Description of Functions **TDI Ident Connection**
Order number: 6FC5297–1AE60–0BP0

/FBMA/ SINUMERIK 840D/810D (08.02 Edition)
Description of Functions **ManualTurn**
Order number: 6FC5297–6AD50–0BP0

/FBO/ SINUMERIK 840D/810D (09.01 Edition)
Description of Functions Configuring of **OP 030 Operator Interface**
(the various sections are listed below)
Order number: 6FC5297–6AC40–0BP0

BA	Operator's Guide
EU	Development Environment (Configuring Package)
PSE	Introduction to Configuring of Operator Interface
IK	Screen Kit: Software Update and Configuration

/FBP/ SINUMERIK 840D (03.96 Edition)
Description of Functions **C–PLC Programming**
Order number: 6FC5297–3AB60–0BP0

/FBR/ SINUMERIK 840D/840Di/810D (09.03 Edition)
Description of Functions **Computer Link (SinCOM)**
Order number: 6FC5297–6AD61–0BP0

NFL Interface to Central Production Computer
NPL Interface to PLC/NCK

/FBSI/ SINUMERIK 840D/SIMODRIVE (11.03 Edition)
Description of Functions **SINUMERIK Safety Integrated**
Order number: 6FC5297–6AB80–0BP2

/FBSP/ SINUMERIK 840D/840Di/810D (08.03 Edition)
Description of Functions **ShopMill**
Order number: 6FC5297–6AD80–0BP1

/FBST/ SIMATIC (01.01 Edition)
Description of Functions **FM STEPDRIVE/SIMOSTEP**
Order number: 6SN1197–0AA70–0YP4

/FBSY/ SINUMERIK 840D/810D (10.02 Edition)
Description of Functions **Synchronized Actions**
Order number: 6FC5297–6AD40–0BP2

/FBT/ SINUMERIK 840D/810D (01.02 Edition)
Description of Functions **ShopTurn**
Order number: 6FC5297–6AD70–0BP1

/FBTC/ SINUMERIK 840D/810D (01.02 Edition)
IT Solutions
Description of Functions **Tool Data Communication SinTDC**
Order number: 6FC5297–5AF30–0BP0

/FBTD/ SINUMERIK 840D/810D (02.01 Edition)
IT Solutions
Description of Functions **Tool Information Systems (SinTDC)**
with Online Help
Order number: 6FC5297–6AE00–0BP0

/FBTP/ SINUMERIK 840D/840Di/810D (01.03 Edition)
Motion Control Information System (MCIS)
Description of Functions **Preventive Maintenance TPM**
Order number: Document is included with the software

/FBU/ **SIMODRIVE 611 universal/universal E** (07.03 Edition)
Description of Functions Closed-Loop Control Component
for Speed Control and Positioning
Order number: 6SN1197–0AB20–0BP8

/FBU2/ **SIMODRIVE 611 universal** (04.02 Edition)
Installation Guide
(included with every SIMODRIVE 611 universal)

/FBW/ SINUMERIK 840D/810D (11.02 Edition)
Description of Functions **Tool Management**
Order number: 6FC5297–6AC60–0BP1

/HBA/ SINUMERIK 840D/840Di/810D (03.02 Edition)
Manual **@Event**
Order number: 6AU1900–0CL20–0BA0

/HBI/ SINUMERIK 840Di (09.03 Edition)
Manual **SINUMERIK 840Di**
Order number: 6FC5297–6AE60–0BP2

/INC/ SINUMERIK 840D840Di//810D (06.03 Edition)
System Overview **Commissioning Tool SINUMERIK SinuCOM NC**
Order number: (part of the Online Help for the Startup Tool)

/PJE/ SINUMERIK 840D/810D (08.01 Edition)
Description of Functions **HMI Embedded Configuring Package**
Software Update, Configuration, Installation
Order number: 6FC5297–6EA10–0BP0

/PS/ SINUMERIK 840D/810D (09.03 Edition)
Planning Guide **Configuring Syntax**
This publication is included with the software and available as a pdf file.

/POS1/ SIMODRIVE **POSMO A** (08.03 Edition)
User Guide **Decentralized Positioning Motor on PROFIBUS DP**
Order number: 6SN2197–0AA00–0BP6

/POS2/ SIMODRIVE **POSMO A** (05.03 Edition)
Installation Guide (included with every POSMO A)

/POS3/ SIMODRIVE **POSMO SI/CD/CA** (07.03 Edition)
User Guide **Distributed Servo Drive Systems**
Order number: 6SN2197–0AA20–0BP5

/POS4/ SIMODRIVE **POSMO SI** (04.02 Edition)
Installation Guide (included with every POSMO SI)

/POS5/ SIMODRIVE **POSMO CD/CA** (04.02 Edition)
Installation Guide (included with every POSMO CD/CA)

/S7H/ SIMATIC S7–300 (2002 Edition)
Installation Manual **Technological Functions**
Order number: 6ES7398–8FA10–8BA0
– Reference Manual: **CPU data** (Hardware Reference Manual)
– Reference Manual: **Module Data**

/S7HT/ SIMATIC S7–300 (03.97 Edition)
Manual STEP 7, **Fundamentals**, V. 3.1
Order number: 6ES7810–4CA02–8BA0

/S7HR/ SIMATIC S7–300 (03.97 Edition)
Manual STEP 7, **Reference Manuals**, V. 3.1
Order number: 6ES7810–4CA02–8BR0

/S7S/ SIMATIC S7–300 (04.02 Edition)
FM 353 for Stepper Drive Positioning Module
Order together with configuring package

/S7L/ SIMATIC S7–300 (04.02 Edition)
FM 354 for Servo Drive Positioning Module
Order together with configuring package

/S7M/ SIMATIC S7–300 (01.03 Edition)
FM 357.2 Multimodule for Servo and Stepper Drives
Order together with configuring package

/SP/ SIMODRIVE **611–A/611–D**
SimoPro 3.1
Program for Configuring Machine-Tool Drives
Order number: 6SC6111–6PC00–0BA□,
Order from: WK Fürth

d) Start-up

/BS/ SIMODRIVE 611 analog (10.00 Edition)
Description **Start-Up Software for**
Main Spindle and Asynchronous Motor Modules Version 3.20
Order number: 6SN1197–0AA30–0BP1

/IAA/ SIMODRIVE 611A (10.00 Edition)
Installation and Start-Up Guide
Order number: 6SN1197–0AA60–0BP6

/IAC/ SINUMERIK 810D (11.02 Edition)
Installation and Start-Up Guide
(including description of SIMODRIVE 611D start-up software)
Order number: 6FC5297–6AD20–0BP1

/IAD/ SINUMERIK 840D/SIMODRIVE 611D (03.04 Edition)
I**nstallation and Start-Up Guide**
(including description of SIMODRIVE 611D start-up software)
Order number: 6FC5297–7AB10–0BP0

/IAM/ SINUMERIK 840D/840Di/810D (03.04 Edition)
Installation and Start-Up Guide HMI
Order number: 6FC5297–6AE20–0BP3

AE1	Updates/Options
BE1	Expanding the Operator Interface
HE1	Online Help
IM2	Start-Up HMI Embedded
IM4	Start-Up HMI Advanced
TX1	Setting Foreign Language Texts

Index

Symbols

A

B

C

D

E

F

G

I

J

L

M

N

O

P

R

S

T

W

Notes

To
SIEMENS AG
A&D MC BMS
P.O. Box 3180

D-91050 Erlangen, Germany

(Phone: ++49-(0)180-5050-222 [Hotline]
Fax: ++49-(0)9131-98-2176 [Documentation]
Email: motioncontrol.docu@erlf.siemens.de)

Suggestions

Corrections

For Publication/Manual:

SINUMERIK 840D/840Di/810D
Description of Functions
Synchronized Actions

Manufacturer/Service Documentation

From

Name

Company/Dept.

Address

Phone: /

Fax: /

Description of Functions

Order No.: 6FC5 297-7AD40-0BP0
Edition: 03.04

Should you come across any printing errors when reading this publication, please notify us on this sheet.
Suggestions for improvement are also welcome.

Suggestions and/or corrections

Overview of SINUMERIK 840D/840Di/810D Documentation (03.2004)

General Documentation

- *SINUMERIK* 840D/810D — Brochure
- *SINUMERIK* 840D/840Di/810D/ — Catalog Ordering Info. NC 60 ***)**
- *SINUMERIK SIROTEC SIMODRIVE* Accessories — Catalog Accessories NC-Z

User Documentation

- *SINUMERIK* 840D/810D/FM-NC — **AutoTurn** – Short Guide – Programming/Setup
- *SINUMERIK* 840D/840Di/810D — Operator's Guide **– HT 6**
- *SINUMERIK* 840D/840Di/810D — **Diagnostics Guide *)**
- *SINUMERIK* 840D/840Di/810D — **Operator's Guide *)** – Short Guide – HMI Embedded – HMI Advanced

User Documentation

- *SINUMERIK* 840D/840Di/810D — **Program. Guide** – Short Guide – Fundamentals ***)** – Advanced ***)** – Cycles – Measuring Cycles – ISO Turning/Milling
- *SINUMERIK* 840D/810D — Operator's Guide – **ManualTurn** – Short Guide ManualTurn – **ShopMill** – Short Guide ShopMill – **ShopTurn** – Short Guide ShopTurn
- *SINUMERIK* 840Di — **System Overview**

Manufacturer/Service Documentation

- *SINUMERIK* — **Configuring** (HW) ***)** – 810D – 840D
- *SINUMERIK* 840D/840Di/810D — **Operator Components** (HW) ***)**
- *SINUMERIK* 840D/810D — Description of Functions **– ManualTurn** **– ShopMill** **– ShopTurn**
- *SINUMERIK* 840D/840Di/810D — Description of Functions **Synchronized Actions**

Manufacturer/Service Documentation

- *SINUMERIK SIMODRIVE* 611D 840D/810D — Description of Functions **Drive Functions *)**
- *SINUMERIK* 840D/840Di/810D — **Description of Functions** – Basic Machine ***)** – Extended Functions – Special Functions
- *SINUMERIK* 840D/840Di/810D — Description of Functions **Tool Management**
- *SINUMERIK* 840D/810D — **Configuring Kit** HMI Embedded
- *SINUMERIK* 840D/810D — Description of Functions Operator Interface **OP 030**
- *SINUMERIK* 840D/810D — **MCIS** – Computer Link – Tool Data Information System – NC Data Management – NC Data Transfer – Tool Data Communication

Manufacturer/Service Documentation

- *SINUMERIK SIMODRIVE* — Description of Functions SINUMERIK **Safety Integrated**
- *SINUMERIK* 840D — Description of Functions **Digitizing**
- *SINUMERIK SIMODRIVE* — **Installation & Start-Up Guide *)** – 810D – 840D/611D – HMI
- *SINUMERIK SIMODRIVE* 840D/840Di 810D 611D — **– Lists *)** **– List Syst. Var.**
- *SINUMERIK SIMODRIVE* 840D 611D — Description of Functions **Linear Motor**
- *SINUMERIK SIMODRIVE* 840D 611D — Description of Functions **– Hydraulics Module** **– Analog Module**
- *MOTION CONTROL SYSTEMS* — **EMC** Guidelines

Electronic Documentation

- *SINUMERIK SIMODRIVE* 840D/840Di/810D 611, Motors — **DOC ON CD *)** The SINUMERIK System

Manufacturer/Service Documentation

- *SINUMERIK* 840D/840Di/810D — Description of Functions **ISO Dialects** for SINUMERIK
- *SINUMERIK* 840Di — **Manual** (HW + Installation and Start-Up)
- *SINUMERIK* 840D/810D — Description of Functions **Remote Diagnosis**
- *SINUMERIK* 840D/840Di/810D — **Manual @ Event**

***)** These documents are a minimum requirement

Siemens AG
Automation & Drives
Motion Control Systems
P. O. Box 3180, D – 91050 Erlangen
Germany

www.siemens.com/motioncontrol

Order No.: 6FC5 297-7AD40-0BP0

Printed in Germany