SIEMENS

SINUMERIK

**SINUMERIK
840D/840Di/810D
Basic logic functions: PLC Basic
program powerline (P3 pl)**

Function Manual

Valid for

Control
SINUMERIK 840D sl/840DE sl
SINUMERIK 840Di sl/840DiE sl
SINUMERIK 840D powerline/840DE powerline
SINUMERIK 840Di powerline/840DiE powerline
SINUMERIK 810D powerline/810DE powerline

Software	*Version*
NCU system software for 840D sl/840DE sl	1.3
NCU system software for 840D sl/DiE sl	1.0
NCU system software for 840D/840DE	7.4
NCU system software for 840Di/840DiE	3.3
NCU system software for 810D/810DE	7.4

Safety Guidelines

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

Danger

indicates that death or severe personal injury **will** result if proper precautions are not taken.

Warning

indicates that death or severe personal injury **may** result if proper precautions are not taken.

Caution

with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.

Caution

without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.

Notice

indicates that an unintended result or situation can occur if the corresponding information is not taken into account.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The device/system may only be set up and used in conjunction with this documentation. Commissioning and operation of a device/system may only be performed by **qualified personnel**. Within the context of the safety notes in this documentation qualified persons are defined as persons who are authorized to commission, ground and label devices, systems and circuits in accordance with established safety practices and standards.

Prescribed Usage

Note the following:

Warning

This device may only be used for the applications described in the catalog or the technical description and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens. Correct, reliable operation of the product requires proper transport, storage, positioning and assembly as well as careful operation and maintenance.

Trademarks

All names identified by ® are registered trademarks of the Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Table of contents

Brief description

<div style="text-align: right; font-size: 2em;">1</div>

General

The PLC basic program organizes the exchange of signals and data between the PLC user program and the NCK (Numerical Control Kernel), HMI (Human-Machine Interface) and MCP (Machine Control Panel) areas. A distinction is made between the following groups for signals and data:

- Cyclic signal exchange
- Eventdriven signal exchange
- Messages

Cyclic signal exchange

The cyclically-exchanged signals consist primarily of bit arrays.

- They contain **commands** transmitted from the PLC to the NCK (such as start or stop) and **status information** from the NCK (such as program running, interrupted, etc.).
- The bit fields are organized into signals for:
  - Mode groups
  - channels
  - Axes/spindles
  - General NCK signals

The cyclic exchange of data is performed by the basic program at the start of the PLC cycle (OB1). This ensures that the signals from the NCK remain constant throughout the cycle.

Event-driven signal exchange NCK → PLC

PLC functions that have to be executed as a function of the workpiece program are triggered by auxiliary functions in the workpiece program. If the auxiliary functions are used to start execution of a block, the type of auxiliary function determines whether the NCK has to wait before executing the function (e.g. during a tool change) or whether the function is executed in parallel to machining of the workpiece (e.g. for tool preparation on milling machines with chaintype magazines).

Data transfer must be as fast and yet as reliable as possible, in order to minimize the effect on the NC machining process. Data transfer is therefore controlled by alarms and acknowledgments. The basic program evaluates the signals and data, acknowledges this to the NCK and transfers the data to the application interface at the start of the cycle. If the data do not require user acknowledgment, this does not affect NC processing.

Event-driven signal exchange PLC → NCK

An "eventdriven signal exchange PLC → NCK" takes place whenever the PLC passes a request to the NCK (e.g., traversal of an auxiliary axis). In this case, the data transfer is also controlled by acknowledgment. When performed from the user program, this type of signal exchange is triggered using a function block (FB) or function call (FC).

The associated FBs (Function Blocks) and FCs (Function Calls) are supplied together with the basic program.

Messages

User messages are acquired and conditioned by the basic program. A defined bit field is used to transfer the message signals to the basic program. The signals are evaluated there and entered in the PLC diagnostics buffer on the occurrence of the message events. If a control unit is present, the messages are displayed on the operator interface.

Note

The function of the machine is largely determined by the PLC program. Every PLC program in the working memory can be edited with the programming device.

Detailed description

2

2.1 Key PLC CPU data for 810D, 840D and 840Di

The tables below show the performance range of the PLC CPUs and the scope of the basic PLC program relative to the various controller types.

Type of control: 810D and 840D

Key CPU data

	810D / 840D	810D / 840D	810D / 840D
PLC CPU	Integrated PLC CPU314	Integrated PLC CPU315-2DP	Integrated PLC CPU315-2DP master/slave
MLFB		6ES7 315-2AF00-0AB0	6ES7 315-2AF01-0AB0
Memory for user and basic program	64, 96, 128 KB	64, 96 128, 160, 192, 224, 256, 288 KB (dependent on option)	96, 160, 224, 288, 352, 416, 480 kByte (dependent on option)
Data block memory	Like user memory	Like user memory	Up to 96 KB
Memory submodule	no	no	no
Bit memories	2048	2048	2048/ 4096 with PLC operating system 03.10.13 or later
Timers	128	128	128
Counters	64	64	64
Clock memories	8	8	8
Program/data blocks OB FB FC DB	 1, 10, 20, 35, 40, 80-82, 85, 87, 100, 121-122 1-127 1-127 1-127	 1, 10, 20, 35, 40, 80-82, 85-87, 100, 121-122 1-127 1-127 1-127	 1, 10, 20, 35, 40, 80-82, 85-87, 100, 121-122 0-255 0-255 1-399
Max. data block length	16 KB	16 KB	16 KB
Max. block length FC, FB	16 KB	24 KB	24 KB
Inputs/outputs (address capacity) - digital - analog	 768 64	 1024/1024 64	 1024/1024 64

	810D / 840D	810D / 840D	810D / 840D
Inputs/outputs 1) (addressing)	Subrack 0 is not available for I/O devices:	Through optional configuring of I/O devices:	Through optional configuring of I/O devices:
- digital - analog	from I/O byte 32 onwards from PI/PO byte 384 onwards	from I/O byte 0 onwards from PI/PO byte 272 onwards	from I/O byte 0 onwards from PI/PO byte 272 onwards
Processing time - Bit commands (I/O) - Word commands	0.3 ms/kA 1-4 ms/kA	0.3 ms/kA 1-4 ms/kA	0.3 ms/kA 1-4 ms/kA
PDIAG (Alarm S,SQ)	no	no	yes
PROFIBUS	N/A	Master	Master/Slave
Number of PROFIBUS slaves		Min. 16, max. 64 SDB 2000 ≤ 8 KB	Min. 16, max. 64 SDB 2000 ≤ 32 KB
programmable block communication PBK	no	no	yes
Consistent Data to standard slave via SFC 14, 15	N/A	26	26
1) Subrack 0 is integrated in the NC. Subracks 1 to 3 are available for I/O devices.			

I/O expansion

	810D / 840D	810D / 840D	810D / 840D
PLC CPU	Integrated PLC CPU314	Integrated PLC CPU315-2DP	Integrated PLC CPU315-2DP master/slave
MLFB		6ES7 315-2AF00-0AB0	6ES7 315-2AF01-0AB0
I/O modules	24	24	24
PROFIBUS DP modules	N/A	yes	yes
Interfaces (MPI)	1	1	1

Types of control: 840Di, 810D and 840D

Key CPU data

	840Di	810D	840D
PLC CPU	Integrated PLC 315-2DP master/slave	Integrated PLC 315-2DP master/slave	Integrated PLC 314C-2DP master/slave
MLFB	6ES7 315-2AF03-0AB0	6ES7 315-2AF03-0AB0	6FC5 314-6CF00-0AB0
Memory for user and basic program	64, 96, 128, 160, 192, 224, 256 KB	64, 96, 128, 160, 192, 224, 288 KB	96, 160, 224, 352, 416, 480 KB (dependent on option)
Data block memory	Like user memory	Like user memory	Up to 96 KB
Memory submodule	no	no	no
Bit memories	4096	4096	4096
Timers	128	128	256
Counters	64	64	256

	840Di	810D	840D
Clock memories	8	8	8
Program/data blocks OB	1, 10, 20, 35, 40, 80-82, 85-87, 100, 121-122	1, 10, 20, 35, 40, 80-82, 85-87, 100, 121-122	1, 10, 20, 35, 40, 80-82, 85-87,100, 121-122
FB	0-255	0-255	0-255
FC	0-255	0-255	0-255
DB	1-399	1-399	1-399
Max. length of data block	16 KB	16 KB	16 KB
Max. block length FC, FB	24 KB	24 KB	24 KB
Inputs/outputs (address capacity) - digital - analog	 1024/1024 64	 1024/1024 64	 1024/1024 64
Inputs/outputs 1) (addressing) - digital - analog	Through optional configuring of I/O devices: from I/O byte 0 onwards from PI/PO byte 272 onwards (Profibus only)	Through optional configuring of I/O devices: from I/O byte 0 onwards from PI/PO byte 272 onwards	Through optional configuring of I/O devices: from I/O byte 0 onwards from PI/PO byte 272 onwards
Processing time - Bit commands (I/O) - Word commands	 0.3 ms/kA 1-4 ms/kA	 0.3 ms/kA 1-4 ms/kA	 0.1 ms/kA 0.25-1.2 ms/kA
PDIAG (Alarm S,SQ)	Yes	Yes	Yes
PROFIBUS	Master	Master/Slave	Master/Slave
Number of PROFIBUS slaves	Max. 64 SDB 2000 ≤ 32 KB	Max. 64 SDB 2000 ≤ 32 KB	Max. 32 SDB 2000 ≤ 32 KB
Max. number of PROFIBUS slots	256	256	256
programmable block communication PBK	Yes	Yes	Yes
Consistent Data to standard slave via SFC 14, 15	26	26	32

1) Subrack 0 is integrated in the NC. Subracks 1 to 3 are available for I/O devices.

I/O expansion

	840Di	810D	840D
PLC CPU	Integrated PLC 315-2DP master/slave	Integrated PLC 315-2DP master/slave	Integrated PLC 314C-2DP master/slave
MLFB	6ES7 315-2AF03-0AB0	6ES7 315-2AF03-0AB0	6FC5 314-6CF00-0AB0
I/O modules	PROFIBUS only	24	24
PROFIBUS DP modules	Yes	Yes	Yes
Interfaces (MPI)	1	1	1

Types of control: 840Di and 840D

Key CPU data

	840Di	840D
PLC CPU	Integrated PLC 317-2DP master/slave	Integrated PLC 317-2DP master/slave
MLFB	6FC5 317-2AJ10-0AB0	6FC5 317-2AJ10-1AB0
Memory for user and basic program	128 to 768 KB	128 to 768 KB
Data block memory	Max. 256 KB	Max. 256 KB
Memory submodule	no	no
Bit memories	32768	32768
Timers	512	512
Counters	512	512
Clock memories	8	8
Program/data blocks: OB FB FC DB	10, 20-21, 32-35, 40, 55-57, 80, 82, 85-87, 100, 121-122 0-2048 0-2048 1-2048	10, 20-21, 32-35, 40, 55-57, 80, 82, 85-87,100, 121-122 0-2048 0-2048 1-2048
Max. length of data block	32 KB	32 KB
Max. block length FC, FB	64 KB	64 KB
Inputs/outputs 1) (address capacity in bytes): - digital / - analog - incl. reserved area - process image Note: The inputs/outputs above 4096 are reserved for integrated drives.	4096/4096 8192/8192 256/256	4096/4096 8192/8192 256/256
Inputs/outputs 2) (addressing): - digital - analog	Through optional configuring of I/O devices: from I/O byte 0 onwards from PI/PO byte 272 onwards (Profibus only)	Through optional configuring of I/O devices: from I/O byte 0 onwards from PI/PO byte 272 onwards
Machining time: - Bit commands (I/O) - Word commands	≤ 0.03 ms/kA 0.1 ms/kA	≤ 0.03 ms/kA 0.1 ms/kA
PDIAG (Alarm S,SQ)	Yes	Yes
PROFIBUS	Master/Slave	Master/Slave
Number of PROFIBUS slaves	max. 125	max. 125
Max. number of PROFIBUS slots	512	512
DP master system no. DP	1	1

	840Di	840D
DP master system no. MPI/DP	2	N/A
programmable block communication PBK	Yes	Yes
Consistent Data to standard slave via SFC 14, 15	128	128
1) Notice!: The inputs/outputs above 4096 are reserved for integrated drives.		
2) Subrack 0 is integrated in the NC. Subracks 1 to 3 are available for I/O devices.		

I/O expansion

	840Di	840D
PLC CPU	Integrated PLC 317-2DP master/slave	Integrated PLC 317-2DP master/slave
MLFB	6FC5 317-2AJ10-0AB0	6FC5 317-2AJ10-1AB0
I/O modules	PROFIBUS only	24
PROFIBUS DP modules	1 (2)	1
Interfaces (MPI)	1 (0)	1

Note

Number of PROFIBUS slaves

Because SDB 2000 and other SDBs must be stored by the PLC operating system in the static RAM area, which the Profibus ASIC can also access, the information from SDB2000 can continue to be transferred to the NCK and the PLC basic program in conditioned form (CPI interface).

This is necessary for controlling the drives and PROFIsafe modules on Profibus. A memory area defined by the PLC is available for these data structures. Its size is limited by the maximum number of slots. This means that during loading, SDBs with fewer slaves than listed above may be rejected. A slot is usually a slave module or the slave itself. Only in a module with both I and Q areas does one module count as 2 slots.

It is, therefore, not possible to specify the size of SDB 2000 exactly.

It is only possible to say whether the configuration is permssible after the SDB container has been loaded into the CPU. The values specified in the tables mentioned above are therefore only intended as guidelines.

If the configuration is impermissible, a request for a general reset is issued when the SDBs are loaded. The cause of the configuring error can be found in the diagnostic buffer on completion of the general reset.

PLC versions

In SW 3.5 and higher on the 840D, version 6 (version code 35.06.03) is installed with PLC 314 and version 3 (version code 35.03.03) with PLC 315-2DP or higher.

These versions are compatible with the corresponding SIMATIC CPU300.
All modules and software packages approved by SIMATIC for these versions and CPUs are therefore suitable. Modules that can only be installed in subrack 0 are the exception (modules FM NC and FM 357 are also exceptions).

Version code: XX.YY.ZZ

- XX: SIMATIC CPU PLC version

- YY: Firmware transfer increment

- ZZ: Internal increment

Example

PLC 315-2DP with MLFB 6ES7 315-2AF00-0AB0:	04.02.14
PLC 315-2DP with MLFB 6ES7 315-2AF01-0AB0:	03.10.23
PLC 314:	07.02.12

HMI version display

The PLC module, the PLC operating system version and the module code appear in the last line of the HMI version display.

Example

PLC module	PLC operating system version	Module code
S7 PLC_315-2DP system	03.10.23	1200

Module codes

The table below shows the relationship between the module code and the corresponding PLC module, the suitable PLC operating system and its current software version:

Module code	PLC module	Suitable PLC operating systems (corresponding SIMATIC MLFB)	PLC operating system SW version
0208	PLC 314	6ES7 314–1AE0–0AB0	07.02.12
1008	PLC 3152DP with ASPC 2 Step C	6ES7 315–2AF00–0AB0	04.02.14
1100	PLC 3152DP with ASPC 2 Step D	6ES7 315–2AF01–0AB0	03.10.23
1200	PLC 3152DP with ASPC 2 Step E	6ES7 315–2AF01–0AB0	03.10.23
		6ES7 315–2AF03–0AB0 FW1.2	12.30.10
1400	PLC 314C2DP with IBC 16	6ES7 314–6CF00–0AB0 FW1.0.2	10.60.20
2200	PLC 317-2DP with IBC 32	6ES7 317–2AJ10–0AB0 FW2.1	20.71.15
MCI 1 (840Di)	PLC 3152DP with ASPC 2 Step E	6ES7 315–2AF03–0AB0 FW1.0	04.20.36
MCI 2 (840Di) 2100	PLC 317-2DP with IBC32	6ES7 317–2AJ10–0AB0 FW2.1	20.70.17

810 D, 840D

The tables below show the key data of the OPI interface and the PLC basic program functionality with reference to SINUMERIK 810D, 840D and 840Di:

OPI interface

	840Di	810D	840D
Number	N/A	N/A	1

PLC basic program functions

	840Di	810D	840D
Axes/spindles	[1]	5	31
channels	[1]	2	10
Mode groups	[1]	1	10
Status/control signals	+	+	+
M decoders (M00-99)	+	+	+
G group decoders	+	+	+
Aux. function distributors	+	+	+
Interrupt-driven output of auxiliary functions	+	+	+
Move axes/spindles from PLC	+	+	+
Async. subprogram interface	-	-	-
Error/operating messages	+	+	+
MCP and handheld unit signals via NCK	+	+	+
Reading/writing of NC variables	+	+	+
PI services	+	+	+
Tool management	+	+	+
Star/delta switchover	+	+	+
Display control handheld unit	+	+	+

[1] Depends on chosen system software package

2.2 Reserve resources (timers, FC, FB, DB, I/O)

Reserved components

The components below are reserved for the basic program:

Component	Reserved range
Timers	T0 - T9
Functions (general)	FC 0 - FC 29
Functions (in ShopMill/ShopTurn)	FC 0 - FC 35
Function blocks	FB 0 - FB 29
Data blocks (general)[1]	DB 1 - DB 62; DB 71 - DB 80
Data blocks (in ShopMill/ShopTurn) 1)	DB 1 - DB 62; DB 71 - DB 89

1) The data blocks for channels, axes/spindles and tool management functions that are not activated may be assigned as required by the user.

PLC 317-2DP

PLC CPU: PLC 317-2DP are reserved for further number bands for SIEMENS applications referring to FC, FB, DB and I/O areas.

FC, FB and DB

Component	Reserved range
Functions	FC 1000 - FC 1023
Function blocks	FB 1000 - FB 1023
Data blocks	DB 1000 - DB 1099

I/O range

Component	Reserved range
Address area	256 - 271 [1]
Inputs/outputs	4096 upwards[2]

1) Reserved for the NC module and future expansions

2) Reserved for integrated drives However, diagnostics addresses for modules can only be placed in the uppermost address range, as suggested by STEP7.

2.3 Starting up hardware configuration of PLC CPUs

General procedure

STEP 7 is used to define the hardware configuration for a PLC CPU, including the associated I/O.

The procedure to be followed is shown below:

1. Load tool box to PG/PC
2. Create a new project (File, New, Project)
3. Insert, Hardware, SIMATIC 300 station
4. Select SIMATIC 300 station1 with mouse
5. Open object by right-clicking with the mouse to start the HWConfig
6. Destination system, load to PG, the hardware equipment complement is read back from the central system
7. Configure distributed I/Os
8. Insert PLC basic program

The addresses for the I/O modules can be changed if necessary (permissible only on certain PLC CPUs, e.g. PLC 3152DP).

As an alternative, the entire hardware configuration can be entered manually (see also appropriate STEP7 documentation). The notices below must be observed.

STEP7, Version 3

With STEP7 Version 3 and higher, the hardware configuration of the SINUMERIK components must be defined via the entries in SIMATIC\RACK 300. The install or setup program of the basic program on the tool box diskettes is required for this purpose.

STEP7 Version 5.1 SP2 and Toolbox 6.03.02

With STEP7 V5.1 SP2 and Toolbox 6.03.02 or later, the SINUMERIK components are stored under SIMATIC 300\SINUMERIK. The current hardware expansion for STEP 7 can also be found under eSupport.
Current path (02/13/2004): sinumerik_software > 840d/810d/fm-nc > patches & fixes > plc > Hardware_fuer_STEP7 > Hardware upto NCU*.5/CCU3/840Di_MCI2 > V6.5.2.0

NCU	MLFB	Comparable SIMATIC CPU MLFB included	Selection from STEP7 hardware catalog
CCU1 810D CPU	6FC5 410-0AA00-0AA0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
CCU2 810D CPU	6FC5 410-0AA01-0AA0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
CCU1 810DE CPU	6FC5 410-0AY01-0AA0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
CCU2 810D CPU	6FC5 410-0AX02-0AA0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 810DE Light CCU1 module with system software (export)	6FC5 410-0AY00-0AA0	6ES7 314-1AE01-0AB0	810D/840D with PLC314

NCU	MLFB	Comparable SIMATIC CPU MLFB included	Selection from STEP7 hardware catalog
SINUMERIK 810D CCU2 module with system software (standard)	6FC5 410-0AX02-1AA0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840DE NCU 561.2 without system software	6FC5 356-0BB11-0AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 571 (export version)	6FC5 357-0BA10-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 571 (export version) with PROFIBUS DP	6FC5 357-0BA11-0AE0	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 571.2 (export version) with PROFIBUS DP	6FC5 357-0BA11-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840DE NCU 571.2 without system software	6FC5 357-0BB11-0AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 572	6FC5 357-0BA20-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 572	6FC5 357-0BA20-1AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 572	6FC5 357-0BA21-0AE0	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 572	6FC5 357-0BA21-1AE0	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 572.2 (export version) with PROFIBUS DP	6FC5 357-0BA21-1AE1	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D/DE NCU 572.2 without system software	6FC5 357-0BB21-0AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D/DE NCU 572.3 without system software	6FC5 357-0BB22-0AE0	6ES7 315-2AF01-0AB0	10D/840D with PLC315-2AF01
SINUMERIK 840D NCU 572 (export version)	6FC5 357-0BY20-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 572 (export version)	6FC5 357-0BY20-1AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 572 (export version) with PROFIBUS DP	6FC5 357-0BY21-0AE0	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 572 (export version) with PROFIBUS DP	6FC5 357-0BY21-1AE0	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 572.2 (export version) with PROFIBUS DP	6FC5 357-0BY21-1AE1	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 572 with digitizing	6FC5 357-0BA24-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 572.2 with digitizing and PROFIBUS DP	6FC5 357-0BA24-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01

NCU	MLFB	Comparable SIMATIC CPU MLFB included	Selection from STEP7 hardware catalog
SINUMERIK 840D/DE NCU 572.2 without system software	6FC5 357-0BB24-0AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 572 (export version) with digitizing	6FC5 357-0BY24-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 572.2 (export version) with digitizing and PROFIBUS DP	6FC5 357-0BY24-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 573	6FC5 357-0BA30-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840DE NCU 573 (export version)	6FC5 357-0BY30-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 573 with digitizing	6FC5 357-0BA31-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 573 (export version) with digitizing	6FC5 357-0BY31-0AE0	6ES7 314-1AE01-0AB0	810D/840D with PLC314
SINUMERIK 840D NCU 573 with PROFIBUS DP	6FC5 357-0BA32-0AE1	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 573 (export version) with PROFIBUS DP	6FC5 357-0BY32-0AE1	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 573 with PROFIBUS DP	6FC5 357-0BA33-0AE0	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 573 (export version) with PROFIBUS DP	6FC5 357-0BY33-0AE0	6ES7 315-2AF00-0AB0	840D with PLC3152AF00
SINUMERIK 840D NCU 573.2 (Pentium Pro) up to 12 axes with PROFIBUS DP	6FC5 357-0BA32-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 573.2 (Pentium Pro) up to 31 axes with PROFIBUS DP	6FC5 357-0BA33-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D/DE NCU 573.2 without system software	6FC5 357-0BB33-0AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D/DE NCU 573.2 Pentium II without system software	6FC5 357-0BB33-0AE1	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 573.2 (Pentium Pro) for digitizing with PROFIBUS DP	6FC5 357-0BA31-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D/DE NCU 573.2 without system software	6FC5 357-0BB31-0AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01

NCU	MLFB	Comparable SIMATIC CPU MLFB included	Selection from STEP7 hardware catalog
SINUMERIK 840D NCU 573.2 (Pentium Pro) (export version) for 12 axes with PROFIBUS DP	6FC5 357-0BY32-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 573.2 (Pentium Pro) (export version) for 31 axes with PROFIBUS DP	6FC5 357-0BY33-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 573.2 (Pentium Pro) (export version) for digitizing with PROFIBUS DP	6FC5 357-0BY31-1AE0	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840Di	6FC5 220-0AA00-1AA0	6ES7 315-2AF03-0AB0	810D/810Di with PLC315-2AF03
SINUMERIK 840Di with PK bus		6ES7 315-2AF03-0AB0	810D/810Di with PLC315-2AF03, PK bus
SINUMERIK 840D NCU 572.3	6FC5 357-0BB22-0AE0	with operating system 03.10.23: 6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 572.3	6FC5 357-0BB22-0AE0	with operating system 12.30.07: 6ES7 315-2AF03-0AB0	810D/840D with PLC315-2AF03
SINUMERIK 840D NCU 573.3	6FC5 357-0BB33-0AE2	6ES7 315-2AF01-0AB0	810D/840D with PLC3152AF01
SINUMERIK 840D NCU 573.3	6FC5 357-0BB33-0AE2	with operating system 12.30.07: 6ES7 315-2AF03-0AB0	810D/840D with PLC315-2AF03 (as of STEP7 V5.0 and Toolbox 05.03.05)
SINUMERIK 840D NCU 572.4	6FC5 357-0BB23-0AE0	6ES7 314-6CF00-0AB0	810D/840D with PLC314C-2DP
SINUMERIK 840D NCU 573.4	6FC5 357-0BB34-0AE1	6ES7 314-6CF00-0AB0	810D/840D with PLC314C-2DP
SINUMERIK 810D CCU3	6FC5 410-0AY03-0AA0	6ES7 315-2AF03-0AB0	810D/840D with PLC315-2AF03
SINUMERIK 840D NCU 571.3	6FC5 357-0BB11-0AE1	6ES7 315-2AF03-0AB0	810D/840D with PLC315-2AF03
SINUMERIK 840D NCU 561.3	6FC5356-0BB11-0AE1	6ES7 315-2AF03-0AB0	810D/840D with PLC315-2AF03 (as of STEP7 V5.0 and Toolbox 05.03.05)
SINUMERIK 840D NCU 571.4	6FC5357-0BB12-0AE0	6ES7 314-6CF00-0AB0	810D/840D with PLC314C-2 DP (STEP7 V5.1 SP3 and higher and Toolbox 06.03.02)
SINUMERIK 840D NCU 561.4	6FC5356-0BB12-0AE0	6ES7 314-6CF00-0AB0	810D/840D with PLC314C-2 DP (STEP7 V5.1 SP3 and higher and Toolbox 06.03.02)
SINUMERIK 840D NCU 573.5	6FC5357-0BB35-0A E0	6ES7 317-2AJ00-0AB0	810D/840D with PLC317-2 DP (STEP7 V5.2 SP1 and higher and Toolbox 06.05.02)
SINUMERIK 840Di with MCI2	6FC5 222-0AA02-1AA0	6ES7 317-2AJ00-0AB0	810Di with PLC317-2 DP (STEP7 V5.2 SP1 and higher and Toolbox 06.05.02)

Note

On the SINUMERIK 810D or 840D, SIMATIC subrack 0 is integrated in the NC. The following components are plugged into this subrack:

- Slot 2: The integrated PLC (PLC 314 or PLC 315-2DP)

- Slot 3: An IM 360

- Slot 4: The FM NCU.
With PLC 314, NC software version 3.5 and higher, the FM NCU must also be defined if further MPI (C bus) devices are included in subrack 1 to subrack 3 (e.g., FM modules with C bus connection). The properties of the FM NCU must not be changed, as process interrupts (e.g. auxiliary functions) of the NCU may, in this case, no longer function.
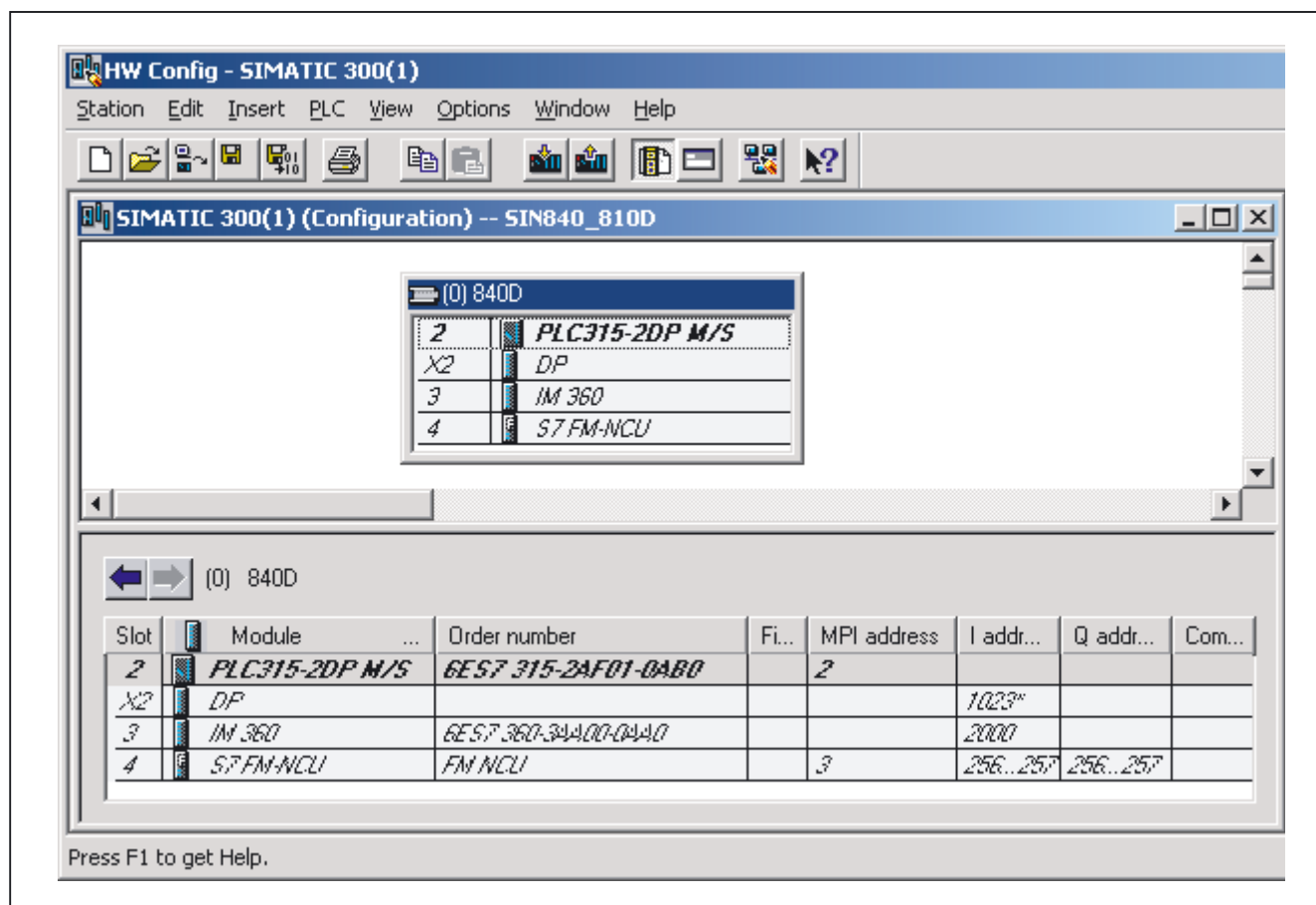
Figure 2-1   Hardware configuration of row 0 on 810D, 840D

MCP (Machine Control Panel) and HHU (HandHeld Unit)

(only for SINUMERIK 810D to SW 3.x)

If the MCP or HHU is configured (deviation from the norm), an additional SIMATIC 300 station must be inserted into the machine project for each operator component. Any type of CPU must be inserted in location 2 on row 0 in this station by means of the hardware configuration (HW config.). The MPI address of the operator component must be set as the MPI address. MPI network (1) can then be marked in the SIMATIC manager. The global data can then be activated via the "Tools" menu item.
The rest of the procedure is described in detail in the Commissioning Manual.

2.4 Starting up the PLC program

2.4.1 Installing the basic program for 810D, 840D

Before initial startup of the NC, an NC and PLC general reset must be carried out to initialize the relevant memory areas. To do this, set switch S3 to setting 1 and switch S4 to setting 3 and switch the control off and on again (POWER ON reset).

Installation

With Toolbox SW 6.1 and higher, the installation is performed by a WINDOWS-compliant setup program for the basic program, hardware selection in STEP7 (SINUMERIK 810D/840D option package) and the NC-Var Selector. To start the installation, run "setup.exe" in the main CD directory. You can then choose which components to install. After the installation, you can select the basic program library directly from STEP 7 (gp8x0d61, 61 is the main basic program version).

The concrete version of the basic program can be scanned for the object properties of the library or the program folder in the comment field.

From software version 3.7 to version 4.2, installation is carried out using INSTALL.BAT (INSTALL1.BAT, INSTALL4.BAT).
This program installs the basic program and additional files for the relevant STEP7 version. With automatic installation in STEP 7 V3 and higher, the TYP, GSD and meta files in the hardware catalog are also augmented and updated.
The hardware components of the SINUMERIK system are then also available for hardware configuration under STEP 7. This eliminates the need for unzipping, mentioned below.
The basic program is supplied in zipped format as a **project** for STEP 7 Version 1.x or as a **library** for STEP 7 Version 2.x and subsequent versions.

General

The OB source programs, including standard parameterization, interface symbols and DB templates for handheld unit and M decoding functions are enclosed in the SIMATIC project or SIMATIC library of the basic program.

STEP 7 must be installed before the basic program.

STEP7 V1.x

The basic program is stored as a compressed file with the name GP840D.EXE (or GP810D.EXE and GPFMNC.EXE) in the main directory on the diskette. The basic program (GP840D.exe) must be copied to the main directory (root) of a drive (e.g., c:\) and then called. The project structure required for the basic program is generated automatically. The catalog name of the basic program is GP840Dxy.S7A. In this case, xy stands for the basic program version.

Note

When STEP 7 V1 is used, the GP840Dxy.S7A catalog must be copied to the root directory. Any existing catalog with the same name GP840Dxy.S7A must be deleted beforehand.

STEP7 V2.x, 3.x

The basic program is stored as a compressed file with the name "GP840D.EXE" in directory S7V2.840 or S7V2.810 or S7V2 on the basic program diskette. The basic program (GP840D.exe) must be copied to subcatalog "S7LIBS" of STEP7 V2 (step7_v2) or subsequent versions thereof and then called. The library structure required for the basic program is generated automatically. The catalog name of the basic program is GP840Dxy. In this case, xy stands for the basic program version. The file "MET.EXE" must be copied to the basic catalog of STEP 7 and called from there via the DOS window with "MET.EXE –O".

Note

The name GP840D specified above refers to the basic program of the SINUMERIK 840D. The basic program is named GP810D on the SINUMERIK 810D and GPFMNC on the FM-NC. With effect from SW 4.2, the basic program for 810D and 840D is combined. It is now called GP8x0D.

2.4.2 Application of basic program

A new CPU program (e.g., "Turnma1") must be set up in a project by means of the STEP7 software for each installation (machine).

Comment

The catalog structures of a project and the procedure for creating projects and user programs are described in the relevant SIMATIC documentation.

STEP7 V1

A network link to the PLC must be activated for the machine CPU program under menu items "Edit", "Configuration". This is done in the "Services", "Parameterize" menu followed by selection of the MPI parameters.

Default:

- **"Networked"**
- "MPI subnetwork number = 0"
- "CPU MPI Addr = 2"

The following must be copied into the CPU program for the machinespecific program files:

1. The basic-program blocks (FCs, FBs, DBs, OBs, SFCs, SFBs, and UDTs) ("File", "Manage Project" menu in the Step7 program editor).

2. File GPOB840D.AWL (or GPOB810D.AWL or GPOBFMNC.AWL) and other STL (AWL) files if appropriate must also be copied from the basic program catalog into the CPU program. The OBs contained in this file are the basis for the user program with the associated basic program calls. Existing user blocks must be copied as STL files to the newly created CPU program (catalog name CPU1.S7D) and compiled.

3. We also recommend that the symbolic names are transferred with the files from the basic program package using the symbol editor.

STEP7 V2

The basic-program blocks are copied using the SIMATIC Manager and "File"/"Open"/"Library".

The following sections must be copied from the library:
AP off: FCs, FBs, DBs, OBs, SFC, SFB, UDT and the SDB container.

For the 810D, the SDB container includes:

- The source_files (SO):
  - GPOB810D or GPOB840D
  - Possibly MDECLIST, HHU_DB and others
- Symbol table (SY)

Note

The SDB container is only included for these control variants.

Compatibility with STEP 7

No interdependencies exist between the basic program (including older program versions) and currently valid versions of STEP 7.

2.4.3 Version codes

Basic program

The version of the basic program is displayed on the version screen of the HMI software along with the controller type.

User Program

The user can define his/her own version code for the PLC user program in any data block in the form of a data string containing a maximum of 54 characters. The data can contain a text of the user's choice. Parameter assignments for this string are made via a pointer in FB 1. Parameterization requires symbolic definition of the data block.

References::
/FB1/ Function Manual, Basic Functions; PLC – Basic Program (P3);
Chapter: Block Description

2.4.4 Machine program

The machine manufacturer creates the machine program using the library routines supplied with the basic program. The machine program contains the logic operations and sequences on the machine. The interface signals to the NC are also controlled in this program. More complex communication functions with the NC (e.g., read/write NC data, tool-management acknowledgments, etc., are activated and executed via basic-program FCs and FBs). The machine program can be created in different creation languages (e.g., STL, LAD, CSF, S7 HIGRAPH, S7GRAPH, SCL). The complete machine program must be generated and compiled in the correct sequence. This means that blocks that are called by other blocks must generally be compiled before the blocks, which call them. If blocks that are called by other blocks are subsequently modified in the interface (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR) as the program is developed, then the call block and all blocks associated with it must be recompiled. This general procedure applies analogously to instance data blocks for FBs. If this sequence of operations is not observed, timestamp conflicts occur when the data are retranslated into STEP7. In some cases, therefore, it may not be possible to recompile blocks, creating problems, for example, with the "Block status" function. It is moreover advisable to generate blocks in ASCIISTL by means of the STEP7 editor when they have been created in the ladder diagram or in single statements (incremental mode).

2.4.5 Data backup

The PLC-CPU does not save any symbolic names, but instead data type descriptions of the block parameters:

- VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR and

- the data types of the global data blocks.

Note

No sensible recompilation is possible without the related project for this machine. This especially affects, for instance the function status of the block or the necessary changes done in the PLC-CPU programs later. It is, therefore, necessary to keep a backup copy of the STEP7 project located in the PLC CPU on the machine. This is a great help for the service case and saves unnecessary consumption of time in restoring the original project.

If the STEP7 project exists and has been created according to the instructions given above, then symbols can be processed in the PLCCPU on this machine. It may also be advisable to store the machine source programs as AWL files in case they are required for any future upgrade.

The source programs of all organization blocks and all instance data blocks should always be available.

2.4.6 PLC series startup, PLC archives:

Generate series archive

Once the blocks have been loaded into the PLC CPU, a series archive can be generated via the HMI operator interface for data backup on the machine. To ensure data consistency, this backup must be created immediately after block loading when the PLC is in the Stop state. It does not replace the SIMATIC project backup as the series archive saves binary data only, and does not back up, e.g., symbolic information. In addition, no CPU DBs (SFC 22 DBs) or SDBs generated in the CPU are saved.

In Toolbox 06.03.03 and STEP 7 V5.1 and higher, the PLC series archive can be generated directly from the corresponding SIMATIC project.
To do this, select the "Options" → "Settings" menu item and the "Archive" tab in STEP 7. This contains an entry "SINUMERIK (*.arc)", which must be selected to create a series startup file. After selection of the archive, select the "File" → "Archive" menu item. The relevant series archive will then be generated. If the project contains several programs, the program path can be selected. A series archive is set up for the selected program path. All blocks contained in the program path are incorporated into the archive, except for CPU-DBs (SFC 22 DBs).

The process of generating a series archive can be automated (comparable to the command interface in STEP 7, V5.1 and higher). In generating this series archive, the command interface is expanded.

Functions

The following functions are available for this expansion:

The functions (shown here in VB script) are not available until server instantiations and Magic have been called:

Functions:

Const S7BlockContainer = 1138689, S7PlanContainer = 17829889	
Const S7SourceContainer = 1122308	
set S7 = CreateObject("Simatic.Simatic.1")	
rem Instance command interface of STEP7	
Set S7Ext = CreateObject("SimaticExt.S7ContainerExt")	
Call S7Ext.Magic("")	

Function **Magic**(bstrVal As String) As Long

Call gives access to certain functions. The function must be called once after server instantiation. The value of bstrVal can be empty. This initiates a check of the correct Step7 version and path name in Autoexec. The functions are enabled with a return parameter of 0.

Return parameter (-1) = incorrect STEP 7 version

Return parameter (-2) = no entry in Autoexec.bat

Function **Magic**(bstrVal As String) As Long

Function **MakeSeriesStartUp**(FileName As String, Option As Long, Container As S7Container) As Long

"Option" parameter:

0:	Normal series startup file with general reset
Bit 0 = 1:	Series startup file without general reset. When project contains SDBs, this option is inoperative. A general reset is then always executed.
Bit 1 = 1:	Series start-up file with PLC restart

Return parameter value:

0	= OK
-1	= Function unavailable, call Magic function beforehand
-2	= File name cannot be generated
-4	= Container parameter invalid or container block empty
-5	= Internal error (memory request rejected by Windows)
-6	= Internal error (problem in STEP 7 project)
-7	= Write error when generating series startup files (e.g., diskette full)

Use in script

```
If S7Ext.Magic("") < 0 Then
     Wscript.Quit(1)
End If
     Set Proj1 = s7.Projects("new")
     set S7Prog = Nothing
Set s7prog = Proj1.Programs.Item(1) 'if there is only one program'
For i = 1 to S7Prog.Next.Count
          Set Cont = S7Prog.Next.Item(i)
     Check block container
          If (Cont.ConcreteType = S7BlockContainer) Then
                Exit For
          End if
Next
Error = S7Ext.MakeSerienIB("f:\dh\arc.dir\PLC.arc", 0, Cont)'Error analysis now'
```

2.4.7   Software upgrades

Software upgrade

Whenever you update the PLC or NCK software, always reset the PLC to its initial state first. This initial clear state can be achieved by means of a general PLC reset. All existing blocks are cleared when the PLC is reset.

It is usually necessary to include the new basic program when a new NC software version is installed. The basic programs blocks must be loaded into the user project for this purpose. OB 1, OB 40, OB 100, FC 12 and DB 4 should not be loaded if these blocks are already included in the user project. These blocks may have been modified by the user. The new basic program must be linked with the user program.

To do this, proceed as follows:

1. Generate the text or source file of all user blocks before copying the basic program.

2. Then copy the new basic program blocks to this machine project (for a description, see Subsection "Application of basic program")

3. All user programs *.awl must then be recompiled in the correct order! (See also the "Machine program" section.)
   This newly compiled machine program must then be loaded to the PLC CPU using STEP 7.

However, it is normally sufficient to recompile the organization blocks (OB) and the instance data blocks of the machine program. This means you only need to generate sources for the organization blocks and the instance data blocks (before upgrading).

Overall reset

A description of how to perform a general PLC reset appears in the Installation and Startup Guide. However, a general reset does not delete the contents of the diagnostic buffer nor the node address on the MPI bus. Another possible general reset method is described below. This method must be used when the normal general reset process does not work.

Proceed as follows:

No.	Action	Effect
1	Control system is switched off	
2	PLC switch setting 3 (MRES) and switch control on again or perform hardware reset.	LED labeled PS flashes slowly.
3	Set PLC startup switch to position 2 (STOP) and back to position 3 (MRES).	The LED labeled PS starts to flash faster.
4	Set PLC startup switch to setting 2 or 0.	

NC variables

The latest NC VAR selector can be used for each NC software version (even earlier versions). The variables can also be selected from the latest list for earlier NC software versions. The data content in DB 120 (default DB for variables) does not depend on the software version, i.e., selected variables in an older software version must not be reselected when the software is upgraded.

2.4.8   I/O modules (FM, CP modules)

Special packages for STEP7 are generally required for more complex I/O modules. Some of these special packages include support blocks (FC, FB) stored in a STEP7 library. The blocks contain functions for operating the relevant module which are parameterized and called by the user program. In many cases, the FC numbers of the CP and FM module handling blocks are also included in the number range of the basic program for the 810D and 840D systems.
What action can be taken if such a conflict occurs?
The block numbers of the basic program must remain unchanged. The block numbers of handling blocks can be assigned new, free numbers using STEP7. These new blocks (with new FC numbers) are then called in the user program with the parameter assignments required by the function.

2.4.9   Troubleshooting

This section describes problems which may occur, their causes and remedies and should be read carefully before hardware is replaced.

Errors, cause/description and remedy			
Serial no. error information	Errors	Cause/description	To correct or avoid errors
1	No connection via MPI to PLC.	The MPI cable is not plugged in or is defective. Possibly, the STEP 7 software is also not correctly configured for the MPI card.	Test: Create a link with the programmer in the STEP7 editor by means of connection "Direct_PLC". A number of node addresses must be displayed here. If they do not appear, the MPI cable is defective/not plugged in.
2	PLC cannot be accessed in spite of PLC general reset.	A system data block SDB 0 has been loaded with a modified MPI address. This has caused an MPI bus conflict due to dual assignment of addresses.	Disconnect all MPI cables to other components. Create the link "Direct_PLC" with the programmer. Correct the MPI address.
3	All four LEDs on the PLC flash (DI disaster)	A system error has occurred in the PLC. **Measures:** The diagnostic buffer on the PLC must be read to analyze the system error in detail. To access the buffer, the PLC must be stopped (e.g., set "PLC" switch to position 2). A hardware reset must then be performed. The diagnostic buffer can then be read out with STEP7. Relay the information from the diagnostic buffer to the Hotline / Development Service. A general reset must be carried out if requested after the hardware RESET. The diagnostic buffer can then be read with the PLC in the Stop state.	Once the PLC program has been RESET or reloaded, the system may return to normal operation. Even in this case, the content of the diagnostic buffer should be sent to the Development Office.

2.5 Linking PLC CPUs to 810D, 840D

2.5.1 General

The AS 300 family is used as the PLC for all systems. The essential difference between the NCU variants lies in the method by which they are linked. On the 840D and 810D, the PLC 314 CPU (user memory capacity up to 128 KB) or PLC 315-2 DP (user memory capacity up to 288 KB) is integrated as a submodule into the NC unit.
The PLC/CPU 315-2 DP also supports distributed I/Os on the PROFIBUS (L2DP). The relevant performance data for individual PLC CPUs can be found in the above table or in Catalog FT70.

2.5.2 Properties of PLC CPUs

SINUMERIK 810D/840D/840Di PLC CPUs are based on standard SIMATIC CPUs in the S7-300 family. As a result, they generally possess the same functions. Functional deviations are shown in the table above. Owing to differences in their memory system as compared to the S7 CPU, certain functions are not available (e.g., blocks on memory card, project on memory card).

Note

With the current SIMATIC CPUs, the PLC is not automatically started after voltage failure and recovery when a PLC Stop is initiated via software operation. In this instance, the PLC remains in the Stop state with an appropriate diagnostic entry for safety reasons. You can start the PLC only via software operation "Execute a restart" or by setting the switch to "Stop" and then "RUN". This behavior is also integrated in the current versions of the SINUMERIK PLC.

2.5.3 Interface on 810D and 840D with integrated PLC

Physical interfaces

As the 810D and 840D systems have an integrated PLC, signals can be exchanged between the NCK and PLC directly via a dualport RAM.

Exchange with operator panel and MCP

Data are generally exchanged with the operator panel (OP), machine control panel (MCP) and handheld unit (HHU) on the 840D via the operator panel interface (OPI), the COM module being responsible for data transport.

All devices specified above can also be operated on the multipoint interface (MPI) in the case of the 840 D. With the 810D, data communication with the operator panel (OP), machine control panel (MCP) and handheld unit (HHU) takes place only via the MPI.

The programming device is connected directly to the PLC via the MPI (multipoint interface).
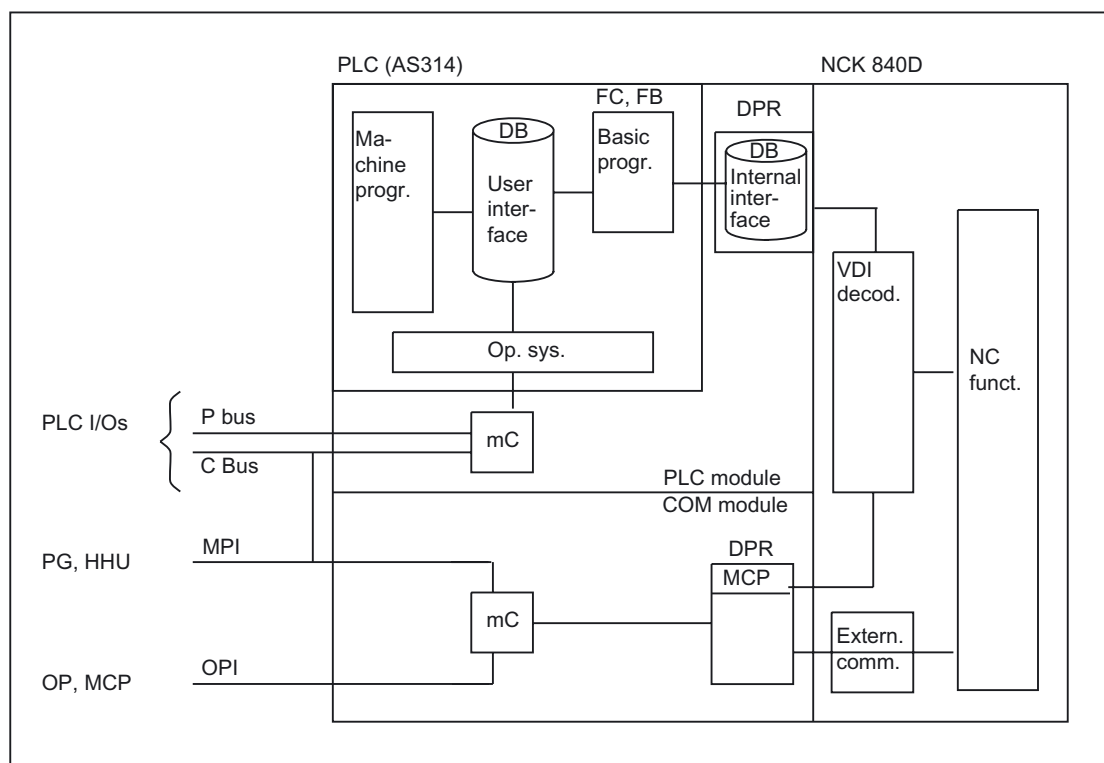
Figure 2-2   NCK/PLC connection on 810D, 840D (integrated PLC)

NCK/PLC interface

The data exchange NCK/PLC is organized on the PLC-side by the basic program. The **Status information** stored by the NC in the internal Dual–Port–RAM (DPR) (such as "Program is running") are copied by the basic program in the data blocks at the start of the cycle (OB 1), which the user can then access (user interface). The user also enters NC **control signals** (e.g. NC start) in the interface data blocks, and these are also transferred to the NC at the start of the cycle.

Auxiliary functions transferred to the PLC dependent on the workpiece program are first evaluated by the basic program (alarmdriven) and then transferred to the user interface at the start of OB1. If the relevant NC block contains auxiliary functions that require the interruption of the NC machining process (e.g. M06 for tool change), the basic program halts the execution of the block on the NC for one PLC cycle. The user can then use the "read disable" interface signal to halt the block execution until the tool change has been completed. If, on the other hand, the relevant NC block does not contain auxiliary functions requiring the interruption of the NC machining process (e.g. M08 for "Cooling on"), the transfer of these "rapid" auxiliary functions is enabled directly in OB 40, so that block execution is only marginally influenced by the transfer to the PLC.

The evaluation and enabling of the **G functions** transferred from the NCK are also alarm-driven, however they are transferred directly to the user interface. Where a G function is evaluated at several points in the PLC program, differences in the information of the G function within one PLC cycle may arise.

In the case of **NC actions** triggered and assigned with parameters by the PLC (e.g. traverse concurrent axes), triggering and parameter assignment is performed using FCs and FBs, not interface data blocks. The FCs and FBs belonging to the actions are supplied together with the basic program. The FCs must be loaded by the user and called in the PLC program of the machine manufacturer (machine program). You can find an overview of the FC, FB and DB blocks, grouped according to their basic and extended functions, in:
References::
/FB1/Function Manual, Basic Functions; PLC Basic Program (P3);
"PLC program commissioning"

OP/PLC interface

Data are exchanged between the OP and PLC via the OP/NC serial bus, COM module and C bus. The COM module transfers the data intact from one bus segment to another. It merely converts the baud rate. The OP is always the active partner (client) and the PLC is always the passive partner (server). Data transmitted or requested by the OP are read from and written to the OP/PLC interface area by the PLC operating system (timing: cycle control point). From the viewpoint of the PLC application, the data are identical to I/O signals.

MCP/PLC interface, HHU/PLC interface (840D only)

Data are exchanged between MCP/PLC and HHU/PLC via the serial bus MCP, HHU/NC, COM module, and NCK. The NCK transfers the MCP/HHU signals to and fetches them from the internal NC/NCK DPR (dual-port RAM). On the PLC side, the basic program handles communication with the user interface. The basic program parameters define the operand areas (e.g. I/O) and the start addresses.

MCP/PLC interface, HHU/PLC interface (810D only)

Data exchange between MCP/PLC and HHU/PLC takes place via the MPI interface on the PLC. The Communication with global data (GD)[1] service is used for this purpose (see also STEP7 User's Guide). The PLC operating system handles the transfer of signals from and to the user interface. The STEP7 **Communication configuration** configuring tool is used to define both GD parameters as well as operand areas (e.g., I/O) and their initial addresses. In SW 2.2 and higher, data exchange is possible as on the 840D.

[1] IC (GD) = Implicit Communication (Global Data)

2.5.4 Diagnostic buffer on PLC

General

The diagnostic buffer on the PLC, which can be read out using STEP 7, displays diagnostic information about the PLC operating system. Moreover, through the basic program and the function: "Alarms / messages" are made available via the FC10 entries in the diagnostic buffer. These alarms and messages are displayed in the diagnostic buffer as event with the "Event-ID": <ID>" without explanatory text.

Meaning of displayed data

An example is presented below to illustrate what information is displayed for an event and how to interpret this information.

Diagnostic buffer (extract, relevant data selected):

Event details: 1 of 10	Event ID: 16# B046
No entry in text database. Hex values are displayed.	
Event ID:	16# **B046**
OB:	16# 01
PK:	16# 01
DatID 1/ 2:	16# 59 C9
Supplementary info1 / 2 / 3:	16# 02**00** 0000 00**20**
Outgoing event:	36:02:459 08.04.03

Type of event
The two most significant digits of Event ID 16# **B0**46 contain the code for the event type:

A1:   Alarm set
A0:   Alarm cancelled
B1:   Message set
B0:   Message cancelled

Alarm/Message Number

The two least significant digits of Event ID 16# B0**46** contain the code for the most significant decimal places of the alarm or message number. The two least significant digits of supplementary information 1 / 2 / 3 must also be taken into account when determining the complete alarm or message number:

- Supplementary information 1 16# 02**00** 0000 0020

- Supplementary information 2 16# 0200 0000 00**20**

All values must be converted to decimal values and lined up in order of alarm or message number:

Event ID:	46 $_{hex}$	=	70 $_{dec}$
Supplementary information 1	00 $_{hex}$	=	00 $_{dec}$
Supplementary information 2	20 $_{hex}$	=	32 $_{dec}$
Alarm number (BM) [1])		=	**70 00 32**

1) Controlled by DB2.DBX184.0
BM = operational message

As such, the event with the event-ID: 16# B046 and the mentioned additional information: means "Message 700032" deleted".

Notes

- The meaning of the message number is specified by the machine manufacturer.

- Events with Event ID 16# xx**28** or 16# xx**29** are generated from the basic program.

- The messages stored in the diagnostic buffer can be read out via the operator interface with the associated message texts.

2.6 Interface structure

2.6.1 General

Interface data blocks

The PLC user interfaces on the 840D and 810D are identical except for the data volume. Mapping in interface data blocks is required on account of the large number of signals. These are global data blocks from the viewpoint of the PLC program. During system startup, the basic program creates these data blocks from current NC machine data (no. of channels, axes, etc.). The advantage of this approach is that the minimum amount of PLC RAM required for the current machine configuration is used.

2.6.2 PLC/NCK interface

General

The PLC/NCK interface comprises a data interface on one side and a function interface on the other. The data interface contains status and control signals, auxiliary functions and G functions, while the function interface is used to transfer jobs from the PLC to the NCK.

Data interface

The data interface is subdivided into the following groups:

- NCKspecific signals

- Mode groupspecific signals

- Channelspecific signals

- Axis/spindle/drivespecific signals

Function interface

The function interface is formed by FBs and FCs. The figure below illustrates the general structure of the interface between the PLC and the NCK.
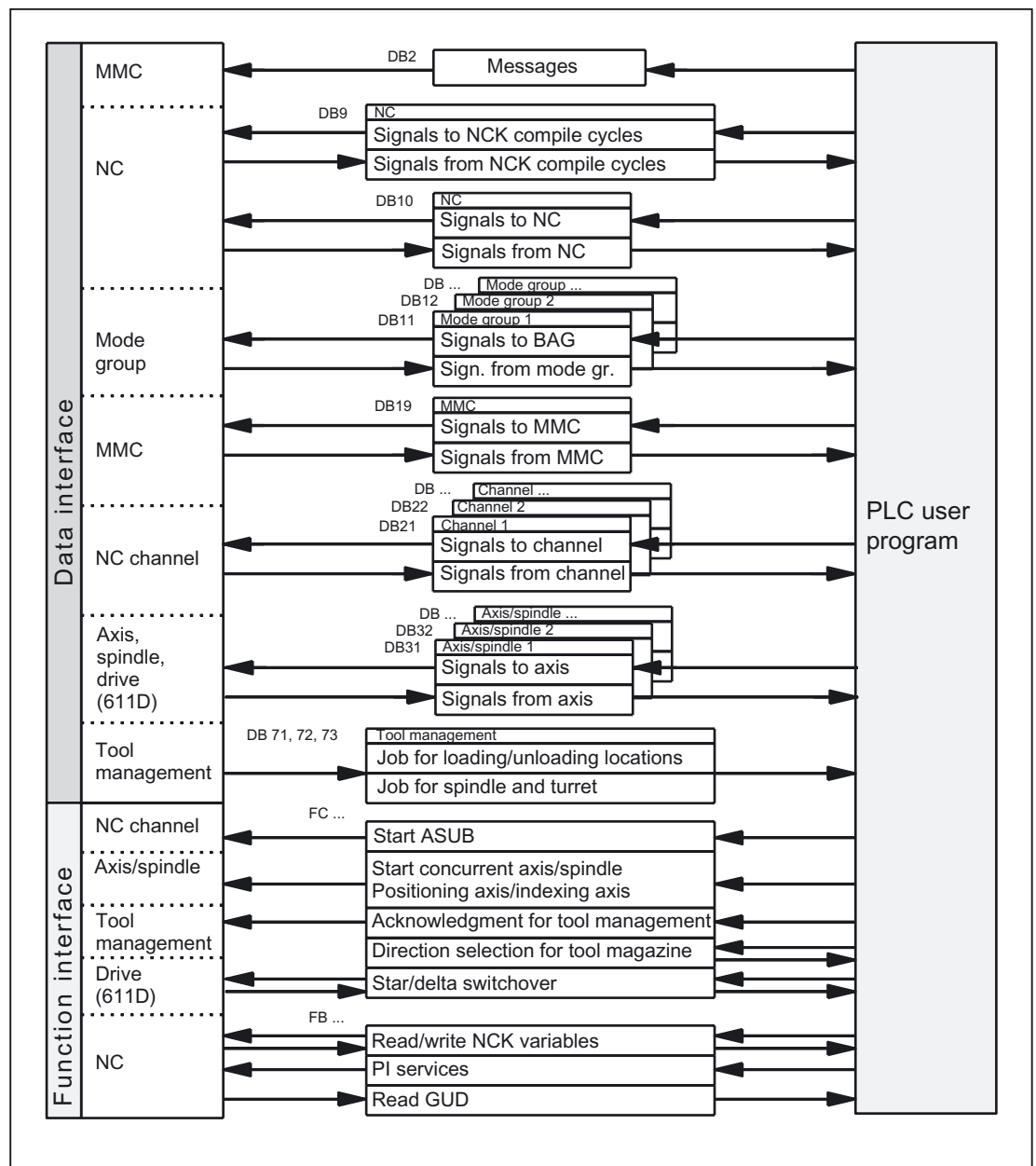
Figure 2-3   PLC/NCK interface

Basics

Compile-cycle signals

In addition to the standard signals exchanged between the PLC and NCK, an interface data block for compile cycles is also generated if required (DB9). The associated signals, which are dependent on the compile cycles, are transmitted cyclically at the start of OB1.

Signals PLC/NC

The group of signals from the PLC to NC includes:

- Signals for modifying the highspeed digital I/O signals of the NC
- Keyswitch and emergency stop signals

Signals NC/PLC

The group of signals from the NC to PLC includes:

- Actual values of the digital and analog I/O signals of the NC
- Ready and status signals of the NC

Also output in this group are the HMI handwheel selection signals and the channel status signals.

The signals for handwheel selection are decoded by the basic program and entered in the machine-/axisspecific interface.
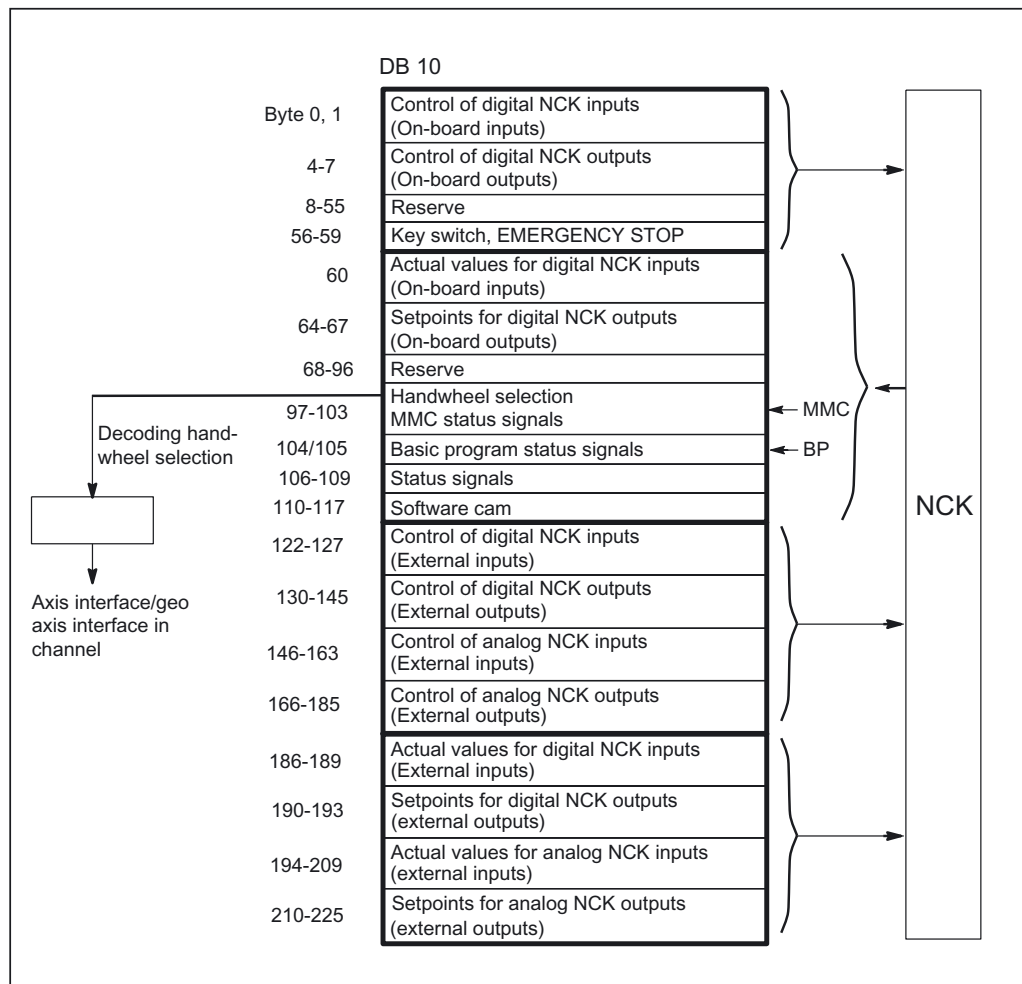
Figure 2-4   PLC/NC interface

Digital/analog inputs/outputs of the NCK

The following must be noted with respect to the digital and analog inputs and outputs of the NCK:

Inputs:

- All input signals or input values of the NCK are also transferred to the PLC.

- The transfer of signals to the NC parts program can be suppressed by the PLC. Instead, a signal or value can be specified by the PLC.

- The PLC can also transfer a signal or value to the NCK even if there is no hardware for this channel on the NCK side.

Outputs:

- All signals or values to be output are also transferred to the PLC.

- The NCK can also transfer signals or values to the PLC even if there is no hardware for this channel on the NCK side.

- The values transferred by the NCK can be overwritten by the PLC.

- Signals and values from the PLC can also be output directly via the NCK I/O devices.

Note

When implementing digital and analog NCK I/Os, you must observe the information in the following references:

References:
/FB2/Function Manual, Extended Functions;
Digital and Analog NCK I/Os (A4)

Signals PLC/Mode group

The operating mode signals set by the machine control panel or the HMI software are transferred to the operating mode group (MG) of the NCK. The mode signals are valid for all NC channels of the mode group on the 810D and 840D. On 840D systems, several mode groups can optionally be defined in the NCK.

The mode group reports its current status to the PLC.
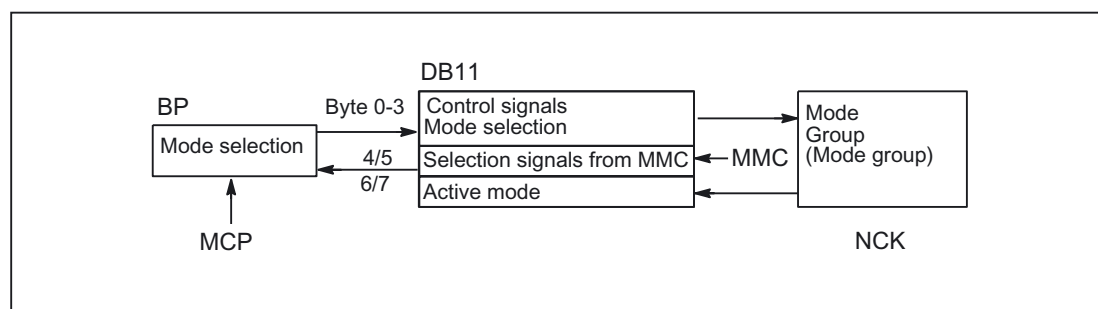
Figure 2-5   PLC/mode group signals (e.g., mode group 1)

Signals PLC/NCK channels

The signal groups below must be considered on the interface:

- Control/status signals

- Auxiliary/G functions

- Tool management signals

- NCK functions

The control/status signals are transmitted cyclically at the start of OB1. The signals entered in the channelspecific interface by the HMI software (HMI signals are entered by the PLC operating system) are also transferred at this time if the signals have been defined on the operator panel front, and not on the MCP.

Auxiliary functions and G functions are entered in the interface data blocks in two ways. First, they are entered with the change signals.

- **M signals** `M00` - `M99` * (they are transferred from the NCK with extended address 0) are also decoded and the associated interface bits set for the duration of one cycle.

  * M signals `M0` - `M99` are transferred from the NCK with extended address 0

- In the case of the **G functions**, the group is additionally decoded and the G functions which are active in the relevant group are entered in the interface data block.

- **S values** are also entered together with the related M signals (`M03`, `M04`, `M05`) in the spindlespecific interface. The axisspecific feedrates are also entered in the appropriate axis-specific interface.

When the **tool management** function is activated in the NCK, the assignment of spindle or revolver and the loading/unloading points are entered (DB71-73) in separate interface DBs.

The triggering and parameter assignment of **NCK functions** is performed by means of PLC function calls.
The following function calls are available, for example:

- Position a linear axis or rotary axis

- Position an indexing axis

- Start a prepared asynchronous subprogram (ASUB)

- Read/write NC variables

- Update magazine assignment

Some of the above functions are described in their own function documentation.
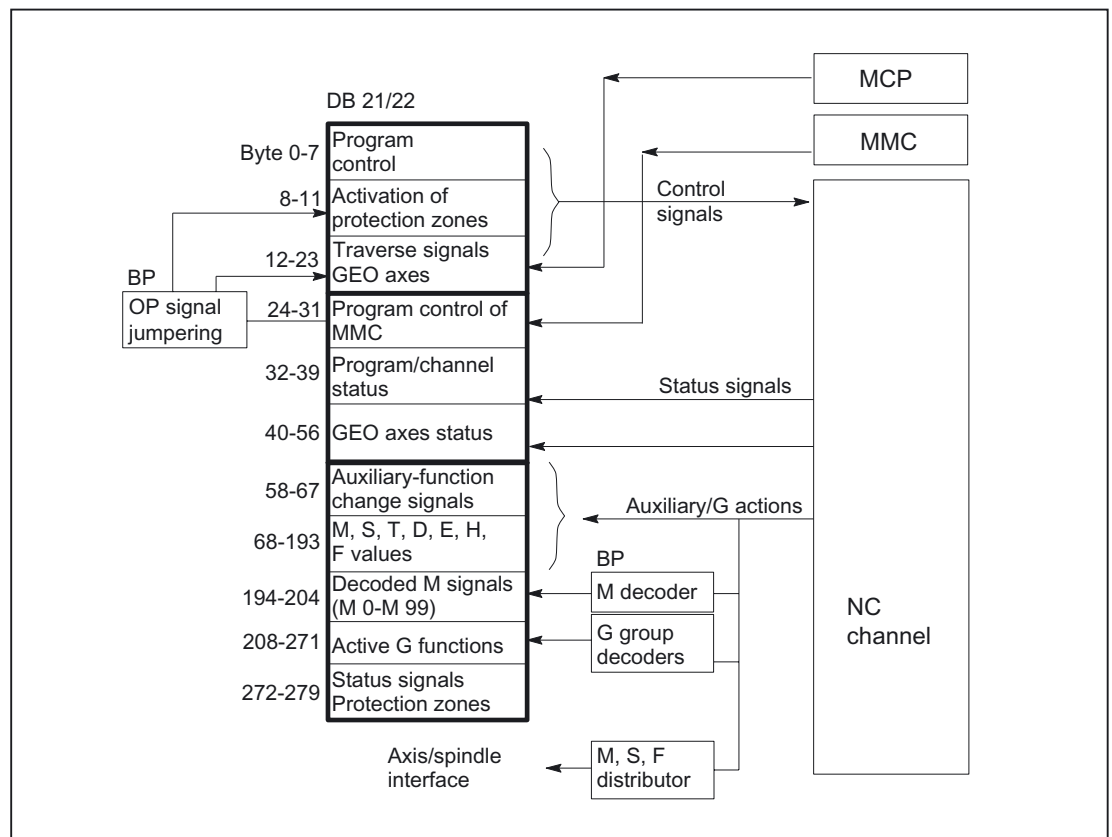
Figure 2-6   PLC/NC channel interface

PLC/axis, spindle, drive signals

The axis-specific and spindle-specific signals are divided into the following groups:

- Shared axis/spindle signals

- Axis signals

- Spindle signals

- Drive signals

The signals are transmitted cyclically at the start of OB1 with the following exceptions:

Exceptions include:

INC-Mode of HMI, axial F-value, M-/S-value.

An axial F value is entered via the M, S, F distributor of the basic program if it is transferred to the PLC during the NC machining process.

The M and S value are also entered via the M, S, F distributor of the basic program if an S value requires processing together with the corresponding M value (`M03`, `M04`, `M05`).
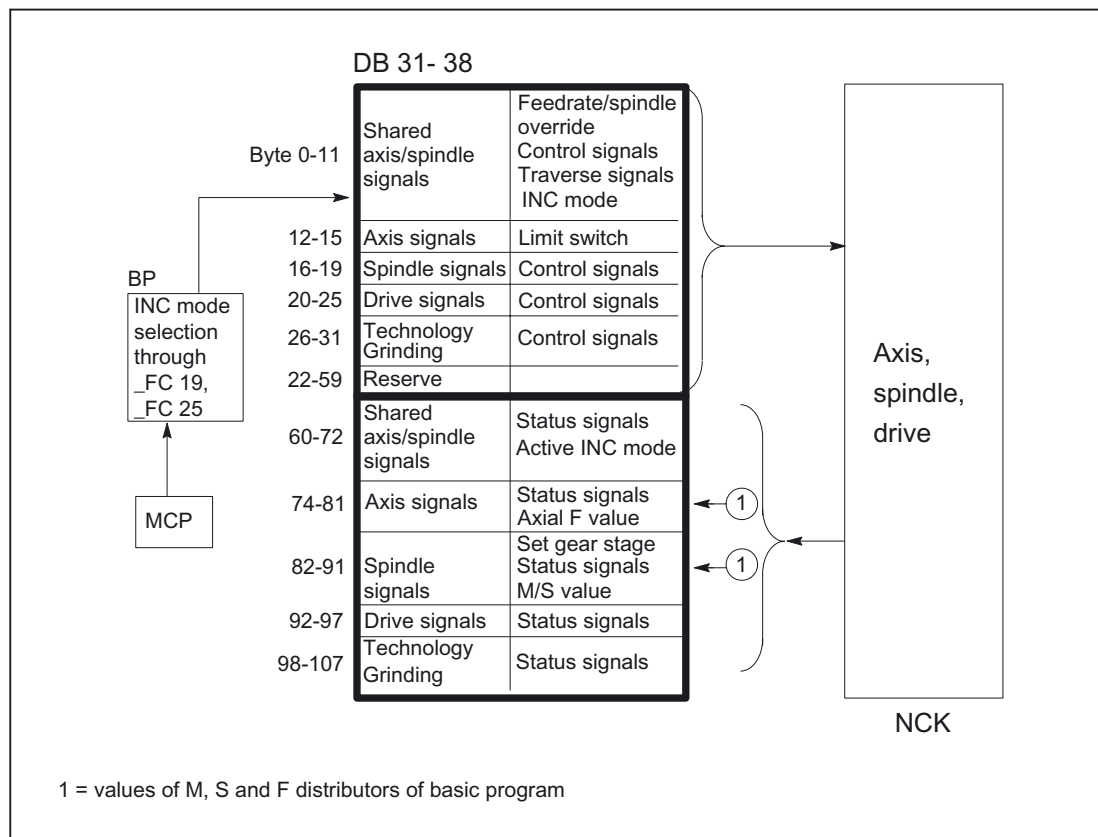
Figure 2-7   Interface between PLC and axes/spindles/drives

2.6.3   Interface PLC/HMI

General

The following groups of functions are required for the PLC/HMI interface:

- Control signals
- Machine operation
- PLC messages
- PLC status display

Control signals

Some control signals are signal inputs from the machine control panel, for example, which have to be taken into account by the HMI software. This group of signals includes, for example, display actual values in MCS or WCS, key disable, etc. These are exchanged via a separate HMI interface data block (DB19).

Machine operation

All operator inputs, which lead to response actions on the machine, are monitored by the PLC. Operator actions are usually performed on the machine control panel. However, it is also possible to perform some operator actions from the operator interface (e.g. mode selection, INC mode selection).

The PLC operating system enters the operating signals sent by the HMI directly into the interface data blocks. In standard cases, the basic program, which decodes the operating signals, allows the operator actions to be performed on the operator interface or on the MCP - given that the capability for this type of operator action is available on the MCP. If required, the user can switch off operation via HMI via an FB1 parameter "MMCToIF".

PLC messages

The signaling functions are based on the system diagnostic functions integrated in the operating system of the AS 300. These have the following characteristics:

- The PLC operating system enters all important system states and state transitions in a **diagnostics status list**. Communication events and I/O module diagnostics data (for modules with diagnostic functions) are also entered.

- Diagnostics events, which lead to a system stop, are also entered with a time stamp in a **diagnostic buffer** (circular buffer) in the chronological order of their occurrence.

- The events entered in the diagnostic buffer are automatically transmitted to human machine interface systems via the MPI or via the OPI through the COM module, once these have issued a ready signal (message service). Transfer to the node ready is a function of the PLC operating system. Receipt and interpretation of the messages are executed by the HMI software.

- An SFC (system function call) can also be used by the user program to enter messages in the diagnostic buffer.

- The events are entered in the diagnostic buffer in coded format.
  The associated message texts must be stored in the operator panel.

An FC for message acquisition (FC10) is prepared in conjunction with the basic program. This FB records events, classifies them into signal groups and reports them to the HMI via the diagnostic buffer.

The message acquisition structure is shown in the figure "Acquisition and signaling of PLC events".
The features include:

- Bit fields for events related to the VDI interface are combined in a single data block (DB2) with bit fields for user messages.

- Bit fields are evaluated at several levels by FC10.

  – **Evaluation 1;** Acquisition of **group signals**

    A group signal is generated for each group of signals when at least one bit signal is set to "1". This signal is generally linked to the disable signal of the VDI interface (on modules with diagnostic functions). The group signals are acquired completely in cycles.

– **Evaluation 2;**Acquisition of

– **Error messages**

A fixed specification exists to define which signals in a group generate an error message when they change from "0" to "1".

– **Evaluation 3;**Acquisition of

– **Operational messages**

A fixed specification exists to define which signals in a group generate an operational message.

- The scope of the user bit fields (user area) is defined as standard as 10 areas with 8 bytes each, but can also be adjusted to suit the requirements of the machine manufacturer via basic program parameters in FB1.

Acknowledgement concept

The following acknowledgment procedures are implemented for error and operational messages:

Operating messages are intended for the display of normal operating states as information for the user. Acknowledgment signals are, therefore, not required for this type of message. An entry is made in the diagnostic buffer for incoming and outgoing messages. The HMI software maintains an uptodate log of existing operating messages using the identifiers "operating message arrived" and "operating message gone".

Error messages are used to display error states on the machine which generally lead to a machine stoppage. Where several errors occur in rapid succession, it is important to be able to distinguish their order of occurrence for troubleshooting purposes. This is indicated, on the one hand, by the order in which they are entered in the diagnostic buffer and on the other, by the time stamp, which is assigned to every entry.

If the cause of the error disappears, the error message is only deleted if the user has acknowledged the message (e.g., by pressing a key on the machine control panel). In response to this signal, the "message acquisition" FC examines which of the reported errors have disappeared and enters these in the diagnostic buffer with the entry "error departed". This enables the HMI software to also maintain an uptodate log of existing error messages. The time of day indicating the time at which the error occurred is maintained for messages, which are still pending (in contrast to a received interrogation).

User Program

The user PLC program merely needs to call the basic program block FC10 with appropriate parameter settings in the cyclic program section and set or reset the bit fields in DB2. All further necessary measures are implemented by the basic program and HMI software.
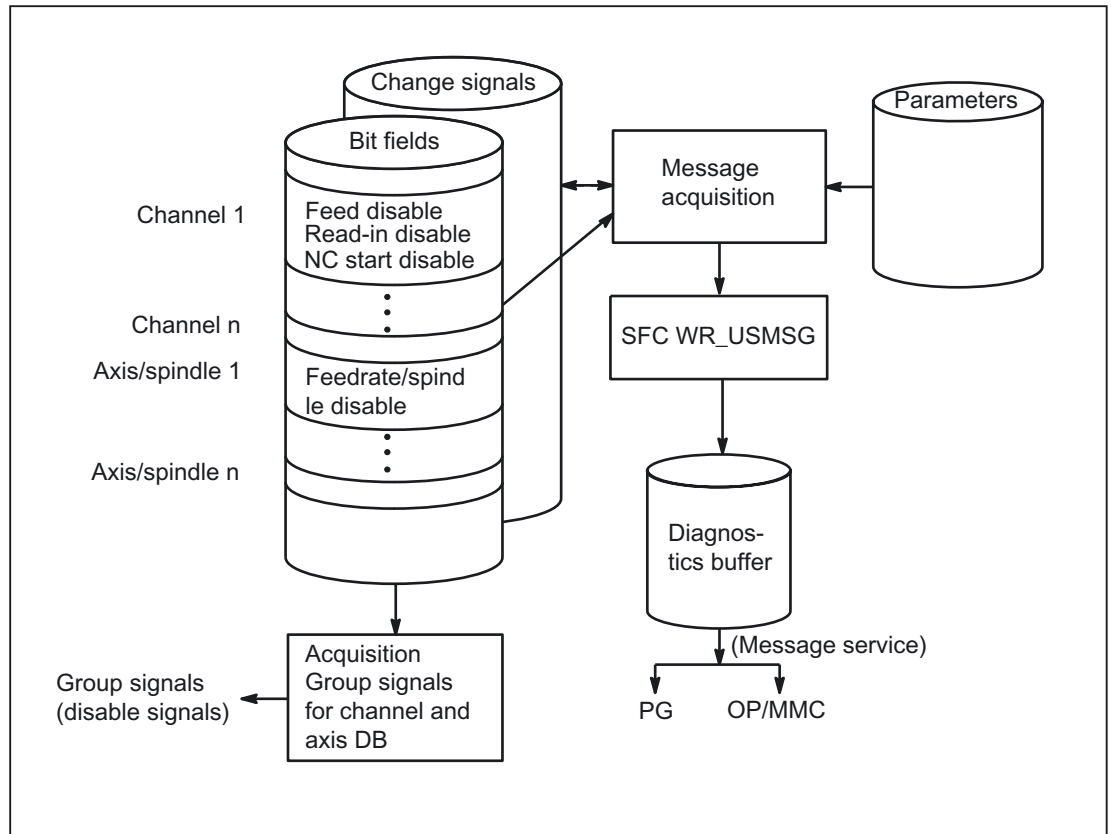
Figure 2-8   Acquisition and signaling of PLC events

2.6.4   PLC/MCP/HHU interface

General

On the SINUMERIK 840D/810D, the machine control panel (MCP) is connected on the same bus that connects the OP with the NC. The advantage of this is that only one bus cable is required to connect the operator unit. The handheld unit (HHU) can be connected to the MPI of the PLC or to the operator panel interface (OPI) (840D only). However, since the OP bus on the 840D supports higher baud rates, two different types of bus topology are provided.

840D topology

On the 840D, the machine control panel is connected to the OPI bus segment (transmission rate 1.5 Mbps) as an active global data node. Where the connection of further keys and displays is required for customized operator panels, an additional keyboard (machine control panel without operating unit) can be used. 64 pushbuttons, switches, etc. and 64 display elements can be connected via ribbon cable.

The signals arriving from the machine control panel are copied by the COM module into the DPR (dualport RAM) for transfer to the NC. The NC in turn transmits them to the PLC (VDI task). The basic program of the PLC enters the incoming signals in the input image. The NC-related signals are generally distributed by the basic program to the VDI interface. This can be modified by the user if required.

The signals (displays) from the PLC to the MCP (displays) are transferred in the opposite direction.

The signals of the handheld unit (HHU) are transferred either via the operator panel interface in the same way as via the machine control panel or by means of the GD service (GD = Global Data) of the MPI interface. The PLC operating system enters the HHU data, for example, in the input image and transfers the display values, e.g., from the output image, back to the HHU. The corresponding parameters are set via system data block SDB210, which is generated with the STEP7 communication configuration tool.
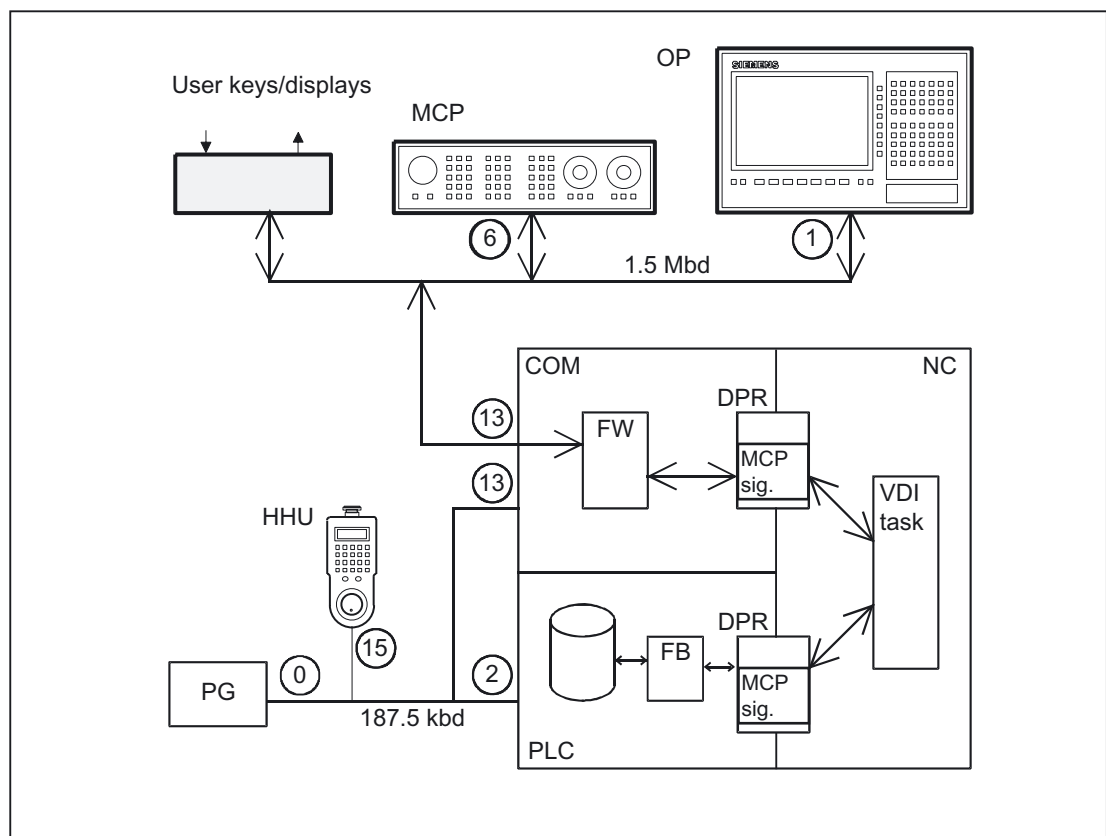
Figure 2-9   Connection of the machine control panel for 810D

810D topology

On the 810D, the machine control panel, handheld unit and OP are connected to the MPI (multipoint Interface) of the AS 300. The transmission rate in this configuration is 187.5 Kbps. The PLC operating system copies the incoming signals straight to the user interface (e.g., input image) at the cycle control point.

Transfer to the VDI interface is performed, as on the 840D, by the user program or by a standard routine of the basic program.
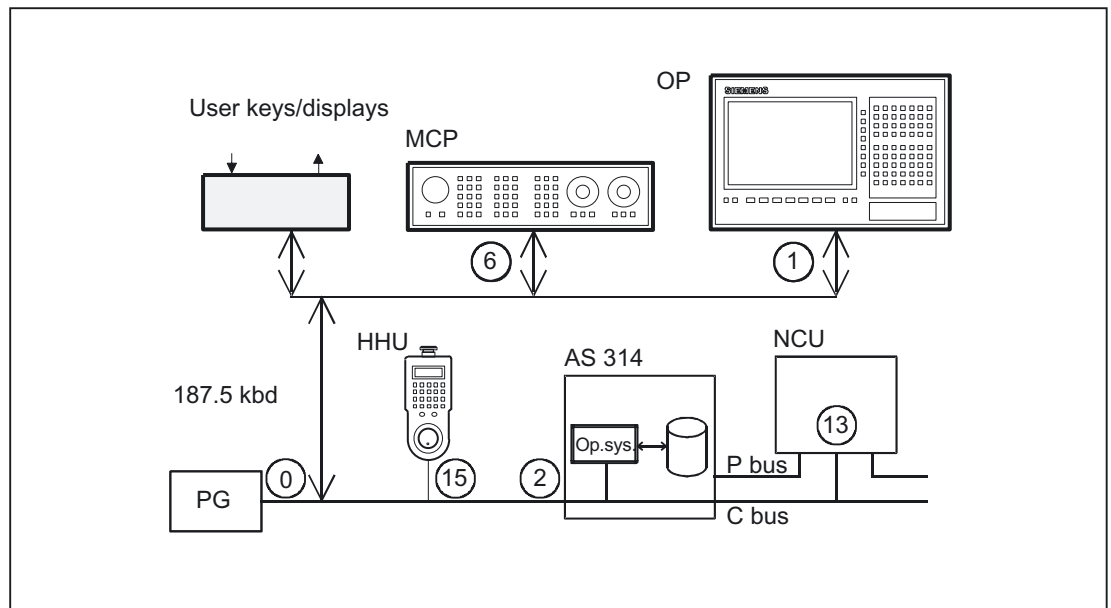
Figure 2-10   Connection of the machine control panel for 810D

Bus addresses

The default bus addresses for the standard configurations are entered in the "Connecting the MCP on the 810D" figures. In addition to the bus addresses, the implicit communication service (global data) also requires the definition of a GD circle number.

The following should be taken into account when allocating bus addresses (node no.):

Bus addresses 840D

The two bus segments on the 840D must be examined separately:

Operator panel bus segment:		
Bus station	Perm. setting range	Standard setting
Operator panel (OP)	1 - 31	1
Machine control panel/keyboard interface	15	(setting via DIP fix)
COM module	31	13
Programming device/PC (e.g., for startup)	fixed	0

PLC bus segment:			
Bus station	Setting range	Default setting	Comment
PLC	31	2	
COM module	Fixed depending on PLC address	3	
Programming device/PC (e.g., for startup)	fixed	0	

MCP interface in the PLC

The signals from the machine control panel are routed via the I/O area by default. A distinction is made between NC and machinespecific signals. NCspecific key signals are normally distributed by FC 19 to the various mode group, NCK, axis and spindlespecific interfaces. The reverse applies to the status signals, which are routed to the machine control panel interface.

Note

FC 19 must be called in the PLC user program.

Customized keys, which can be used to trigger a wide range of machine functions, must be evaluated directly by the user program. The user program also routes the status signals to the output area for the LEDs.
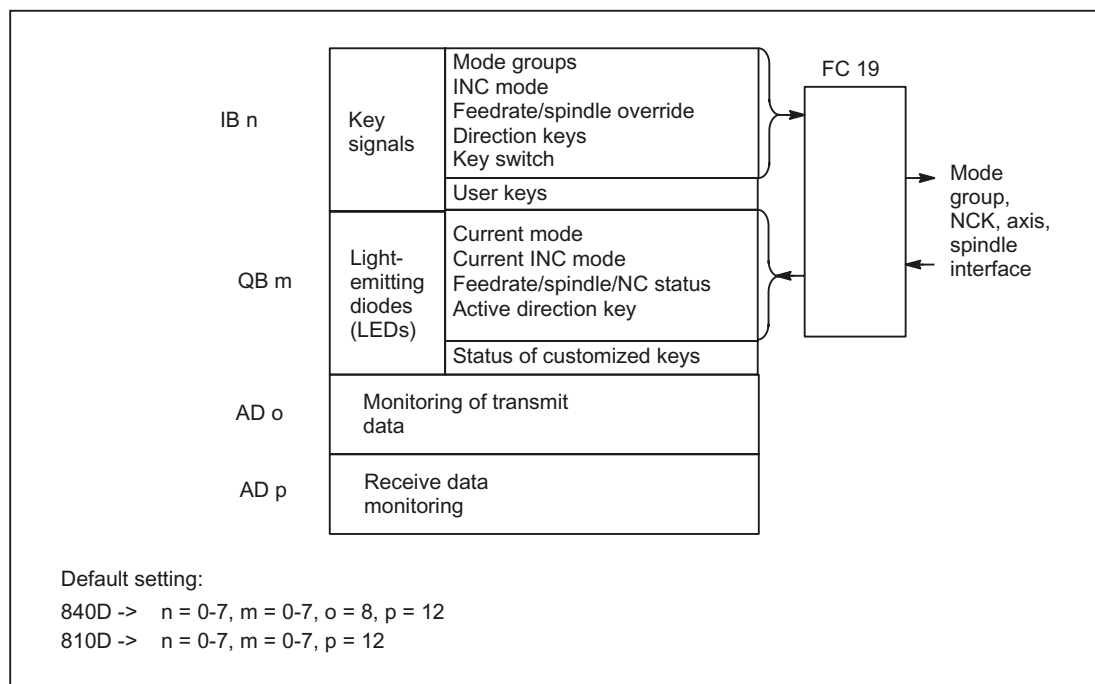
Figure 2-11   Interface to and from machine control panel

2.7   Structure and functions of the basic program

2.7.1   General

General

The program is modular in design, i.e., it is structured according to NCK functions.

In the operating system, a distinction is made between the following levels of execution:

- Startup and synchronization (OB 100)
- Cyclic mode (OB 1)
- Process interrupt handling (OB 40)

Each section of the basic program - as illustrated in the figure below - must be called by the user in OBs 1, 40 and 100.

Figure 2-12   Structure of the PLC program

2.7.2 Startup and synchronization of NCK PLC

Loading the basic program

The basic program must be loaded with the S7 tool when the PLC is in the Stop state. This ensures that all blocks in the basic program will be initiated correctly the next time they are called. An undefined state may otherwise develop on the PLC (e.g., all PLC LEDs flashing).

Startup

The synchronization of NCK and PLC is performed during startup. The system and user data blocks are checked for integrity and the most important basic program parameters are verified for plausibility. In the event of an error, the basic program outputs an error identifier to the diagnostic buffer and switches the PLC to STOP.

A warm restart is not provided, i.e., following system initialization, the operating system runs organization block OB 100 and always commences cyclic execution at the start of OB 1.

Synchronization

The PLC is synchronized with the HMI and NCK during power up.

Sign of life

After a proper initial start and the first complete OB1 cycle (initial setting cycle) the PLC and NCK continuously exchange sign of life signals. If the signoflife signal from the NCK fails to arrive, the PLC/NCK interface is neutralized and the signal "NCK CPU ready" in DB 10 is set to zero.

2.7.3 Cyclic operation (OB1)

General

The NCK PLC interface is processed completely in cyclic mode. From a chronological viewpoint, the basic program runs ahead of the user program. In order to minimize the execution time of the basic program, only the control/status signals are transmitted cyclically; transfer of the auxiliary functions and G functions only takes place on request.

The following functions are performed in the cyclic part of the basic program:

- Transmission of the control/status signals
- Distribution of the auxiliary and G functions
- M decoding (`M00 - M99`),
- M, S, F distribution
- Transmission of the machine control panel signals via the NCK (on the 840D only)
- Acquisition and conditioning of the user errors and operating messages.

Control/Status signals

A shared feature of the control and status signals is that they are bit fields. The basic program updates them at the start of OB1.

The signals can be subdivided into the following groups:

- General signals

- Mode-groupspecific signals such as operating modes

- Channelspecific signals such as program and feed modifications

Axis- and spindlespecific signals such as feed disable

Auxiliary and G functions

The auxiliary and G functions have the following characteristics:

- Transfer to the PLC is blocksynchronous (referred to a parts program block)

- Transfer is acknowledgecontrolled.

- The acknowledgment times have an immediate effect on the execution time of NC blocks containing auxiliary functions requiring acknowledgment.

The value range is presented in the table below:

Function	Structure		Value range		Data type	
	1. Value	2. Value	1. Value	2. Value	1. Value	2. Value
G function		G function		255[1]		Byte
M word	M group	M word	99	99.999.999	Word	DWord
S word	Spindle no.	S word	6	Floating point[2]	Word	DWord
T word	Magazine no.	T word	99	65535	Word	Word
D word	-	D word	99	255	Byte	Byte
H word	H group	H word	99	Floating point	Word	DWord
F word	Axis No.	F word	18	Floating point	Word	DWord

[1] relative number, transferred for each G group

[2] corresponding STEP 7 format (24-bit mantissa, 8-bit exponent)

The M, S, T, H, D and F values sent by the NCK are output together with the accompanying change signals to the **CHANNEL DB** interface via the auxiliary/G functions (see documentation "Lists of SINUMERIK 840D, 810D"). The function value and the extended address are transferred to the appropriate data word. The accompanying modification signal is activated to 1 for one PLC cycle. When the modification signal is reset, the acknowledgment is passed to the NCK. The acknowledgment of highspeed auxiliary functions is given by the basic program immediately the basic program detects the auxiliary function.

In addition to distribution of the auxiliary and G functions, selected signals are processed as described below.

M decoder

M functions can be used to transfer both switching commands and fixed point values. Decoded dynamic signals are output to the **CHANNEL DB** interface for standard M functions (range `M00 - M99`); signal length = 1 cycle time.

G group decoders

In the case of G functions sent by the NCK, the related groups are decoded and the current G number is entered in the corresponding interface byte of the CHANNEL DB, i.e., all active G functions are entered in the channel DBs. The entered G functions are retained even after the NC program has terminated or aborted.

Note

During system startup, all G group bytes are initialized with the value "0".

M, S, F distributor

The M, S, F, distributor is used to enter spindlespecific M words M(1...6)=[3,4,5], S words and F words for axial feeds in the appropriate **spindle and axis data blocks**. The criterion for distribution is the extended address, which is passed to the PLC for M words, S words and axial F words.

MCP signal transmission

On the 840D, the MCP signals are transferred to the NCK via the serial bus (MPI) and from there to the PLC. A function call from the basic program transfers the signals to the interface of inputs and outputs specified by basic program parameters. The status signals for controlling the LEDs on the machine control panel are passed in the opposite direction.

User messages

The acquisition and processing of the user error and operational messages is performed by an FC in the basic program.

2.7.4 Time-alarm processing (OB 35)

General

The user must program **OB 35** for time-alarm processing. The default time base setting of **OB 35** is 100 ms. Another time base can be selected using the STEP7 application "S7 Configuration". However, the OB 35 with a time base setting not less than approx. 15 ms must not be used without additional measures, otherwise this would cause the PLC CPU to stop. The stop is caused by reading of the system state list during power up of the HMI. This reading process blocks priority class control for approx. 8 to 12 ms. The OB 35 with a time base set to a rather lower value is then no longer processed correctly. If, however, small time base settings are required for OB 35, the stop can be prevented by programming OB 80 with at least the program command "BE".

2.7.5 Process interrupt processing (OB 40)

General

A process interrupt **OB 40** (interrupt) can, for example, be triggered by appropriately configured I/Os or by certain NC functions. Due to the different origin of the interrupt, the PLC user program must first interpret the cause of the interrupt in OB 40. The cause of the interrupt is contained in the local data of OB 40.

(For more information please also refer to the SIMATIC STEP7 description or the STEP7 online help).

2.7.6 Response to NC failure

General

During cyclic operation, the PLC continuously monitors NC availability by querying the sign-oflife character. If the NCK is no longer reacting, then the NCK PLC interface is neutralized and IS **NCK CPU ready** in **signals from NC group (DB 10.DBX 104.7)** is reset. The signals sent by the NCK to the PLC are also set to an initial state.

The PLC itself remains active so that it can continue to control machine functions.

Signals NCK to PLC

The signals sent by the NCK to the PLC are divided into the following groups:

- Status signals from the NCK, channels, axes and spindles

- Modification signals of the auxiliary functions

- Values of the auxiliary functions

- Values of the G functions

Status signals:

The status signals from the NCK, channels, axes, and spindles are reset.

Auxiliary-function modification signals:

Auxiliary-function modification signals are also reset.

Auxiliary-function values:

Auxiliary-function values are retained so that it is possible to trace the last functions triggered by the NCK.

G-function values:

G-function values are reset (i.e., initialized with the value 0).

PLC → NCK signals

The signals sent by the PLC to the NCK are divided into control signals and tasks that are transferred by FCs to the NCK.

Control signals:

The control signals from the PLC to the NCK are frozen; cyclic updating by the basic program is suspended.

Jobs from PLC to NCK:

The FCs and FBs, which are used to pass jobs to the NCK, must no longer be processed by the PLC user program, as this could lead to incorrect checkback signals. During runup of the control, a job
(e.g., read NCK data) must not be activated in the user program until the **NCK-CPU ready** signal is set.

2.7.7 Functions of the basic program called from the user program

General

In addition to the modules of the basic program, which are called at the start of OBs 1, 40 and 100, functions are also provided, which can be called at a suitable point in the user program and supplied with parameters.

These functions can be used, for example, to pass the following jobs from the PLC to the NCK:

- Traverse concurrent axes (FC 15, FC 16),
- Start asynchronous subprograms (ASUBs) (FC 9),
- Select NC programs and NC blocks (FB 4),
- Control of spindle (FC 18),
- Read/write variables (FB 2, FB 3).

Note

The following note will later help you to check and diagnose a function call (FCs, FBs of basic program). These are FCs and FBs, which are controlled by a trigger signal (e.g., via parameter Req, Start, etc.), and which supply an execution acknowledgment as an output parameter (e.g., via parameter Done, NDR, Error, etc.). A variable compiled of other signals, which produce the trigger for the function call should be set. Start conditions may be reset only as a function of the states of parameters Done, NDR and Error.
This control mechanism may be positioned in front of or behind the function call. If the mechanism is placed after the call, the output variables can be defined as local variables (advantage: Reduction of global variables, flags, data variables and timerelated advantages over data variables).
The trigger parameter must be a global variable (e.g., flag, data variable).
In OB 100, jobs still activated by the user program (Parameter Req,
Start, etc.:= TRUE) must be set to zero at the named parameters. A POWER OFF/ON could result in a state in which jobs are still active.

Concurrent axes

The distinguishing features of concurrent axes are as follows:

- They can be traversed either from the PLC or from the NC.

- They can be started from a function call on the PLC in all operating modes.

- The start is independent of NC block boundaries.

Function calls are available for positioning (FC 15) and indexing axes (FC 16).

```
FC 15                                      FC 16
         PosAx                                       PartAx
── Start           InPos ──           ── Start             InPos ──
── AxisNo          Active ──          ── AxisNo            Active ──
── IC             StartErr ──         ── IC              StartErr ──
── Inch             Error ──          ── DC                Error ──
── HWheelOv                           ── Plus
── Pos                                ── Minus
── FRate                              ── Pos
                                      ── FRate

  Input              Output
  parameters         parameters
```
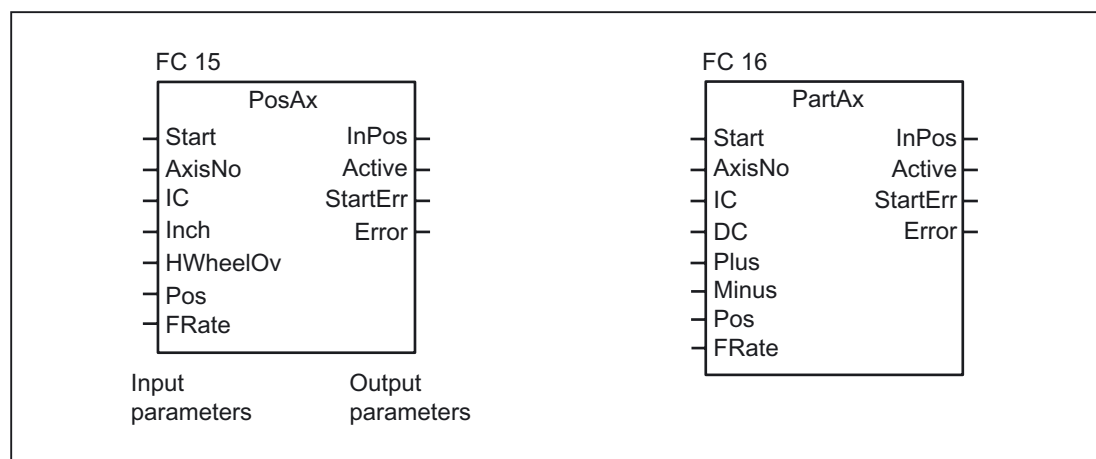
ASUBs

Asynchronous subprograms (ASUBs) can be used to activate any selected function in the NC. Before an asynchronous subprogram can be started from the PLC, it must be ensured that it is available and prepared by the NC program or by FB 4 PI services (ASUB). ASUBs can only be started in MDA or Automatic mode with **running** parts program.

Once prepared in this way, it can be started at any time from the PLC. The NC program running on the channel in question is interrupted by the asynchronous subprogram. The asynchronous subprogram is started by FC 9.

```
FC 9
         ASUP
── Start            Activ ──
── ChanNo           Done ──
── IntNo            Error ──
                  StartErr ──

        Ref
```
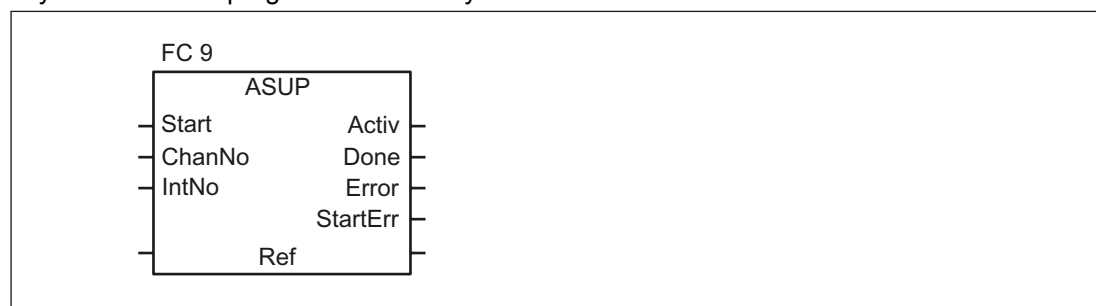
Note

If an asynchronous subprogram has not been prepared by an NC program or by FB 4 (ASUB) (e.g., if no interrupt no. has been assigned), a start error is output.

Read/Write NC variables

NCK variables can be read with FB GET while values can be entered in NCK variables with FB PUT. The NCK variables are addressed via identifiers at inputs Addr1 to Addr8. The identifiers (symbols) point to address data, which must be stored in a global DB. To allow generation of this DB, PC software is supplied with the basic program with which the required variables can be selected from a table, which is also supplied. The selected variables are first collected in a second, projectrelated list. Command **Generate DB** creates a *.AWL file, which must be linked to the program file for the machine concerned and compiled together with the machine program.

1 to 8 values can be read or written with a read or write job. If necessary, the values are converted (e.g., NCK floating point values (64 bits) are converted to PLC format (32 bits with 24-bit mantissa and 8bit exponent) and vice versa). A loss of accuracy results from the conversion from 64 bits to 32bit REAL. The maximum precision of 32bit REAL numbers is approximately 10 to the power of 7.

```
        FB 2                                    FB 3
             GET                                     PUT
    ─ Req              Error ─             ─ Req              Error ─
    ─ NumVar           NDR ─               ─ NumVar           Done ─
    ─ Addr1-8          State ─             ─ Addr1-8          State ─
    ─ Unit1-8                              ─ Unit1-8
    ─ Column1-8                            ─ Column1-8
    ─ Line1-8                              ─ Line1-8

                  RD8                                    SD8
```
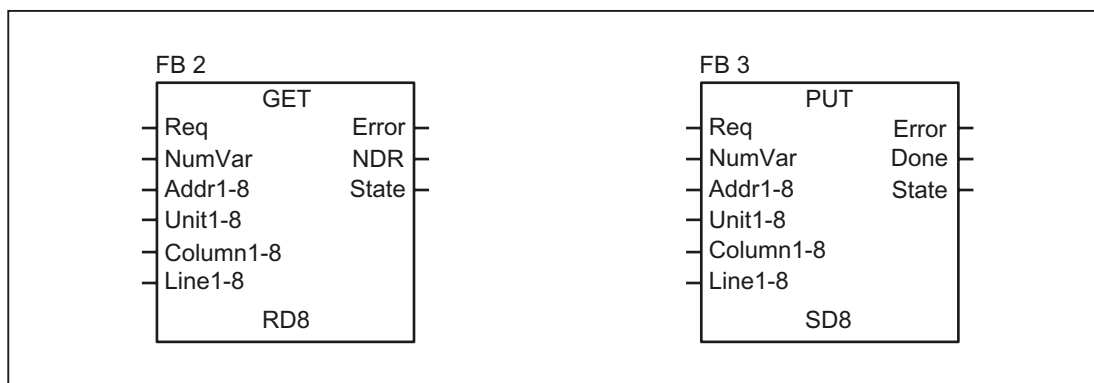
2.7.8 Symbolic programming of user program with interface DB

General

> **Note**
>
> The files NST_UDT.AWL and TM_UDT.AWL are supplied with the PLC basic program.

The compiled UDT blocks from these two files are stored in the CPU program of the basic program.

A UDT is a data type defined by the user that can, for example, be assigned to a data block generated in the CPU.

Symbolic names of virtually all the interface signals are defined in these UDT blocks.

The UDT numbers 2, 10, 11, 19, 21, 31, 71, 72, 73 are used.

The assignments have been made as follows:

UDT assignments		
UDT number	Assignment to interface DB	Description
UDT2	DB 2	Alarms/messages
UDT10	DB10	NCK signals
UDT11	DB11	Mode group signals
UDT19	DB19	HMI signals
UDT21	DB21 to DB30	Channel signal
UDT31	DB31 to DB61	Axis/spindle signals
UDT71	DB71	Tool management: Load/unload locations
UDT72	DB72	Tool management: Change in spindle
UDT3	DB73	Tool management: Change in revolver

To symbolically program the interface signals, the interface data blocks must first be symbolically assigned using the symbol editor.

For example, symbol "AxisX" is assigned to operand DB31 with data type UDT31 in the symbol file.

After this input, the STEP 7 program can be programmed in symbols for this interface.

> **Note**
>
> Programs generated with an earlier software version that utilize the interface DBs described above can also be converted into symbol programs. To do so, however, a fully qualified instruction is needed for data access in the earlier program (e.g., "U DB31.DBX 60.0" - this command is converted to "AxisX.E_SpKA" when the symbols function is activated in the editor).

Description

Abbreviated symbolic names of the interface signals are defined in the two STL files NST_UDT.AWL and TM_UDT.AWL.

In order to create the reference to the names of the interface signals, the name is included in the comment after each signal.

The symbolic names, commands and absolute addresses can be viewed by means of a STEP 7 editor command when the UDT block is opened.

Note

Unused bits and bytes are listed, for example, with the designation "f56_3".

"56": Byte address of the relevant data block

"3": Bit number in this byte

2.7.9   M decoding acc. to list

Description of functions

When the **M decoding according to list** function is activated via the GP parameter of FB1 "ListMDecGrp" (number of M groups for decoding), up to 256 M functions with extended address can be decoded by the basic program.

The assignment of the M function with extended address and the bit to be set in the signal list is defined in the decoding list. The signals are grouped for this purpose.

The signal list contains 16 groups with 16 bits each as decoded signals.

There is only one decoding list and one signal list i.e. this is a cross-channel function.

The M functions are decoded. Once they are entered in the decoding list, then the associated bit in the signal list is set.

When the bit is set in the signal list, the readin disable in the associated NCK channel is set simultaneously by the basic program.

The readin disable in the channel is reset once the user has reset all the bits output by this channel and thus acknowledged them.

The output of an M function decoded in the list as a highspeed auxiliary function does not result in a readin disable.

The figure below shows the structure of the **M decoding according to list**:
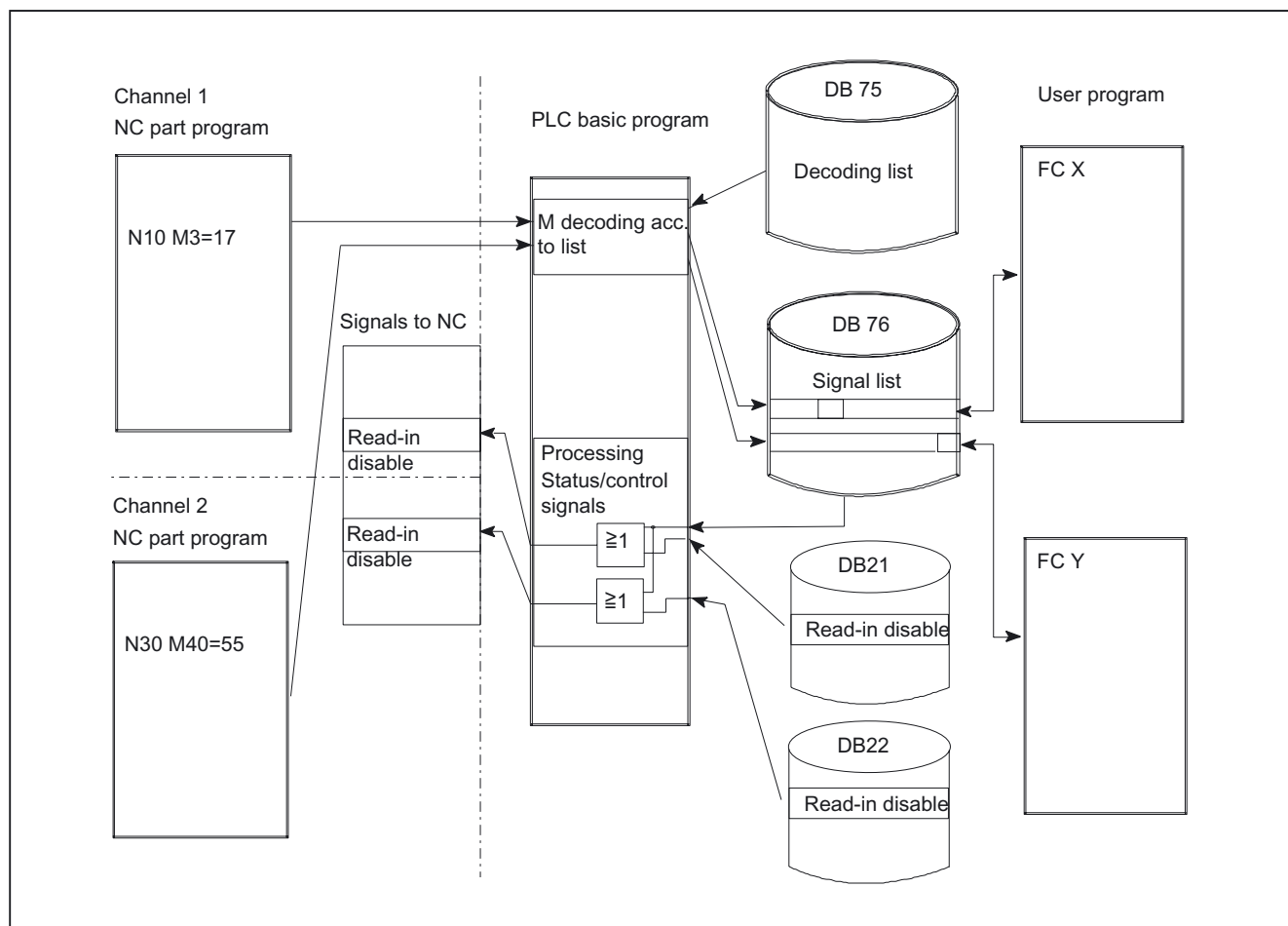
Figure 2-13   M decoding acc. to list

Activation of the function

The number of groups to be evaluated/decoded is indicated in the basic program parameter "ListMDecGrp" when FB 1 is called in OB 100 (see also FB 1 description). M decoding is activated if this value is between 1 and 16. Before the function is activated, the decoding list DB75 must be transferred to the PLC followed by a restart.

Structure of decoding list

The source file for the decoding list (MDECLIST.AWL) is supplied with the basic program. DB 75 is created when the STL source is compiled.

There must be an entry in decoding list DB 75 for every group of M functions to be decoded.

A maximum of 16 groups can be created.

16 bits are available in each group in the list of decoded signals.

The assignment between the M function with extended address and the bit to be set in the signal list is specified via the first and last M functions in the decoding list.

The bit address is generated correspondingly from the first M function ("MFirstAdr") to the last M function ("MLastAdr") from bit 0 up to maximum bit 15 for each group.

Each entry in the decoding lists consists of 3 parameters, each of which is assigned to a group.

Assignment of groups			
Group	Extended M address	First M address in group	Last M address in group
1	MSigGrp[1].MExtAdr	MSigGrp[1].MFirstAdr	MSigGrp[1].MLastAdr
2	MSigGrp[2].MExtAdr	MSigGrp[2].MFirstAdr	MSigGrp[2].MLastAdr
...	...	...	...
16	MSigGrp[16].MExtAdr	MSigGrp[16].MFirstAdr	MSigGrp[16].MLastAdr

Type and value range for signals			
Signal	Type	Value range	Remark
MExtAdr	Int	0 ... 99	Extended M address
MFirstAdr	DInt	0 to 99.999.999	First M address in group
MLastAdr	Dint	0 to 99.999.999	Last M address in group

Signal list

Data block DB 76 is set up when the function is activated.

A bit is set in the appropriate group in DB 76 for an M signal decoded in the list.

At the same time, a readin disable is set in the channel in which the M function has been output.

Example

Three groups of M commands are to be decoded in the following example:

- · M2 = 1 to M2 = 5
- · M3 = 12 to M3 = 23
- · M40 = 55

Structure of the decoding list in DB 75:

Example parameters				
Group	Decoding list (DB 75)			Signal list
	Extended M address	First M address in group	Last M address in group	DB 76
1	2	1	5	DBX0.0 to DBX0.4
2	3	12	23	DBX2.0 to DBX3.3
3	40	55	55	DBX4.0

```
DATA_BLOCK DB 75
TITLE =
 VERSION : 0.0
     STRUCT
         MSigGrp : ARRAY [1 .. 16 ] OF STRUCT
             MExtAdr :    INT ;
             MFirstAdr :  DINT;
             MLastAdr :   DINT;
         END_STRUCT;
     END_STRUCT;
BEGIN
     MSigGrp[1].MExtAdr :=   2;
     MSigGrp[1].MFirstAdr    L#1;
     :=
     MSigGrp[1].MLastAdr     L#5;
     :=
     MSigGrp[2].MExtAdr :=   3;
     MSigGrp[2].MFirstAdr    L#12;
     :=
     MSigGrp[2].MLastAdr     L#23;
     :=
     MSigGrp[3].MExtAdr :=   40;
     MSigGrp[3].MFirstAdr    L#55;
     :=
     MSigGrp[3].MLastAdr     L#55;
     :=
END_DATA_BLOCK
```

Structure of FB 1 in the OB100

(enter the number of M groups to be decoded in order to activate the function):

```
Call FB 1, DB 7(
...
ListMDecGrp := 3,// M decoding of three groups
....
);
```

The appending of the entry in OB 100 and transfer of DB 75 (decoding list) to the AG must be followed by a restart. During the restart, the basic program sets up DB76 (signal list).

If the NC program is started at this point and the expanded M function (e.g., M3=17) is processed by the NCK, this M function will be decoded and bit 2.5 set in DB 76 (see decoding list DB 75). At the same time, the basic program sets the read-in disable and the processing of the NC program is halted (in the corresponding NC-channel DB the entry "expanded address M function" and "M function no." is made).

The readin disable in the channel is reset once the user has reset and, therefore, acknowledged, all the bits output by this channel in the signal list (DB 76).

2.7.10 PLC machine data

General

The user has the option of storing PLCspecific machine data in the NCK. The user can then process these machine data after the power-up of the PLC (OB 100). This enables, for example, user options, machine expansion levels, machine configurations, etc., to be implemented.

The interface for reading these data lies in the DB 20. However, DB20 is set up by the basic program during powerup only when user machine data are used i.e. sum of GP parameters "UDInt", "UDHex" and "UDReal" is greater than zero.

The size of the individual areas and thus also the total length of DB20
is set by PLC machine data:
MD14504 MAXNUM_USER_DATA_INT
MD14506 MAXNUM_USER_DATA_HEX
MD14508 MAXNUM_USER_DATA_FLOAT
is set and specified to the user in the GP parameters:
"UDInt", "UDHex" and "UDReal"
.
The data is stored in the DB 20 by the BP in the sequence:
Integer-MD, Hexa-Fields-MD, Real-MD.

The integer and real values are stored in DB 20 in S7 format.
Hexadecimal data are stored in DB20 in the order in which they are input (use as bit fields).
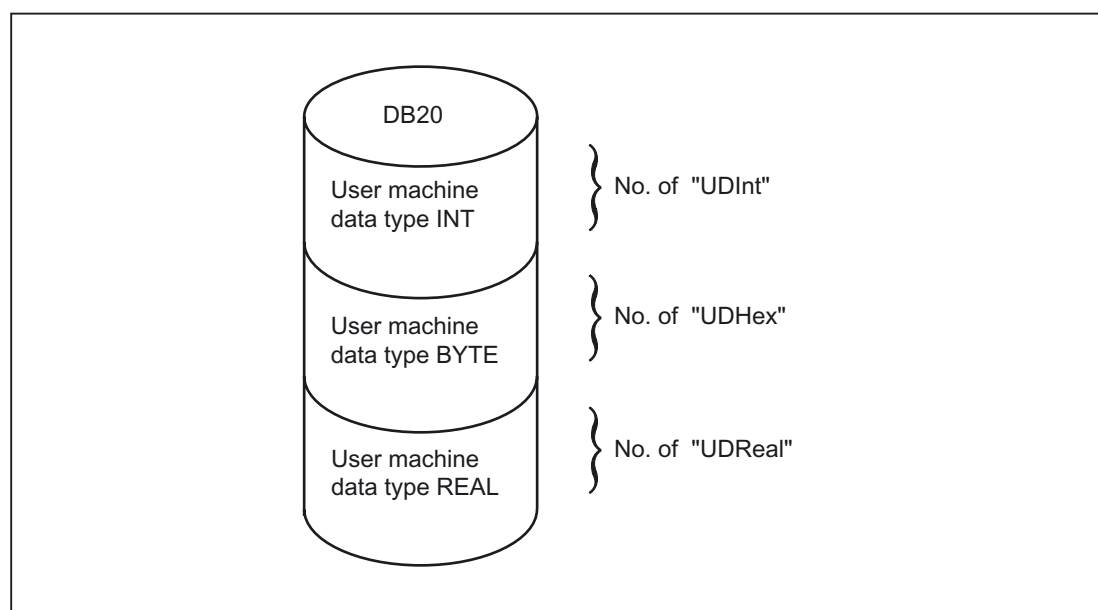
Figure 2-14   DB 20

Note

If the number of PLC machine data used is increased later, then DB20 must be deleted beforehand. To prevent such extensions in use having any effect on the existing user program, the data in DB20 should be accessed in symbolic form wherever possible, e.g., by means of a structure definition in the UDT.

Interrupts	
400120	**Delete DB 20 in PLC and restart**
Explanation	DB length is not the same as the required DB length
Response	Interrupt display and PLC Stop
To correct or avoid errors	Delete DB 20 followed by RESET
Continuation	After cold restart

Example

The project in the example requires 4 integer values, 2 hexadecimal fields with bit information and 1 real value.

Machine data:

MD14510 USER_DATA_INT[0]	123	
MD14510 USER_DATA_INT[1]	456	
MD14510 USER_DATA_INT[2]	789	
MD14510 USER_DATA_INT[3]	1011	

...

MD14512 USER_DATA_HEX[0]	12
MD14512 USER_DATA_HEX[1]	AC

...

MD14514 USER_DATA_FLOAT[0]   123.456

GP Parameter (OB 100):

```
CALL FB 1, DB 7 (
        MCPNum :=        1,
        MCP1In :=        P#E0.0,
        MCP1Out :=       P#A0.0,
        MCP1StatSend :=  P#A8.0,
        MCP1StatRec :=   P#A12.0,
        MCP1BusAdr :=    6,
        MCP1Timeout :=   S5T#700MS,
        MCP1Cycl :=      S5T#200MS,
        NCCyclTimeout := S5T#200MS,
```

```
        NCRunupTimeout     S5T#50S;
        :=
```

BP parameters (to scan runtime):

```
        l gp_par.UDInt;    //=4,
        l gp_par.UDHex;    //=2,
        l gp_par.UDReal;   //=1 )
```

During PLC power-up, DB20 was generated with a length of 28 bytes:

DB 20	
address	data
0.0	123
2.0	456
4.0	789
6.0	1011
8.0	b#16#12
9.0	b#16#AC
10.0	1.234560e+02

The structure of the machine data used is specified in a UDT:

```
TYPE UDT 20
   STRUCT
      UDInt :  ARRAY [0 .. 3 ] OF INT ;
      UDHex0 : ARRAY [0 .. 15 ] OF BOOL ;
      UDReal : ARRAY [0 .. 0 ] OF REAL  //Description as field, for
             ;                          // later expansions

   END_STRUCT;
END_TYPE
```

Note

ARRAY OF BOOL are always sent to even-numbered addresses. For this reason, an array range of 0 to 15 must generally be selected in the UDT definition or all Boolean variables specified individually.

Although only a REAL value is used initially in the example, a field (with one element) has been created for the variable. This ensures that extensions can be made easily in the future without the symbolic address being modified.

Symbolic accesses

An entry is made in the symbol table to allow data access in symbolic form:

Symbol	Operand	Data type
UData	DB 20	UDT 20

Access operations in user program (list includes only symbolic read access):

```
...
        L       "UData".UDInt[0];
        L       "UData".UDInt[1];
        L       "UData".UDInt[2];
        L       "UData".UDInt[3];

        U       "UData".UDHex0[0];
        U       "UData".UDHex0[1];
        U       "UData".UDHex0[2];
        U       "UData".UDHex0[3];
        U       "UData".UDHex0[4];
        U       "UData".UDHex0[5];
        U       "UData".UDHex0[6];
        U       "UData".UDHex0[7];
        ...     ...
        U       "UData".UDHex0[15];

        L       "UData".UDReal[0];
...
```

2.7.11   Configuration of machine control panel, handheld unit

General

The communications system integrated in the NC permits a maximum of 2 machine control panels and one handheld unit to exchange data with the 810D and the 840D. An SDB210 is not required to transfer the signals of these components. The information given below is based on the assumption that no SDB210 has been installed for the components concerned.

Parameterization of components is always performed by means of a call of basic program block FB1 in OB100. FB1 stores its parameters in the associated instance data block (DB7, symbolic "gp_par"). Separate parameter sets are provided for each machine control panel and the handheld unit. The input/output addresses of the user must be defined in these parameter sets. These input and output addresses are also used in FC19, FC24, FC25, FC26 and FC13. Addresses for status information, MPI or OPI (a GD parameter set must be set for the handheld unit rather than an MPI address) must also be defined. The default time settings for timeout and cyclic forced retriggering do not have to be changed.

Activation

Each component is activated either via the number of machine control panels (parameter MCPNum) or, in the case of the handheld unit, parameter BHG := 2 (BHG := 1 corresponds to a link via the MPI interface in conjunction with an SDB210). Whether a component is to be linked to the OPI or the MPI is determined by parameters MCPMPI and BHGMPI.

Handheld unit

The handheld unit addresses the MPI or OPI by means of a GD parameter set. These parameter values must be assigned according to the handheld unit settings. However, the parameter names on the handheld unit are the reverse of the parameter names in the basic program. All Send type parameters on the handheld unit must be defined as Rec type (and Rec in Send type) in the basic program.

Control signals

Parameters MCP1Stop, MCP2Stop and BHGStop can be used to stop communication with individual components (parameter setting = 1). This stop or activation of communication can be applied in the current cycle. However, the change in value must be implemented through the symbolic notation of the parameters and not by means of another FB 1 call.

Example of stopping transfer from the first machine control panel:

```
SET;
S              gp_par.MCP1Stop;
```

Setting parameters MCP1Stop, MCP2Stop, BHGStop also results in a suppression of alarms 40260 to 40262.

MPI switchover, OPI address

An existing connection with an MCP (Machine Control Panel) or HHU (HandHeld Unit) can be aborted. Another MCP or HHU component already connected to the bus (with another MPI or OPI address) can then be activated. Proceed as follows to switch addresses:

1. Stop communication with component to be decoupled via parameter MCP1Stop or MCP2Stop or BHGStop = 1.

2. After checkback in DB10 byte 104 (relevant bits 0, 1, 2 set to 0). Change in the bus address or GD parameter set of this unit to that of the new component.

3. In this PLC cycle, communication with the new component can now be activated again by means of parameter MCP1Stop or MCP2Stop or BHGStop = 0.

4. Communication with the new component is taking place when the checkback in DB10 byte 104 (relevant bits 0, 1, 2 is set to 1).

As described in Section Control signals, all parameters must be programmed according to data type.

Switching off flashing MCP:

With MCP firmware V5.01.02 and higher, flashing can be suppressed in offline mode. No communication takes place in offline mode (e.g., if the MCP connection fails). Via the status bits 24 or 25 in the MCP1StatSend, MCP2StatSend a value can be specified in the output data when the communication is active.
The acknowledgement that the value as been imported, is returned in the same bits in the MCP1StatRec, MCP2StatRec. Following a successful checkback signal, the status bits in send status must be reset.
Example about realization is present on the toolbox.

Temperature monitoring of the MCP:

With MCP firmware V5.01.02 and higher, an increased temperature is signaled back via bit 28 = 1 in MCP1StatRec, MCP2StatRec.

Configuring

Essentially, there are two communication mechanisms for transferring data between the MCP/HHU and PLC. These mechanisms are determined by the connection of the MCP and HHU. With the first mechanism, data are transported via the COM module (840D/810D). The parameter setting is thereby done completely via the MSTT/BHG parameter in the FB1.
In the second case, data are transferred via the PLC operating system (FM-NC) by means of the evaluation of SDB210 (global data) or via the Profibus configuration. The mechanism is parameterized via STEP7 -> Global Data or in HW Config. To allow the basic program to access these data and implement MCP/HHU failure monitoring, the addresses set via SDB210 (global data) must be declared in the FB1 parameters in the basic program.

An overview of the various interfacing options as a function of the NC type used is given below. In each case, the parameter set of FB1 and the valid status information relevant for the respective data transmission method are specified.

If an error is detected due to a timeout monitor, a corresponding entry is made in the diagnostic buffer of the PLC CPU (errors 400260 to 400262). In this case, the input signals from the MCP or from the handheld unit (MCP1In/MCP2In or BHGIn) are initialized with 0. If it is possible to resynchronize the PLC and MCP/HHU, communication is resumed automatically and the error message reset by the BP.

840D: OPI/MPI connection

Communication starts from the PLC BP via the NCK and COM mode, i.e., even a link via the MPI does not require an SDB210. Parameter settings are made via the relevant parameters in FB1.
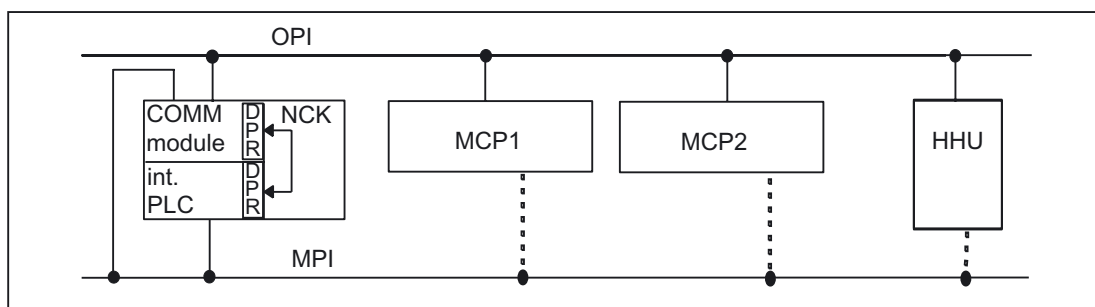
Figure 2-15   840D: OPI/MPI connection

Relevant parameters (FB1)		
MCP		**HHU**
MCPNum=1 or 2 (number of MCPs)		BHG=2 (transfer via COM module)
MCP1In	MCP2In	BHGIn
MCP1Out	MCP2Out	BHGOut
MCP1StatSend	MCP2StatSend	BHGStatSend
MCP1StatRec	MCP2StatRec	BHGStatRec
MCP1BusAdr	MCP2BusAdr	BHGInLen
MCP1Timeout	MCP2Timeout	BHGOutLen
MCP1Cycl	MCP2Cycl	BHGTimeout
MCPMPI = FALSE (OPI), TRUE (MPI)		BHGCycl

Relevant parameters (FB1)		
MCP		HHU
MCP1Stop	MCP2Stop	BHGRecGDNo
MCPBusType=0		BHGRecGBZNo
		BHGRecObjNo
		BHGSendGDNo
		BHGSendGBZNo
		BHGSendObjNo
		BHGMPI = FALSE (OPI), TRUE (MPI)
		BHGStop

Status information		
Available in	Bit No.	Description
MCP1StatSend MCP2StatSend BHGStatSend	4	Syntax error in GD package: Error in parameter set (FB1)
MCP1StatSend MCP2StatSend BHGStatSend	27	Transmitter: Time-out
MCP1StatRec MCP2StatRec BHGStatRec	10	Receiver: Time-out

An error entry is also made in the PLC diagnostic buffer for timeouts (bits 10 and 27),

resulting in the following error messages on the operator interface:

- 400260: MCP 1 failure

- 400261: MCP 2 failure

- 400262: HHU failure

An MSTT or BHG failure is detected immediately after a cold restart even if no data have yet been exchanged between the MSTT/BHG and PLC.
The monitoring function is activated, as soon as all the components report "Ready" after the power-up.

840D: MPI connection for HHU (not for new development)

Communications for HHU by PLC operating system and parameterization via SDB210.

Communication for the MCP is controlled from the PLC BP via the NCK and COM module as described above.
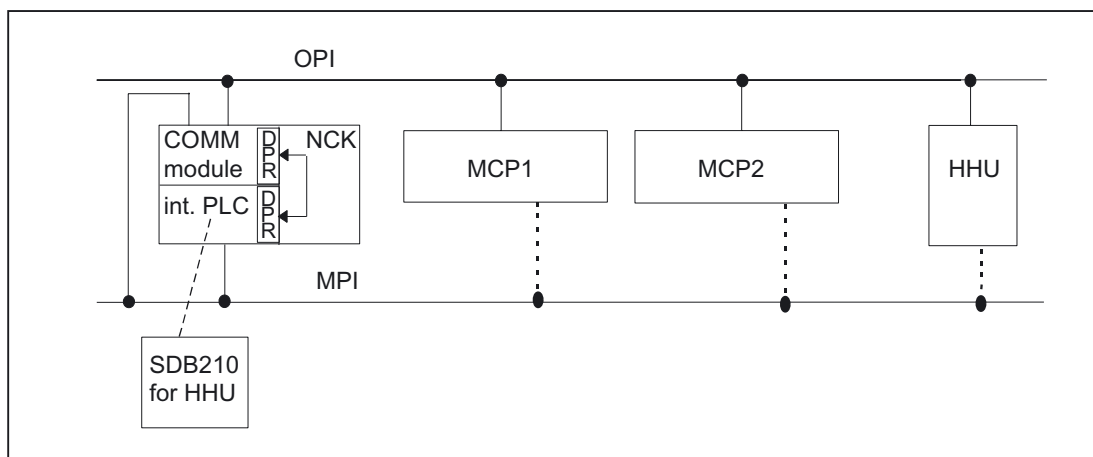
Figure 2-16   840D: MPI connection for HHU

Relevant parameters (FB1):

Communication between the PLC and HHU is implemented through configuring and subsequent loading of SDB210 (Global Data). To allow the basic program to access the HHU data and implement HHU failure monitoring, the addresses set via global data must be declared in FB1 parameters.

Relevant parameters (FB1)	
MCP	HHU (parameterization via SDB210)
Parameter settings are made via the relevant parameters in FB1.	BHG = 1 (transfer via SDB210)
	BHGIn (as parameterized in SDB210)
	BHGOut (as parameterized in SDB210)
	BHGStatRec (as parameterized in SDB210)
	BHGTimeout (as parameterized in SDB210)

Status information (MCP1 and MCP2 see table Status information)		
Available in	Bit No.	Description
BHGStatRec	10	Receiver: Time-out

An error entry is also made in the PLC diagnostic buffer for timeouts,

resulting in error message 400262 on the operator interface: HHU failure

An HHU failure is only recognized if data exchange has taken place previously with the HHU. The first exchange of data with the HHU activates the monitoring function.

MPI connection

Communication starts from the PLC BP via the NCK and COM mode, i.e., even a link via the MPI does not require an SDB210. Parameter settings are made via the relevant parameters in FB1.
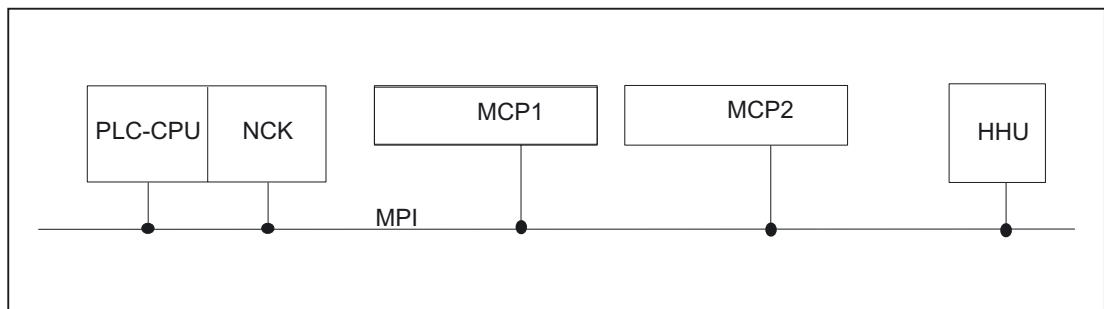
Figure 2-17   MPI connection

Relevant parameters (FB1)		
MCP		**HHU**
MCPNum=1 or 2 (number of MCPs)		BHG=2 (transfer via COM module)
MCP1In	MCP2In	BHGIn
MCP1Out	MCP2Out	BHGOut
MCP1StatSend	MCP2StatSend	BHGStatSend
MCP1StatRec	MCP2StatRec	BHGStatRec
MCP1BusAdr	MCP2BusAdr	BHGInLen
MCP1Timeout	MCP2Timeout	BHGOutLen
MCP1Cycl	MCP2Cycl	BHGTimeout
MCPMPI = TRUE (MPI)		BHGCycl
MCP1Stop	MCP2Stop	BHGRecGDNo
MCPBustype=0		BHGRecGBZNo
		BHGRecObjNo
		BHGSendGDNo
		BHGSendGBZNo
		BHGSendObjNo
		BHGMPI = TRUE (MPI)
		BHGStop

Status information		
Available in	Bit No.	Description
MCP1StatSendMCP2Stat Send BHGStatSend	4	Syntax error in GD package: Error in parameter set (FB1)
MCP1StatSendMCP2Stat SendB HGStatSend	27	Transmitter: Time-out
MCP1StatRec MCP2StatRec BHGStatRec	10	Receiver: Time-out

An error entry is also made in the PLC diagnostic buffer for timeouts (bits 10 and 27),

resulting in the following error messages on the operator interface:

- 400260: MCP 1 failure

    Or

- 400261: MCP 2 failure

- 400262: HHU failure

An MSTT or BHG failure is detected immediately after a cold restart even if no data have yet been exchanged between the MSTT/BHG and PLC.
The monitoring function is activated, as soon as all the components report "Ready" after the power-up.

MPI connection and 810D (SW 4 and lower)

Communications for MCP and HHU by PLC operating system and parameterization via SDB210.
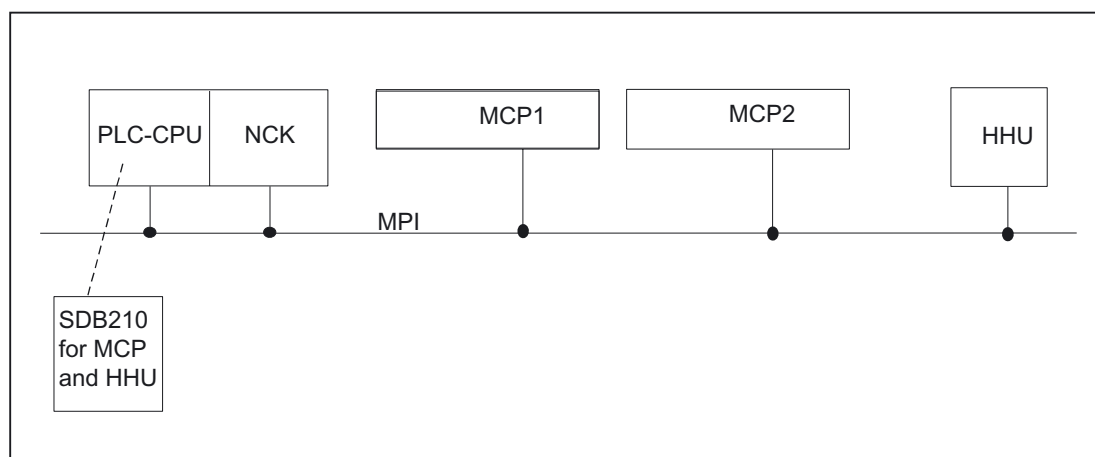
Figure 2-18   MPI connection and 810D

Relevant parameters (FB1):

Communication between the PLC and HHU is implemented through configuring and subsequent loading of SDB210 (Global Data). To allow the basic program to access the HHU data and implement HHU failure monitoring, the addresses set via global data must be declared in FB1 parameters.

Relevant parameters (FB1) (all entries as parameterized in SDB210 global data)		
MCP		HHU
MCPNum=1 or 2 (number of MCPs)		BHG=1 (MPI)
MCP1In	MCP2In	BHGIn
MCP1Out	MCP2Out	BHGOut
MCP1StatRec	MCP2StatRec	BHGStatRec
MCP1Timeout	MCP2Timeout	BHGTimeout

Status information		
Available in	Bit No.	Description
MCP1StatRec MCP2StatRec BHGStatRec	10	Receiver: Time-out

An error entry is also made in the PLC diagnostic buffer for timeouts,

resulting in the following error messages on the operator interface:

- 400260: MCP 1 failure

- 400261: MCP 2 failure

- 400262: HHU failure

An MCP/HHU failure is only detected following a cold restart if the unit has already been involved in data exchange. The first exchange of data with the MCP/HHU activates the monitoring function.

840D: PROFIBUS connection

With an MCP PROFIBUS connection, these components must be considered in the STEP 7 hardware configuration. The addresses must be stored in the input and output log range. These start addresses must also be stored in the pointer parameters of the FB1. The FB1 parameters listed below are used for further parameterization. There is no PROFIBUS variant of the HHU. For this reason, an OPI connection is shown for the HHU in this diagram. PROFIBUS slave address can be stored in MCP1BusAdr and MCPBusAdr.
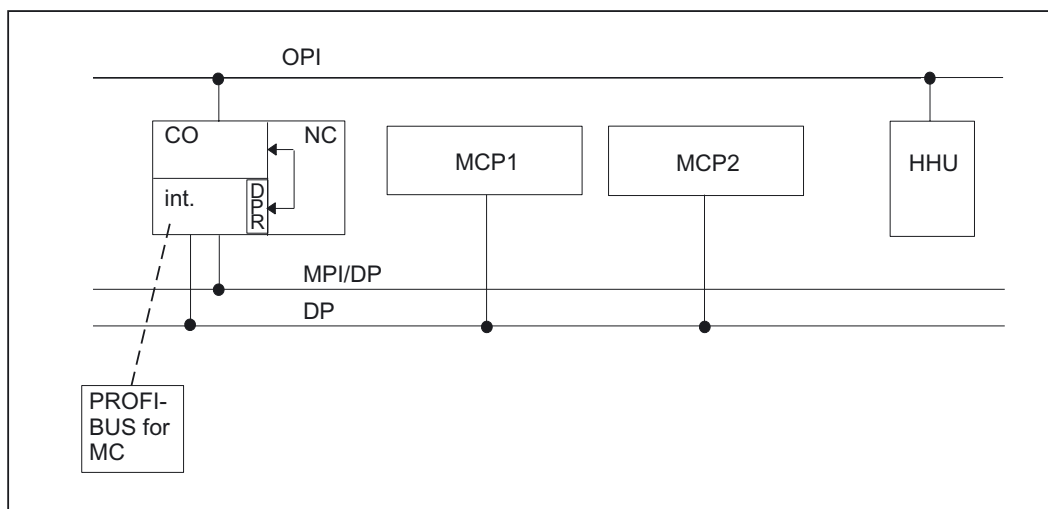
Relevant parameters (FB1)		
MCP		**HHU**
MCPNum = 1 or 2 (number of MCPs)		HHU = 2 (via COM module)
MCP1In	MCP2In	BHGIn
MCP1Out	MCP2Out	BHGOut
MCP1StatSend (n.r.)	MCP2StatSend (n.r.)	BHGStatSend
MCP1StatRec (n.r.)	MCP2StatRec (n.r.)	BHGStatRec
MCP1BusAdr	MCP2BusAdr	BHGInLen
MCP1Timeout (n.r.)	MCP2Timeout (n.r.)	BHGOutLen
MCP1Cycl (n.r.)	MCP2Cycl (n.r.)	BHGTimeout
MCPMPI = FALSE		BHGCycl
MCP1Stop (n.r.)	MCP2Stop (n.r.)	BHGRecGDNo
MCPBusType = 3		BHGRecGBZNo
		BHGRecObjNo
		BHGSendGDNo
		BHGSendGBZNo
		BHGSendObjNo
		BHGMPI = FALSE
		BHGStop

MCP failure switches the PLC to the STOP state. If this is undesirable, OB 82, OB 86 can be used to avoid a stop. Furthermore, the MCP (as Profibus slave) can also be activated/deactivated via the SFC12.
Alarm messages are not generated by the basic program in case of a failure.
M to N is not possible with the MCP version.

2.8 SPL for Safety Integrated

See:
References:
/FBSI/Description of Functions, Safety Integrated

2.9 Assignment overview

2.9.1 Assignment: NC/PLC interface

The assignment of the NC/PLC interface is comprehensively described in:

References:
/LIS/ Lists

2.9.2 Assignment: FB/FC

FB number	FC number	Meaning
1		Basic program
2 - 29		Reserved for Siemens
	1	Reserved for Siemens
	2 - 29	Reserved for Siemens
	30 - 35	See below: ShopMill, ManualTurn
	30 – 127 (siehe Note)[1]	**User area**
30 – 127 (see Note)		**User area**

Note

The actual upper limit of the block number (FB/FC) depends on the active PLC CPU which is located in the selected NCU. See section "Key PLC CPU data for 810D, 840D". For FC and FB assignment, see section "Memory requirements of basic PLC program for 810D, 840D".

ManualTurn

ManualTurn uses FC 30 to 35 and DB 81 to 89.

ManualTurn is an operating system for conventional, cyclecontrolled turning machines. The above FCs and DBs can be used provided that the machine to be configured is not a turning machine with a maximum of two axes and one spindle. If your machine is of this type and if you might require conventional operating methods in addition to CNC functions, then you should not use the FCs and DBs above.

ShopMill

ShopMill uses FC30 to 35 and DB 81 to 89.

ShopMill is an operating system for 2 1/2D milling machines in the workshop environment. The above FCs and DBs can be used provided that the machine to be configured is not a milling machine for 2 1/2D machining operations. However, if you intend to use your machine for applications of this type, then the FCs and DBs should not be used.

2.9.3 Assignment: DB

Note

Only as many data blocks as are required according to the NC machine data configuration are set up.

Overview of data blocks			
DB no.	Designation	Name	Package
1		Reserved for Siemens	BP
4	PLC-MELD	PLC messages	BP
8		Reserved for Siemens	
9	NC-COMPILE	Interface for NC compile cycles	BP
10	NC INTERFACE	Central NC interface	BP
11	Mode group 1	Interface mode group	BP
12		Computer link and transport system interface	
14		Reserved for Siemens	
15		Basic program	
16		PI Service definition	
17		Version ID	
18		Reserved for basic program	
19		HMI interface	
20		PLC machine data	

Overview of data blocks			
DB no.	Designation	Name	Package
21 - 30	CHANNEL 1 ... n	Interface NC channels	BP
31 - 61	AXIS 1 ... m	Interfaces for axes/spindles or free for user assignment	BP
62 - 70		Free for user assignment	
71 - 74		Tool management	BP
75 - 76		M group decoding	
77 - 80		Reserved for Siemens	
81 - 89		See below: ShopMill, ManualTurn	
81 –127 (see Note)		**User area**	

Note

The actual upper limit of the block number (DB) depends on the active PLC CPU which is located in the selected NCU. See section "Key PLC CPU data for 810D, 840D".

ManualTurn

ManualTurn uses FC30 to 35 and DB81 to 89.

ManualTurn is an operating system for conventional, cyclecontrolled turning machines. The above FCs and DBs can be used provided that the machine to be configured is not a turning machine with a maximum of two axes and one spindle. If your machine is of this type and if you might require conventional operating methods in addition to CNC functions, then you should not use the FCs and DBs above.

ShopMill

ShopMill uses FC30 to 35 and DB81 to 89.

ShopMill is an operating system for 2 1/2D milling machines in the workshop environment. The above FCs and DBs can be used provided that the machine to be configured is not a milling machine for 2 1/2D machining operations. However, if you intend to use your machine for applications of this type, then the FCs and DBs should not be used.

2.9.4   Assignment: Timers

Timer No.	Meaning
0 - 9	Reserved for Siemens
10 – 127	User area

Note

The actual upper limit of the block number (timer) depends on the active PLC CPU which is located in the selected NCU. See section "Key data of PLC CPUs for 810D, 840D".

2.10   Memory requirements of basic PLC program for 840D

General

The basic program consists of basic and optional functions. The **basic functions** include cyclic signal exchange between the NC and PLC.
The **options** include, for example, the FCs, which can be used if required.

The table below lists the memory requirements for the basic functions and the options. The data quoted represent guide values, the actual values depend on the current software version.

Memory requirements of blocks with SINUMERIK 840D

Block type no.	Function	Remark	Block size (bytes)	
			Load memory	Working memory
Basic functions in basic program				
FB 1, 15, 16, 17, 18		Must be loaded	3616	3052
FC 1, 2, 3, 4, 11, 20		Must be loaded	7208	6608
DB 4, 5, 7, 8, 17, 19		Must be loaded	2490	966
DB 2, 3, 6		Are generated by the BP	992	812
OB 1, 40, 100		Must be loaded	490	282
		Total	14796	11720

PLC/NCK, PLC/HMI interface				
DB 10	PLC/NCK signals	Must be loaded	318	262
DB 11	Signals PLC/Mode group	Is generated by BP	80	44
DB 21, 30	PLC/channel signals	Are generated by BP as a function of NCMD	352 each	316 each
DB 31, ...61	PLC/axis or spindle signals	Are generated by BP as a function of NCMD	180 each	144 each

Basic program options				
	Machine control panel			
FC 19	Transfer of MCP signals, M variant	Must be loaded when M variant of MCP is installed	1498	1258
FC 25	Transfer of MCP signals, T variant	Must be loaded when T variant of MCP is installed	1358	1160
FC 24	Transfer of MCP signals, slim variant	Must be loaded when slim variant of MCP is installed	1358	1160
FC 26	Transfer of MCP signals, HPU variant	Must be loaded for HPUs	1358	1160
FC 14	MPI/OPI transfer	Must be loaded if MCPNum > 0	942	802
	Handheld unit			
FC 13	Display control HHU	Can be loaded for handheld units	1264	1044
	Error/operating messages			
FC 10	Acquisition FM/BM	Load when FM/BM is used	1572	1350
	ASUB			
FC 9	ASUB start	Load when PLC ASUBs are used	656	538

Basic program options				
	Concurrent axes			
FC 15	Positioning of linear/rotary axes	Load for axis positioning by PLC	656	546
FC 16	Positioning of indexing axes	Load for axis positioning by PLC	674	560
	Star/delta changeover			
FC 17	Star/delta switchover of MSD	Load for star/delta switchover	612	494
	Spindle control			
FC 18	Spindle control	Load for spindle control from PLC	826	676
	PLC/NC communication			
FB 2	Read NC variable	Load for Read NC variable	396	224
DB n1)	Read NC variable	One instance DB per FB 2 call	426 each	270 each
FB 3	Write NC variable	Load for Write NC variable	396	224
DB m1)	Write NC variable	One instance DB per FB 3 call	426 each	270 each

Basic program options				
FB 4	PI services	Load for PI services	334	214
DB o1)	PI services	One instance DB per FB 4 call	234 each	130 each
DB 16	PI services description	Load for PI services	1190	408
FB 5	Read GUD variables	Load for PI services	532	365
DB p	Read GUD variables	One instance DB per FB 5 call	308 each	166 each
FB 6	General communication	Load for Read/write NC variables and PI services	5986	5228
DB 15	General communication	Instance DB for FB 6	440	172
	Tool management			
FC 6	Basic function	Load for tool management option	1382	1182
FC 7	Transfer function turret	Load for tool management option	530	430
FC 8	Transfer function	Load for tool management option	1002	834
FC 22	Direction selection	Load if direction selection is required	404	300
DB 71	Loading locations	Generated by BP as a function of NC MD	30*B	30*B
DB 72	Spindles	Generated by BP as a function of NC MD	48*Sp	48*Sp
DB 73	Turret	Generated by BP as a function of NC MD	44*R	44*R
DB 74	Basic function	Generated by BP as a function of NC MD	(B+Sp+R)*20	(B+Sp+R)*20
	Compile cycles			
DB 9	Interface PLC compile cycles	Is generated by BP as a function of NC option	472	436

1): DB number must be specified by PLC user

Example:

Based on the memory requirements in the table above, the memory requirements have been determined for two sample configurations (see table below).

Block type no.	Function	Remark	Block size (bytes)	
			Load memory	Working memory
Minimum configuration (1 spindle, 2 axes and TMCP)				
See above	Basic program, base		14796	11720
	Interface DBs		1290	1054
	MCP		2300	1962
		Total	18386	14736

Block type no.	Function	Remark	Block size (bytes)	
			Load memory	Working memory
Maximum configuration (2 channels, 4 spindles, 4 axes, TMCP)				
See above	Basic program, base		14796	11720
See above	Interface DBs		2542	2090
See above	MCP		2300	1962
See above	Error/operating messages		1572	1350
See above	ASUBs	1 ASUB initiation	656	538
See above	Concurrent axis	For 2 turrets	674	560
See above	PLC/NC communication	1 x read variable and 1 x write variable	8070	6388
See above	Tool management	2 turrets with one loading point each	3430	2854
See above	Compile cycles		472	436
		Total	34512	27898

2.11 Supplementary conditions and NC VAR selector

2.11.1 Supplementary conditions

2.11.1.1 Programming and parameterizing tools

Basics

Hardware

Programming devices or PCs with the following equipment are required for the PLCs installed on the 810D and 840D:

	Minimum	Recommendation
Processor	80486	Pentium
RAM (MB)	32	Or more
Hard disk, free capacity (MB)	200	> 400
Interfaces	MPI incl. cable Memory card	
Graphics	VGA or TIGA	SVGA
Mouse	yes	

	Minimum	Recommendation
Operating system	Windows 95/98/NT	Windows 95/98/NT or higher
	STEP 7 and higher	STEP 7 and higher
	Version 4	Version 5.1

The **STEP7 package for the S7300** can be obtained and installed on equipment meeting the above requirements in cases where the package has not already been supplied with the programming device.

The following functions are possible with this package:

- Programming

  - Editors and compilers for STL (complete scope of the language incl. SFB/SFC calls), LAD, FBD

  - Creation and editing of assignment lists (symbol editor)

  - Data block editor

  - Input and output of blocks ON/OFF line

  - Insertion of modifications and additions ON and OFF line

  - Transfer of blocks from programming device to the PLC and vice versa

- Parameterizing

  - Parameterizing tool **HW Config** for CPU and I/O device parameterization

  - **Communication Configuration** configuration tool for setting the CPU communication parameters

  - Output of system data such as hardware and software version, memory capacity, I/O expansion/assignment

- Testing and diagnostics (ONLINE)

  - Variable status/forcing (I/Os, flags, data block contents, etc.)

  - Status of individual blocks

  - Display of system states (ISTACK, BSTACK, system status list)

  - Display of system messages

  - PLC STOP/complete restart/overall reset triggering from the programming device

  - Compress PLC

- Documentation

  - Printout of individual or all blocks

  - Allocation of symbolic names (also for variables in data blocks)

  - Input and output of comments within each block

  - Printout of test and diagnostics displays

  - Hardcopy function

  - Cross-reference list

  - Program overview

  - Assignment plan I/O/M/T/C/B/P/D

- Archiving of utility routines

  – Allocation of the output statuses of individual blocks

  – Comparison of blocks

  – Rewiring

  – STEP 5 -> STEP 7 converter

- Option packages

  – Programming in S7–HIGRAPH, S7–GRAPH, SCL.

    These packages can be ordered from the SIMATIC sales department.

  – Additional packages for configuring modules
    (e.g., CP3425 -> NCM package)

Note

More information about possible functions can be found in SIMATIC catalogs and STEP 7 documentation.

2.11.1.2 SIMATIC documentation required

References:
SIMATIC S 7 System Overview
S7-300, CPU 314, CPU 315-2DP Operation List
Programming with STEP 7
STEP 7 User's Guide
STEP 7 Programming Manual; Designing User Programs
STEP 7 Reference Manual; STL Instruction List
STEP 7 Reference Manual; LAD Ladder Diagram
STEP 7 Reference Manual; Standard and System Functions
STEP 7 Manual: Conversion of STEP 5 Programs
STEP 7 General Index
CPU 314, CPU 315-2DP Manual

2.11.1.3 Relevant SINUMERIK documents

References:
/IAD/840D, 611D Commissioning Manual; PLC Interface
/IAG/810D, 611D Commissioning Manual; PLC Interface
/BH/Operator Components Manual (HW)/840D/FM NC/810D
/FB/840D, 810D Function Manual
/LIS/840D/810 D Lists
/FBP/PLC C Programming

2.11.2   NC VAR selector

2.11.2.1   Overview

General

A catalog with an optional catalog name must be set up via the Windows Explorer. The selected data of the VAR selector (data.VAR and data.AWL (STL)) must be stored in this catalog. An "Insert", "External source" into the STEP 7 machine project for the "Data.AWL" file must then be carried out via the STEP 7 manager (V3 and higher). The source container must be selected in the manager for this purpose. This action stores this file in the project structure. Once the file has been transferred, these AWL (STL) files must be compiled with STEP 7.

The PC application "NC VAR selector" fetches the addresses of required NC variables and processes them for access in the PLC program (FB 2/FB 3). This enables the programmer to select NC variables from the entire range of NC variables, to store this selection of variables, to edit them by means of a code generator for the STEP 7 compiler and finally to store them as an ASCII file (*.AWL) in the machine CPU program. This process is shown in the figure "NC VAR selector".

Note

The latest NC VAR selector can be used for each NC software version (even earlier versions). The variables can also be selected from the latest list for earlier NC software versions. The data content in DB 120 (default DB for variables) does not depend on the software version, i.e., selected variables in an older software version must not be reselected when the software is upgraded.
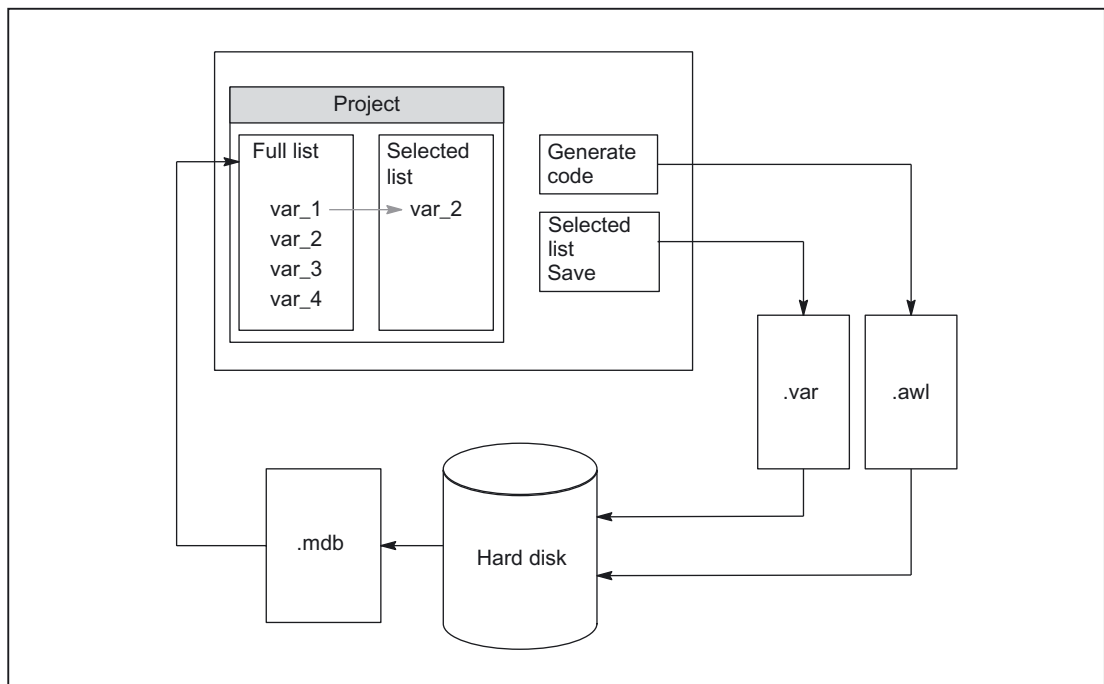
Figure 2-19 NC VAR selector

After the "NC VAR selector" application has been started, select a list of variables of an NC variant (hard disk → file Ncv.mdb) to display all the variables contained in this list in a window.

In SW V6.3 and higher, the variable lists ncv*.mdb are separated according to:	
NC variables including machine and setting data:	ncv_NcData.mdb
Machine data for 611D drive:	ncv_611d.mdb
Machine data for 611D linear drive:	ncv_611dLinear.mdb
Machine data of the 611D drive, Performance 2:	ncv_611d_P2.mdb
Machine data of the 611D linear drive, Performance 2:	ncv_611d_P2Linear.mdb
Machine data of the hydraulic drive:	ncv_Hydraulics.mdb

The user can also transfer variables to a second list (separate window). This latter selection of variables can then be stored in an ASCII file or edited as a STEP 7 source file (.awl) and stored.

Once he has generated a PLC data block by means of the STEP 7 compiler, the programmer is able to read or write NCK variables via the basic program function blocks "PUT" and "GET" using the STEP 7 file.

The list of selected variables is also stored as an ASCII file (file extension .var).

The variable list supplied with the "NC VAR selector" tool is adapted to the current NC software version. This list does not contain any variables (GUD variables) defined by the user. These variables are processed by the function block FB 5 in the basic program.

Note

The latest version of the "NC VAR selector" is capable of processing all previous NC software versions. It is not therefore necessary to install different versions of the "NC VAR selector" in parallel.

System features, supplementary conditions

The PC application "NC VAR selector" requires Windows 95 (or later operating system).

The assignment of names to variables is described in:
References:
/LIS/ Lists; Chapter: Variables,
or in the Variables Help file (integrated in NC VAR selector).

2.11.2.2   Description of Functions

Overview

The figure below illustrates how the NC VAR selector is used within the STEP 7 environment.
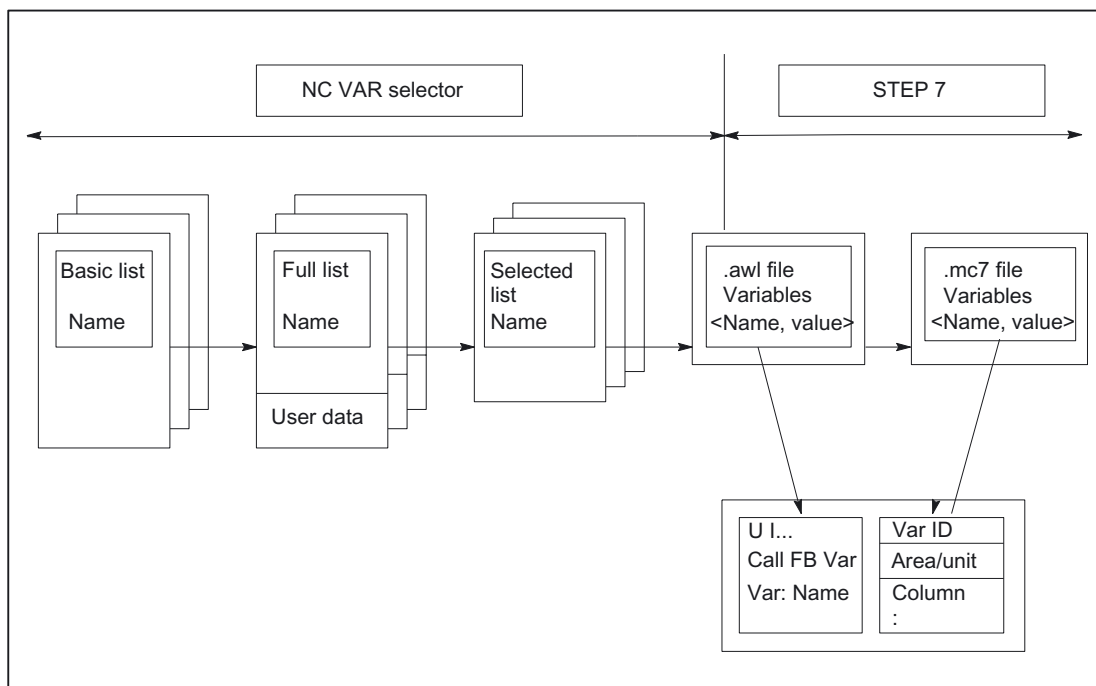
Figure 2-20   Application of NC VAR selector in the STEP 7 environment

The NC VAR selector is used to generate a list of selected variables from a list of variables and then to generate an **.awl** file that can be compiled by the STEP 7 compiler.

Note

A *.awl file contains the names and alias names of the NC variables, as well as information about their address parameters.

Any data block generated from this file will only contain the address parameters (10 bytes per parameter).

- The generated data blocks must always be stored in the machinespecific file storage according to STEP 7 specifications.
- To ensure that the parameterization of the GET/PUT (FB 2/3) blocks with respect to NC addresses can be implemented with symbols, the freely assignable, symbolic name of the generated data block must be included in the STEP 7 symbol table.

Basic display/Basic menu

After the NC VAR selector has been selected (started), the basic display with all input options (upper menu bar) appears on the screen. All other displayed windows are placed within the general window.
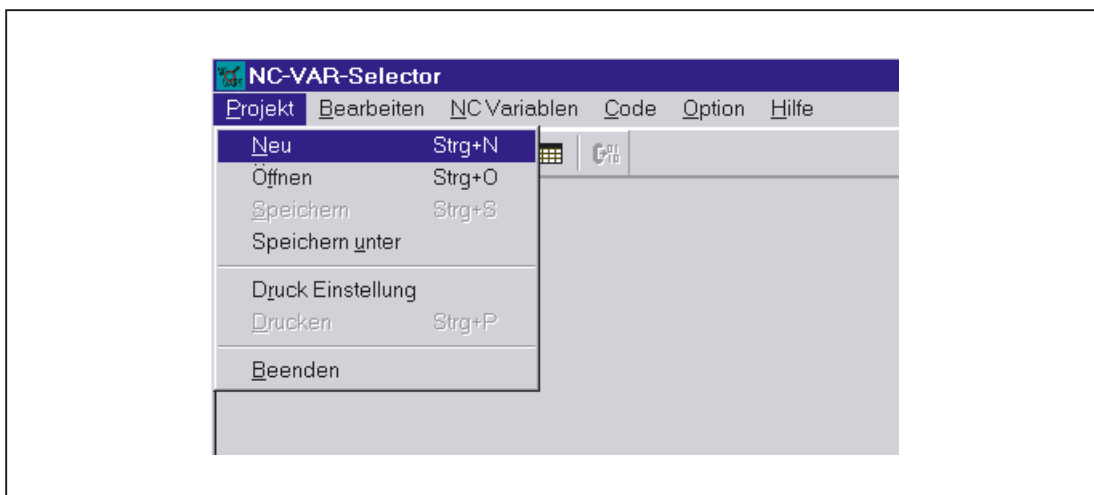
Figure 2-21   Basic display with basic menu

Project menu item

All operator actions associated with the project file (file of selected variables) are performed under this menu item.

Terminating the application

The application can be terminated by selecting the "End" option under the "Project" menu item.

Creating a new project

A new project (new file for selected variables) can be set up under the "Project" menu item.

A window is displayed for the selected variables when "NEW" is selected. The file selection for the NC variable list is then displayed after a prompt (applies only if the NC variable list is not already open).

			Projekt: unknown	
Bereich	Baustein	VariablenName		S7 Alias Name
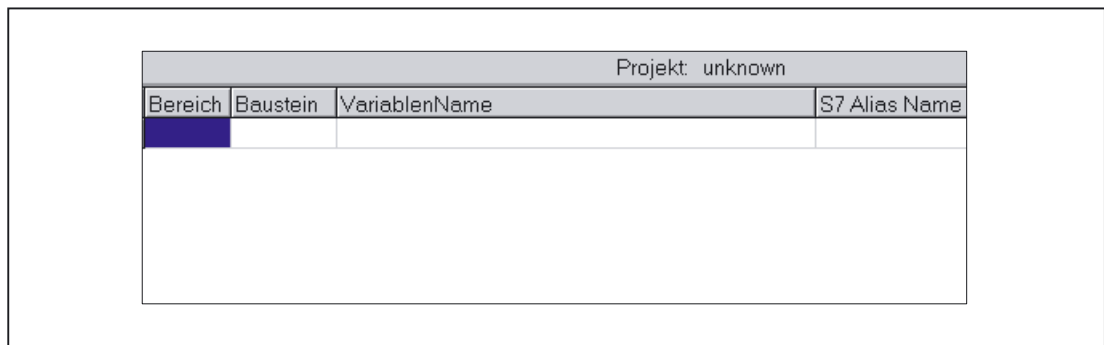
Figure 2-22   Window with selected variables for new project

The selected variables are displayed in a window.

Opening an existing project

Select "Open" under the "Project" menu item to open an existing project (variables already selected). A file selection window is displayed allowing the appropriate project with extension ".var" to be selected.
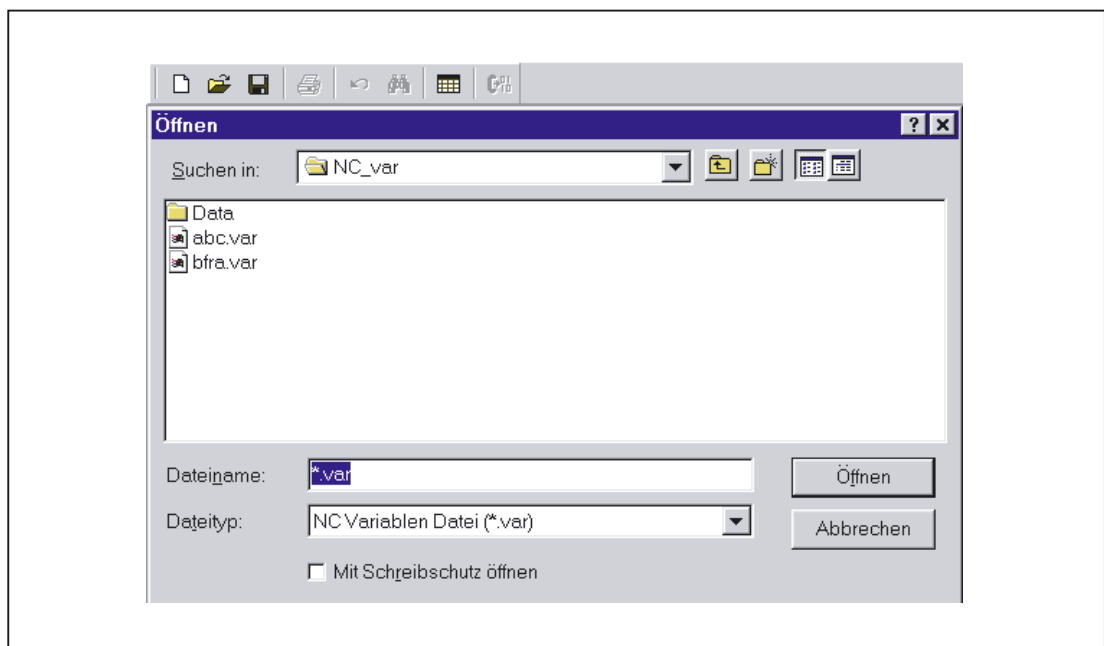
Figure 2-23   Selection window for existing projects

If, after selection of the project, new variables are to be added, a complete list of NCK variables must be selected (see: Selecting complete list). No complete list need be called if the user only wishes to delete variables from the project.

Storing a project

The variable list is stored using the "Project", "Save" or "Save As...." menu items.

"Save" stores the variable list under a path, which is already specified. If the project path is not known, then the procedure is as for "Save As....".

"Save As..." displays a window in which the path for the project to be stored can be specified.

Printing a project

The "Print" command under the "Project" menu item can be selected to print a project file. The number of lines per page is selected under the "Print Setting" menu item. The default setting is 77 lines.

Edit menu item

The following operator actions are examples of those, which can be carried out directly with this menu item:

- Transfer variables
- Delete variables
- change alias names
- Find variables

These actions can also be canceled again under Edit.

Undoing actions

Operator actions relating to the creation of the project file (transfer variables, delete variables, change alias names) can be undone in this menu.

NC variables menu item

The basic list of all variables is saved in the NC Var selector path Data\Swxy (xy stands for SW no., e.g., SW 5.3:=xy=53). This list can be selected as an NC variables list. The available variable lists are provided thematically.

Selecting an NC variable list

A list of all the NC variables for an NC version can now be selected and displayed via the "NC Variable List", "Select" menu item.
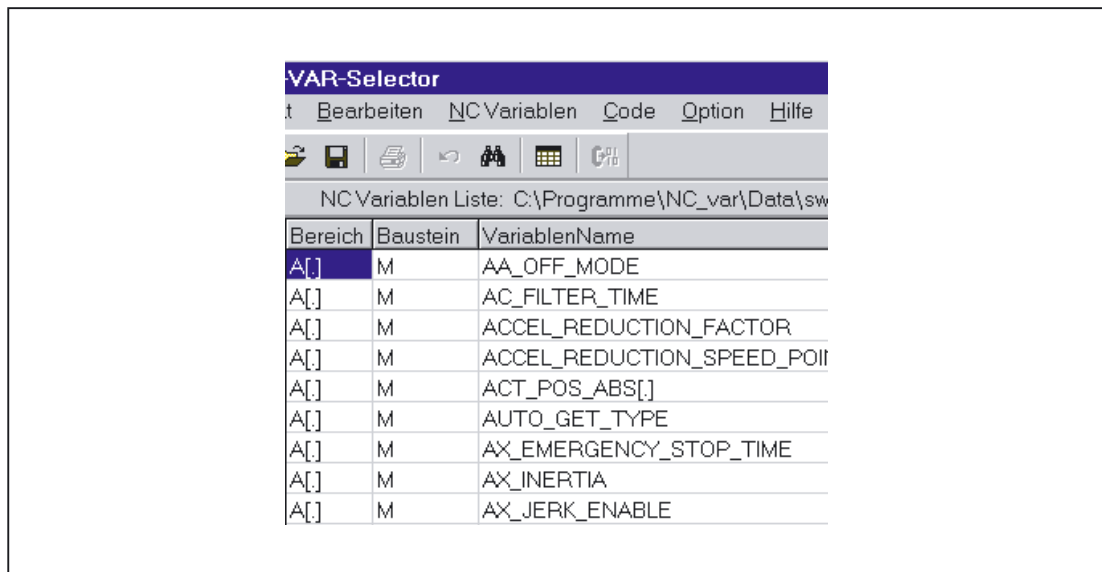
Figure 2-24   Window with selected Complete List

The field variables (e.g., axis area, T area data, etc.) are indicated by means of brackets ([.]). Additional information must be specified here. When the variables are transferred to the project list, the additional information required is requested.

Displaying subsets

Double-click on any table field (with the exception of variable fields) to display a window in which filter criteria can be preset.
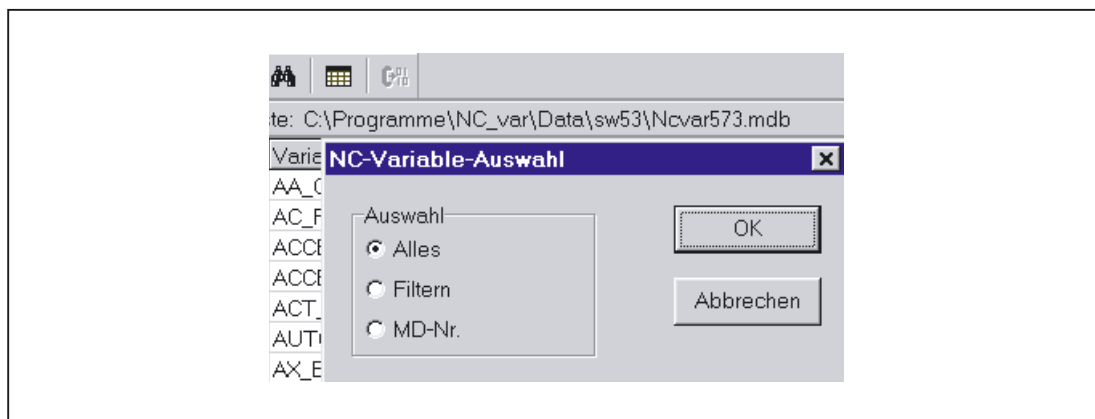
Figure 2-25   Window with filter criteria for displaying list of variables

There are three options:

- Display all data
- Input area, block and name (incl. combinations)
- Display MD/SE data number

The following wildcards can also be used:

*	To extend the search criterion as required

Example search criteria

Name search criterion: CHAN*	Found:	CHAN_NAME
		chanAlarm
		chanStatus
		channelName
		chanAssignment

Selecting variables

A variable is selected by means of a simple mouse click and transferred to the window of selected variables by double-clicking.

This action can also be undone under the "Edit" menu item.

Alias name

The variable names provided can be up to 32 characters in length. To make variables clearly identifiable in the data block to be generated, several ASCII characters are added to the selected name. However, the STEP 7 compiler recognizes only 24 ASCII characters as an unambiguous STEP 7 variable. Since it cannot be precluded that variable names can only be differentiated by the last 8 character positions, **ALIAS** names are used for names, which are too long. When a variable is selected, the length of the STEP 7 name to be used is therefore checked. If the name is longer than 24 characters, the user must enter an additional name, which is then used as the alias.

In this case, the user must ensure that the alias name is unambiguous.

Alias input can always be activated by the user in the "Options" menu.
An alias name can then be entered every time a variable is transferred.

It is also possible to edit alias names at a later point in time by doubleclicking on the S7 variable name field. This action can also be undone under the "Edit" menu item.

Figure 2-26   Screen with complete list and selected variables

Scrolling

A scroll bar is displayed if it is not possible to display all variables in the window. The remaining variables can be reached by scrolling (page up/down).

Variables in multi-dimensional structures

If variables are selected from multidimensional structures, then the column and/or line number as well as the area number must be entered so that the variables can be addressed. The required numbers can be found in the NC variables documentation.

References:
/LIS/ **Lists;** Variables

By entering a zero (0) as the block number or the line or column index, it is possible to use the variable in the S7 PLC as a pointer to these data. When reading or writing these data via the functions "PUT" and "GET", the optional parameters "UnitX", "ColumnX" and "LineX" must be filled with the necessary information.
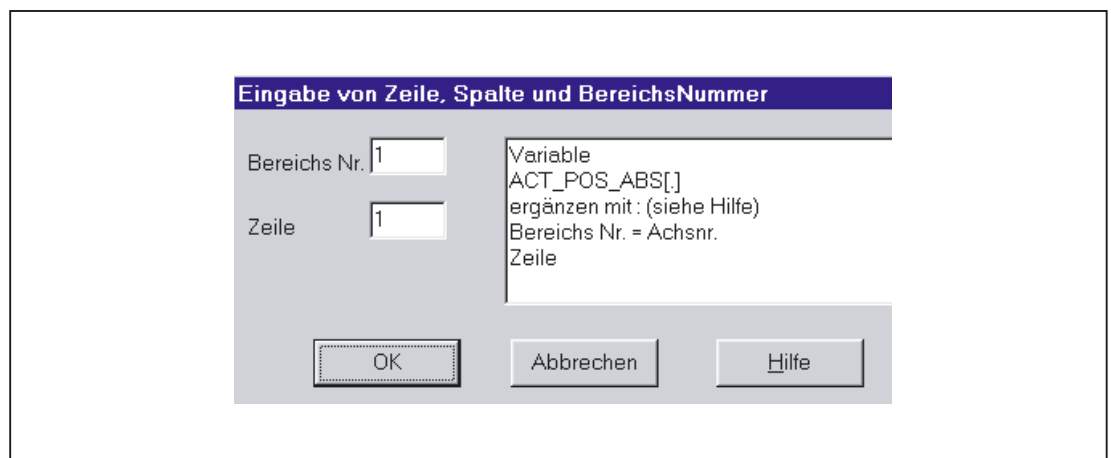
Figure 2-27   Entry field for line, column and block no.

Delete variables

Variables are deleted in the window of selected variables by selecting the appropriate variables (single mouse click) and pressing the "Delete" key. No deletion action is taken with the doubleclick function. It is possible to select several variables for deletion (see "Selecting variables").

This action can also be undone under the "Edit" menu item.

Note

Deleting of variables results in a change of the absolute addresses of the pointer structures to the variables. When changing the variable selection, it is, therefore, absolutely necessary to **generate** one or several **text files of all user blocks** **prior to the change**. This is the only way to ensure that the assignment of the variables in FB "GET" or FB "PUT" remains correct, even after recompilation.

Storing a selected list

Once variables have been selected, they can be stored under a project name. The files are stored on a projectspecific basis.

A window is displayed for the file to be stored. The project path and name for the file must be selected in the window.
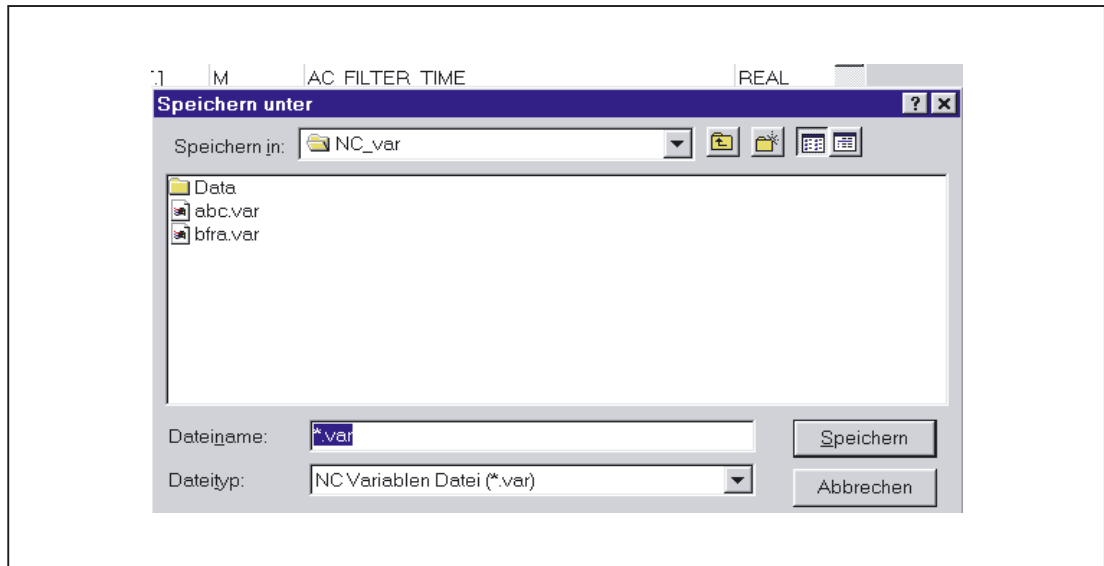
Figure 2-28   Window for project path and name of file to be stored

Code generation

This menu item contains three selection options:

1. Settings (input of data block number to be generated) and other settings

2. Generate (create data block)

3. In the STEP 7 project (transferring the data block to a STEP 7 project)

Settings

Under this menu item, the DB number and the symbol for this DB number for which the code is created is entered.

Under the "Unit System" tab, a selection is made to determine how the unit system variables are calculated in the PLC.

Under the "Generate" tab, the project creation is defined for the relevant target system.

Generate

Under this menu item, the STEP 7 file from the selected variable list with extension ".awl" is set.

A file is generated when "Select" is clicked:

An **.awl** file that can be used as an input for the STEP 7 compiler.

A window opens, in which path and name for the .awl file to be generated must be specified for the file to be saved.

In a STEP7 project

The generated AWL file is transferred to a selectable SIMATIC project (program path) and compiled. Furthermore, the symbol can also be transferred. This function will be available as of STEP7 Version 5.1 and NCVar Selector 6.04.05. This process take some time due to the time require by STEP7. Before transferring a new AWL file, the file window of the AWL file must be closed in the LAD/FUP/AWL Editor.

Option menu item

The following can be selected under the "Option" menu item:

- The current language
- The mode for alias input (always/>24 characters)

Help menu item

The information below can be viewed by selecting the corresponding submenu item:

- The Operator's Guide
- The Description of Variables

The copyright and the version number can also be displayed.

2.11.2.3 Startup, installation

The Windows application "NC Var selector" is installed using the SETUP program supplied with the package.

2.12 Block descriptions

2.12.1 FB 1: RUN_UP Basic program, startup section

Description of Functions

The synchronization of NCK and PLC is performed during startup. The data blocks for the NC/PLC user interface are created with reference to the NC configuration defined in the machine data and the most important parameters verified for plausibility. In the event of an error, FB 1 passes an error identifier to the diagnostics buffer and switches the PLC to the STOP state.

To enable an orderly startup of the control, it is vital to synchronize the NCK and PLC, as these systems have their own types of powerup procedure. During startup routine, therefore, the CPUs perform "subsidiary startup functions" and exchange ID information to ensure that the procedure is functioning correctly.
Since the startup procedure is asynchronous, it is unavoidable that one CPU may have to "wait" until the other has "caught up". This is automatically managed by the basic program.

The PLC 314 and PLC 315-2DP only know the start-up type cold restart. A warm restart is not provided, i.e. following system initialization, the operating system runs organization block OB100 and always commences cyclic execution at the start of OB1.

Users need only supply the FB parameters that are relevant to their applications. The preset values in the associated instance DB 7 do not need to be assigned. The block can only be called in OB 100.

Output parameters

The output parameters in FB 1 provide the PLC user with information about the control system configuration. These data can also be accessed in the cyclic program section.

There are two access options:

1. Direct access to the DB 7 data block (instance of the FB 1) in symbolic format (e.g., L gp_par.MaxChan; in this case, gp_par is the symbolic name of the DB 7)

2. Assignment of a flag; during parameterization of the FB 1, the data element is assigned to the relevant parameter (e.g., MaxChan:=MW 20) Information about the maximum number of channels can then be polled in memory word 20 in the rest of the user program.

Note

An additional SDB210 must be generated via the STEP7 Communication Configuration tool for the **operator components** connected on the **MPI interface**.
The corresponding procedure is explained in the Installation and Startup Guide.

Declaration SINUMERIK 810D, 840D

Code			Comment
FUNCTION_BLOCK FB 1			
VAR_INPUT			
MCPNum:	INT:= 1;		//0: No MCP
			//1: 1 MCP (default)
			//2: 2 MCPs
MCP1In:	POINTER;		//Start addr. MCP 1 input signals
MCP1Out:	POINTER;		//Start addr. MCP 1 output signals
MCP1StatSend:	POINTER;		//Status DW for sending MCP 1
MCP1StatRec:	POINTER;		//Status DW for receiving MCP 1
MCP1BusAdr:	INT:= 6;		//Default
MCP1Timeout:	S5TIME:= S5T#700MS;		
MCP1Cycl:	S5TIME:= S5T#200MS;		
MCP2In:	POINTER;		//Start addr. MCP 2 input signals
MCP2Out:	POINTER;		//Start addr. MCP 2 output signals
MCP2StatSend:	POINTER;		//Status DW for sending MCP 2
MCP2StatRec:	POINTER;		//Status DW for receiving MCP 2
MCP2BusAdr:	INT ;		
MCP2Timeout:	S5TIME:= S5T#700MS;		
MCP2Cycl:	S5TIME:= S5T#200MS;		
MCPMPI:	BOOL:= FALSE;		
MCP1Stop:	BOOL:= FALSE;		
MCP2StopI:	BOOL:= FALSE;		
MCP1NotSend:	BOOL:= FALSE;		
MCP2NotSend:	BOOL:= FALSE;		
MCPSDB210:	BOOL:= FALSE;		
MCPCopyDB77:	BOOL:= FALSE;		
MCPBusType:	BYTE = 0;		
HHU:	INT ;		//Handheld unit interface
			//0: No HHU
			//1: HHU on MPI
			//2: HHU on OPI
BHGIn:	POINTER;		//Transmit data of the HHU
BHGOut:	POINTER;		//Receive data of the HHU
BHGStatSend:	POINTER;		//Status DW for sending HHU
BHGStatRec:	POINTER;		//Status DW for receiving HHU
BHGInLen:	BYTE:= B#16#6;		//Input 6 bytes
BHGOutLen:	BYTE:= B#16#14;		//Output 20 bytes
BHGTimeout:	S5TIME:= S5T#700MS;		
BHGCycl:	S5TIME:= S5T#100MS;		
BHGRecGDNo:	INT:= 2;		
BHGRecGBZNo:	INT:= 2;		
BHGRecObjNo:	INT:= 1;		

Code		Comment
BHGSendGDNo:	INT:= 2;	
BHGSendGBZNo:	INT:= 1;	
BHGSendObjNo:	INT:= 1;	
BHGMPI:	BOOL:= FALSE;	
BHGStop:	BOOL:= FALSE;	
BHGNotSend:	BOOL:= FALSE;	
NCCyclTimeout:	S5TIME:= S5T#200MS;	
NCRunupTimeout:	S5TIME:= S5T#50S;	
ListMDecGrp:	INT:= 0;	
NCKomm:	BOOL:= FALSE;	
MMCToIF:	BOOL:= TRUE;	
HWheelMMC:	BOOL:= TRUE;	//Handwheel selection via HMI
MsgUser:	INT:= 10;	//Number of user areas in DB 2
UserIR:	BOOL:= FALSE;	//User programs in OB 40,
		//Observe local data expansion!
IRAuxfuT:	BOOL:= FALSE;	//Evaluate T function in OB 40
IRAuxfuH:	BOOL:= FALSE;	//Evaluate H function in OB 40
IRAuxfuE:	BOOL:= FALSE;	//Evaluate DL function in OB 40
UserVersion:	POINTER;	//Pointer to string variable indicated in
		//version display

```
END_VAR


VAR_OUTPUT
```

MaxBAG:	INT ;	
MaxChan:	INT ;	
MaxAxis:	INT ;	
ActivChan:	ARRAY[1..10] OF BOOL;	
ActivAxis:	ARRAY[1..31] OF BOOL;	
UDInt :	INT ;	
UDHex:	INT ;	
UDRea l :	INT ;	

```
END_VAR
```

Description of formal parameters SINUMERIK 810D, 840D

The table below lists all formal parameters of the RUN_UP function for 810D, 840D:

Signal	I/O	Type	Value range	Remark
MCPNum	I	INT	0 to 2	Number of active MCPs 0: No MCP installed
MCP1In MCP2In	I	POINTER	I0.0 to I120.0 or F0.0 to F248.0 or DBn DBX0.0 to DBXm.0	Start address for input signals of relevant machine control panel
MCP1Out MCP2Out	I	POINTER	Q0.0 to Q120.0 or F0.0 to F248.0 or DBn DBX0.0 to DBXm.0	Start address for output signals of relevant machine control panel
MCP1StatSend MCP2StatSend	I	POINTER	Q0.0 to Q124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address for status double word for data transmission to the machine control panel: DW#16#08000000: Time-out, otherwise 0
MCP1StatRec MCP2StatRec	I	POINTER	Q0.0 to Q124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address for status double word for data reception by machine control panel: DW#16#00000400: Time-out, otherwise 0
MCP1BusAdr MCP2BusAdr	I	INT	1...15	Bus address of machine control panel
MCP1Cycl MCP2Cycl	I	S5time	Recommendation: 200 ms	Time reference for cyclic updating of signals to machine control panel
MCPMPI	I	BOOL		1: All machine control panels connected to the MPI bus (without GD parameterization)
MCP1Stop MCP2Stop	I	BOOL		0: Start transfer of machine control panel signals 1: Stop transfer of machine control panel signals
MCP1NotSend MCP2NotSend	I	BOOL		0: Send and receive operation activated 1: Receive machine control panel signals only
MCPSDB210	I	BOOL		0: No SDB 210 for MCP 1: Activate timeout monitors on SDB 210 for MCP
MCPCopyDB77	I	BOOL		1: Copy between DB 77 and MCP pointers on DB 7 Can only be used with standard SDB 210 configuration on DB 77.
MCPBusType	I	BYTE		0: MPI or OPI b#16#33: PROFIBUS for MCP1 and MCP2 b#16#55: MSTT (i.e. Ethernet)

Signal	I/O	Type	Value range	Remark
HHU	I	INT		Handheld unit interface: 0: No HHU 1: HHU to MPI with SDB 210 configuration (for SW V3.x): 2: HHU to OPI or MPI if FB 1 parameter BHGMPI is also set to TRUE (SW V4.x and higher)
BHGIn	I	POINTER	I0.0 to I124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address PLC receive data from HHU
BHGOut	I	POINTER	Q0.0 to Q124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address PLC transmit data to HHU
BHGStatSend	I	POINTER	Q0.0 to Q124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address for status double word for transmitting data to HHU: DW#16#08000000: Time-out, otherwise 0
BHGStatRec	I	POINTER	Q0.0 to Q124.0, F0.0 to F252.0 or DBn DBX0.0 to DBXm.0	Start address for status double word for receiving data from HHU: DW#16#00000400: Time-out, otherwise 0
BHGInLen	I	BYTE	HHU default: B#16#6 (6 Byte)	Quantity of data received from handheld unit
BHGOutLen	I	BYTE	HHU default: B#16#14 (20 Byte)	Quantity of data transmitted to handheld unit
BHGTimeout	I	S5time	Recommendation: 700 ms	Cyclic sign-of-life monitoring for handheld unit
BHGCycl	I	S5time	Recommendation: 100 ms	Time reference for cyclic updating of signals to handheld unit
BHGRecGDNo	I	INT	HHU default: 2	Receive GD circle no.
BHGRecGBZNo	I	INT	HHU default: 2	Receive GI no.
BHGRecObjNo	I	INT	HHU default: 1	Object number for receive GI
BHGSendGDNo	I	INT	HHU default: 2	Transmit GD circle no.
BHGSendGBZNo	I	INT	HHU default: 1	Transmit GI no.
BHGSendObjNo	I	INT	HHU default: 1	Object number for transmit GI
BHGMPI	I	BOOL		1: Handheld unit coupled to MPI (without SDB 210 config.) Parameter HHU must be set to 2.
BHGStop	I	BOOL		0: Start transmission of handheld unit signals 1: Stop transmission of handheld unit signals

Signal	I/O	Type	Value range	Remark
BHGNotSend	I	BOOL		0: Send and receive operation activated
				1: Receive HHU signals only (SW 4 and higher)
NCCyclTimeout	I	S5time	Recommendation: 200 ms	Cyclic sign-of-life monitoring NCK
NCRunupTimeout	I	S5time	Recommendation: 50 s	Powerup monitoring NCK
ListMDecGrp	I	INT	0...16	Activation of expanded M group decoding
				0: not active
				1...16: Number of M groups
NCKomm	I	BOOL		PLC NC communications services (FB 2/3/4/5/7: Put/Get/PI_SERV/GETGUD)
				TRUE: active
MMCToIF	I	BOOL		Transmission of MMC/HMI signals to interface (modes, program control, etc.)
				TRUE: Active
HWheelMMC	I	BOOL		TRUE: Handwheel selection via MMC/HMI
				FALSE: Handwheel selection via user program
MsgUser	I	INT	0...32	Number of user areas for messages (DB 2)
UserIR	I	BOOL		Local data expansion OB 40 required for processing of signals from user
IRAuxfuT	I	BOOL		Evaluate T function in OB 40
IRAuxfuH	I	BOOL		Evaluate H function in OB 40
IRAuxfuE	I	BOOL		Evaluate DL function in OB 40
UserVersion	I	POINTER		Pointer to string variable. The associated string variable is indicated in the version display (max. 41 characters).
MaxBAG	Q	INT	1..10	Number of mode groups
MaxChan	Q	INT	1..10	Number of channels
MaxAxis	Q	INT	1..31	Number of axes
ActivChan	Q	ARRAY[1..10] OF BOOL		Bit string for active channels
ActivAxis	Q	ARRAY[1..31] OF BOOL		Bit string for active axes
UDInt	A	INT		Number of integer machine data in DB 20

Signal	I/O	Type	Value range	Remark
UDHex	A	INT		Number of hexadecimal machine data in DB 20
UDReal	A	INT		Number of real machine data in DB 20

Note

Explanations of formal parameters of this function for FM-NC:
For signals:

MCP1 StatRec (machine control panel)
MCP2 StatRec (machine control panel)
BHGStatRec (handheld unit)

the start addresses apply for the status doubleword for reception of DW#16#00000400;

MCP1 Timeout (machine control panel)
MCP2 Timeout (machine control panel)

700 ms recommended cyclic life-time monitoring of the machine control panel.

MCP/HHU monitoring (for 810D, 840D)

The following status information regarding communication with the machine control panels is output in the event of an error:

Available in:	Bit No.	Description
MCP1StatRec MCP2StatRec BHGStatRec	10	Receiver: Time-out
SINUMERIK 840D only: MCP1StatSend MCP2StatSend BHGStatSend	27	Transmitter: Time-out

In addition, an error entry is generated in the diagnostics buffer of the PLC,

resulting in the following error messages on the operator interface:

- 400260: MCP 1 failure or

- 400261: MCP 2 failure

- 400262: HHU failure

In this case, the input signals from the MCP or from the handheld unit (MCP1In/ MCP2In or BHGIn) are initialized with 0. If it is possible to resynchronize the PLC and MCP/HHU, communication is resumed automatically and the error message reset by the BP.

Call example for SINUMERIK 810D

An example call for the FB 1 in OB 100 appears below. This example is part of the diskette with the basic program for 810D.

```
ORGANIZATION_BLOCK OB 100
VAR_TEMP
    OB100_EV_CLASS :                BYTE ;
    OB100_STRTUP :                  BYTE ;
    OB100_PRIORITY :                BYTE ;
    OB100_OB_NUMBR :                BYTE ;
    OB100_RESERVED_1 :              BYTE ;
    OB100_RESERVED_2 :              BYTE ;
    OB100_STOP :                    WORD ;
    OB100_RESERVED_3 :              WORD ;
    OB100_RESERVED_4 :              WORD ;
    OB100_DATE_TIME :               DATE_AND_TIME;
END_VAR
BEGIN
    CALL FB 1, DB 7 (
                    MCPNum :=        1,
                    MCP1In :=        P#E0.0,
                    MCP1Out :=       P#A0.0,
```

```
                                    MCP1StatSend :=            P#A8.0,
                                    MCP1StatRec :=             P#A12.0,
                                    MCP1BusAdr :=              14,
                                    MCP1Timeout :=            S5T#700MS,
                                    MCPMPI:=                   TRUE,
                                    NCCyclTimeout :=          S5T#200MS,
                                    NCRunupTimeout :=         S5T#50S);
//INSERT USER PROGRAM HERE
END_ORGANIZATION_BLOCK
```

Call example for SINUMERIK 840D

An example call for the FB 1 in OB 100 appears below. This example is part of the diskette with basic program for 840D.

```
ORGANIZATION_BLOCK OB 100
VAR_TEMP
      OB100_EV_CLASS :                 BYTE ;
      OB100_STRTUP :                   BYTE ;
      OB100_PRIORITY :                 BYTE ;
      OB100_OB_NUMBR :                 BYTE ;
      OB100_RESERVED_1 :               BYTE ;
      OB100_RESERVED_2 :               BYTE ;
      OB100_STOP :                     WORD ;
      OB100_RESERVED_3 :               WORD ;
      OB100_RESERVED_4 :               WORD ;
      OB100_DATE_TIME :                DATE_AND_TIME;
END_VAR
BEGIN
      CALL FB 1, DB 7 (
                      MCPNum :=            1,
                      MCP1In :=            P#E0.0,
                      MCP1Out :=           P#A0.0,
                      MCP1StatSend :=      P#A8.0,
                      MCP1StatRec :=       P#A12.0,
                      MCP1BusAdr :=        6,
                      MCP1Timeout :=       S5T#700MS,
                      MCP1Cycl :=          S5T#200MS,
                      NC-CyclTimeout :=    S5T#200MS,
                      NC-RunupTimeout :=   S5T#50S);
//INSERT USER PROGRAM HERE
END_ORGANIZATION_BLOCK
```

2.12.2 FB 2: Read GET NC variable

Description of Functions

The PLC user program can read variables from the NCK area using FB GET. This function block is capable of multi-instances and an instance-DB from the user area belongs to the FB 2.

When FB 2 is called with a positive signal edge change at control input "Req", a job is started, which reads the NCK variables referenced by ADDR1ADDR8 and then copies them to the PLC operand areas referenced by RD1 to RD8. Successful completion of the read process is indicated by a logical "1" in status parameter NDR.

The **read process** lasts for several PLC cycles (normally 1-2). The block can be called up in cyclic mode only.

Any errors are indicated by Error and State.

In order to reference the NC variables, all required variables are first selected with the "NC VAR selector" tool and generated as STL source in a data block. A name must then be assigned to this DB in the symbol table.
"DB name.S7 name" is transferred as the actual parameter of the NCK variable address (Addr1 to Addr8) when FB 2 is called.

Variable addressing

For some NC-variables, it is necessary to select area no. and/or line or column from the NC-VAR selector. A basic type can be selected for these variables,
i.e., area/column/line are preset to "0".
The contents of the area number, line and column specified by the NC VAR selector are checked for a "0" in the FB. If a "0" is present, the value is transferred to the input parameter. The user must supply the required parameters (UnitX/ColumnX/LineX) before calling FB GET.
Here, unit corresponds to the area No., column to the column and line to the row.

Notice

FB 2 can read NC variables only
if basic program parameter NCKomm ="1" has been set (in OB 100: FB 1, DB 7). The call is permitted only in cyclic program OB 1.

When **channel-specific** variables are read, in a task (FB 2-call)
via Addr1 to Addr8 only the variables of exactly **one** channel may be addressed.

In areas V and H, different logic axis numbers must not be assigned in one job. (Failure to observe this rule results in Error:= TRUE, State:= W#16#02).

NCK variables within **one** group can be combined in a job:

	range				
Group 1	C[1]	N	B	A	T
Group 2	C[2]	N	B	A	T
Group 3	V[.]	H[.]			

The same rules apply to channels 3 to 10 as illustrated as examples in the above table in groups 1 and 2.

Note

Especially when reading several long strings, the number of usable variables can be less than 8.

Declaration of the function

```
FUNCTION_BLOCK FB 2
VAR_INPUT
    Req :                 BOOL ;
    NumVar :              INT ;
    Addr1 :               ANY ;
    Unit1 :               BYTE ;
    Column1 :             WORD ;
    Line1 :               WORD ;
    Addr2 :               ANY ;
    Unit2 :               BYTE ;
    Column2 :             WORD ;
    Line2 :               WORD ;
    Addr3 :               ANY ;
    Unit3 :               BYTE ;
    Column3 :             WORD ;
    Line3 :               WORD ;
    Addr4 :               ANY ;
    Unit4 :               BYTE ;
    Column4 :             WORD ;
    Line4 :               WORD ;
    Addr5 :               ANY ;
    Unit5 :               BYTE ;
    Column5 :             WORD ;
    Line5 :               WORD ;
    Addr6 :               ANY ;
    Unit6 :               BYTE ;
```

```
    Column6 :              WORD ;
    Line6 :                WORD ;
    Addr7 :                ANY ;
    Unit7 :                BYTE ;
    Column7 :              WORD ;
    Line7 :                WORD ;
    Addr8 :                ANY ;
    Unit8 :                BYTE ;
    Column8 :              WORD ;
    Line8 :                WORD ;
    FMNCNo :               INT;1)
END_VAR
VAR_OUTPUT
Error :                   BOOL ;
NDR :                     BOOL ;
State :                   WORD ;
END_VAR


VAR_IN_OUT
    RD1 :                  ANY ;
    RD2 :                  ANY ;
    RD3 :                  ANY ;
    RD4 :                  ANY ;
    RD5 :                  ANY ;
    RD6 :                  ANY ;
    RD7 :                  ANY ;
    RD8 :                  ANY ;
END_VAR
```

Description of formal parameters

The table below list all formal parameters of the GET function.

Signal	Type	Type	Value range	Remark
Req	I	BOOL		Job start with positive signal edge
NumVar	I	INT	1 to 8 (corresponds to use of Addr1 to Addr8)	Number of variables to be read
Addr1 to Addr8	I	ANY	[DBName].[VarName]	Variable identifiers from **NC Var selector**
Unit1 to Unit8	I	BYTE		Area address, optional for variable addressing
Column1 to Column8	I	WORD		Column address, optional for variable addressing
Line1 to Line8	I	WORD		Line address, optional for variable addressing
Error	A	BOOL		Negative acknowledgment of job or execution of job impossible
NDR	A	BOOL		Job successfully executed Data are available
State	A	WORD		See error identifiers
RD1 to RD8	I/O	ANY	P#Mm.n BYTE x... P#DBnr.dbxm.n BYTE x	Target area for read data

Error identifiers

If it was not possible to execute a job, the failure is indicated by "logic 1" on status parameter error. The error cause is coded at the block output State:

State		Description	Note
WORD H	WORD L		
1 to 8	1	Access error	In high byte number of Var in which error occurred
0	2	Error in job	Incorrect compilation of Var. in a job
0	3	Negative acknowledgment, job not executable	Internal error, any help: NC RESET
1 to 8	4	Insufficient local user memory available	Read var. is longer than specified in RD1 to RD8; in high byte number of var in which error occurred
0	5	Format conversion error	Error on conversion of var. type double: Var. is not within S7 REAL area
0	6	FIFO full	Job must be repeated since queue is full
0	7	Option not set	BP parameter "NCKomm" is not set
1 to 8	8	Incorrect target area (RD)	RD1 to RD8 may not be local data
0	9	Transmission occupied	Job must be repeated
1 to 8	10	Error in variable addressing	Unit or column/line contains value 0
0	11	Address of variable invalid	Check Addr (or variable name), area, unit
0	12	NumVar = 0	Check parameter NumVar

Configuration steps

Proceed as follows to read NC variables:

• Select variables with the NC VAR selector.

• Store the selected variables in a *.VAR file in the required project catalog (*.S7D).

• Generate a STEP 7 *.STL source file.

• Generate a DB with the associated address data.

• Enter the symbol for the generated DB in the symbol table so that it is possible to access the address parameters symbolically in the user program.

• Set FB2 parameters

Pulse diagram
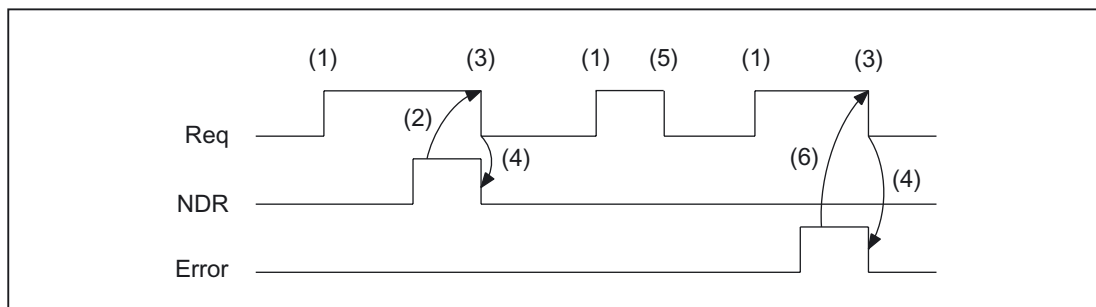
(1)   Activation of function

(2)   Positive acknowledgment: Receive new data

(3)   Reset function activation after receipt of acknowledgment

(4)   Signal change by means of FB

(5)   If function activation is reset prior to receipt of acknowledgment, the output signals are not updated without the operational sequence of the activated function being affected.

(6)   Negative acknowledgment: Error has occurred, error code in output parameter state

Call example

Reading of three channel-specific machine data from channel 1, whose address specifications are stored in DB120.

Select data with NC VAR selector and store in file DB120.VAR; then create file DB120.AWL:

range	Block	Name	Type	No.	Byte	S7 Name
C[1]	M (GND)	MD20070: AXCONF_MACHAX_USED[1]	char	20070	1	C1AxConfMachAxUsed1
C[1]	M (GND)	MD20070: AXCONF_MACHAX_USED[2]	char	20070	1	C1AxConfMachAxUsed2
C[1]	M (GND)	MD20090: SPIND_DEF_MASTER_SPIND	int	20090	1	C1SpindDefMasterSpind

S7 (ALIAS) names have been selected in order to:

- Incorporate the channel designation into the name

    and

- Remove the characters [ ], which are not legal in a STEP 7 symbol.

Entry of the name in the S7 SYMBOL table (e.g., NCVAR for DB 120):

Symbol	Operand	Data type
NCVAR	DB 120	DB 120

File DB120.AWL must be compiled and transferred to the PLC.

Parameterization of FB 2 with instance DB 110:

```
DATA_BLOCK DB 110              //Unassigned user DB, as instance for FB 2
FB 2
BEGIN
END_DATA_BLOCK
Function FC "VariablenCall": VOID
    U        I 7.7;       //Unassigned machine control panel key
    S        M 100.0;     //Activate req.
    U        M 100.1;     //NDR completed message
    R        M 100.0;     //Terminate job
    U        I 7.6;       //Manual error acknowledgment
    U        M 102.0;     //Error pending
    R        M 100.0;     //Terminate job
    CALL FB 2, DB 110 (
            Req :=       M 100.0,
            NumVar :=    3,                    //Read 3 variables
            Addr1 :=     NCVAR.C1AxConfMachAxUsed1,
            Addr2 :=     NCVAR.C1AxConfMachAxUsed2,
            Addr3 :=     NCVAR.C1SpindDefMasterSpind,
            Error :=     M102.0,
            NDR :=       M100.1,
            State :=     MW104,
            RD1 :=       P#DB99.DBX0.0 BYTE 1,
            RD2 :=       P#DB99.DBX1.0 BYTE 1,
            RD3 :=       P#M110.0 INT 1);
```

Example Variable addressing

Reading of two R parameters of channel 1, whose address specifications are stored in DB 120 as the basic type. The R parameter number is parameterized via parameter LineX.

```
DATA_BLOCK DB 120
 VERSION : 0.0
STRUCT
    C1_RP_rpa0_0:
    STRUCT
    SYNTAX_ID :          BYTE := B#16#82;
    area_and_unit :      BYTE := B#16#41;
    column :             WORD := W#16#1;
    line :               WORD := W#16#0;
    block type :         BYTE := B#16#15;
    NO. OF LINES :       BYTE := B#16#1;
    type :               BYTE := B#16#F;
    length :             BYTE := B#16#8;
```

```
    END_STRUCT;

END_STRUCT;

BEGIN

END_DATA_BLOCK
    CALL FB 2, DB 110 (
                Req :=          M 0.0,
                NumVar :=       2,
                Addr1 :=        "NCVAR".C1_RP_rpa0_0,
                Line1 :=        W#16#1,
                Addr2 :=        "NCVAR".C1_RP_rpa0_0,
                Line2 :=        W#16#2,
                FMNCNo :=       1,
                Error :=        M 1.0,
                NDR :=          M 1.1,
                State :=        MW 2,
                RD1 :=          P#M 4.0 REAL 1,
                RD2 :=          P#M 24.0 REAL 1);
```

Data types

The data types of the NCK are listed in the NC-VAR selector with the variables. The tables below give the assignments to the S7 data types.

Classification of data types	
NCK data type	S7 data type
double	REAL
float	REAL
long	DINT
integer	DINT
uint_32	DWORD
int_16	INT
uint_16	WORD
unsigned	WORD
char	CHAR or BYTE
string	STRING
bool	BOOL

2.12.3   FB 3: PUT write NC variables

Description of functions

The PLC user program can write variables in the NCK area using FB PUT.

Every FB 3 call must be assigned a separate instance DB from the user area. (multiinstance capability in SW 3.7 and higher).

When FB 3 is called with a positive signal edge change at control input Req, a job is started to overwrite the NC variables referenced by Addr1 to Addr8 with the data of the PLC operand areas locally referenced by SD1 to SD8. Successful completion of the write process is indicated by a logical "1" in status parameter "Done".

The **write process** lasts for several PLC cycles (normally 1-2). The block can be called up in cyclic mode only.

Any errors are indicated by Error and State.
In order to reference the NC variables, all required variables are first selected with the "NC VAR selector" tool and generated as STL source in a data block. A name must then be assigned to this DB in the symbol table. "DB name.S7 name" is transferred as the actual parameter of the NCK variable address (Addr1 to Addr8) when FB3 is called.

Variable addressing

For some NC variables, it is necessary to select area no. and/or line or column in the NC VAR selector. A basic type can be selected for these variables,
i.e., area/column/line are preset to "0".
The contents of the area number, line and column specified by the NC VAR selector are checked for a "0" in the FB. If a "0" is present, the value is transferred to the input parameter. The user must supply the required parameters (UnitX/ColumnX/LineX) before calling FB PUT.

Machine data, GUD

In order to define machine data and GUDs without a password, the protection levels of the data you want to access must be redefined to the lowest level.

The process is described in the /IADC/ Commissioning instructions, "Protection level concept" or in the Programming Manual Job Planning, "Define protection levels for user data (GUD)".

Notice

FB 3 can read NC variables only
if basic program parameter NCKomm ="1" has been set (in OB 100: FB 1, DB 7). The call is permitted only in cyclic program OB 1.

When **channel-specific** variables are read, in a task (FB 3-call)
via Addr1 to Addr8 only the variables of exactly **one** channel may be addressed.

In areas V and H, different logic axis numbers must not be assigned in one job. (Failure to observe this rule results in Error:= TRUE, State:= W#16#02).

NCK variables within **one** group can be combined in a job:

	Area				
Group 1	C[1]	N	B	Q	T
Group 2	C[2]	N	B	Q	T
Group 3	V[.]	H[.]			

The same rules apply to channels 3 to 10 as illustrated as examples in the above table in groups 1 and 2.

Note

Especially when reading several long strings, the number of usable variables can be less than 8.

Declaration of the function

```
FUNCTION_BLOCK FB 3
VAR_INPUT
    Req :                   BOOL ;
    NumVar :                INT ;
    Addr1 :                 ANY ;
    Unit1 :                 BYTE ;
    Column1 :               WORD ;
    Line1 :                 WORD ;
    Addr2 :                 ANY ;
    Unit2 :                 BYTE ;
    Column2 :               WORD ;
    Line2 :                 WORD ;
    Addr3 :                 ANY ;
    Unit3 :                 BYTE ;
    Column3 :               WORD ;
    Line3 :                 WORD ;
    Addr4 :                 ANY ;
    Unit4 :                 BYTE ;
    Column4 :               WORD ;
    Line4 :                 WORD ;
    Addr5 :                 ANY ;
    Unit5 :                 BYTE ;
    Column5 :               WORD ;
    Line5 :                 WORD ;
    Addr6 :                 ANY ;
    Unit6 :                 BYTE ;
    Column6 :               WORD ;
    Line6 :                 WORD ;
    Addr7 :                 ANY ;
    Unit7 :                 BYTE ;
    Column7 :               WORD ;
    Line7 :                 WORD ;
    Addr8 :                 ANY ;
    Unit8 :                 BYTE ;
    Column8 :               WORD ;
    Line8 :                 WORD ;
END_VAR
VAR_OUTPUT
    Error :                 BOOL ;
    Done :                  BOOL ;
    State :                 WORD ;
END_VAR
VAR_IN_OUT
```

```
        SD1 :              ANY ;
        SD2 :              ANY ;
        SD3 :              ANY ;
        SD4 :              ANY ;
        SD5 :              ANY ;
        SD6 :              ANY ;
        SD7 :              ANY ;
        SD8 :              ANY ;

END_VAR
```

Description of formal parameters

The table below lists all formal parameters of the PUT function.

Signal	I/O	Type	Value range	Remark
Req	I	BOOL		Job start with positive signal edge
NumVar	I	INT	1 to 8 (corresponds to use of Addr1 to Addr8)	Number of variables to be written
Addr1 to Addr8	I	ANY	[DBName].[VarName]	Variable identifiers from **NC Var selector**
Unit 1 to Unit 8	I	BYTE		Area address, optional for variable addressing
Column 1 to Column 8	I	WORD		Column address, optional for variable addressing
Line 1 to Line 8	I	WORD		Line address, optional for variable addressing
Error	Q	BOOL		Negative acknowledgment of job or execution of job impossible
Done	Q	BOOL		Job successfully executed
State	Q	WORD		See error identifiers
SD1 to SD8	I/O	ANY	P#Mm.n BYTE x... P#DBnr.dbxm.n BYTE x	Data to be written
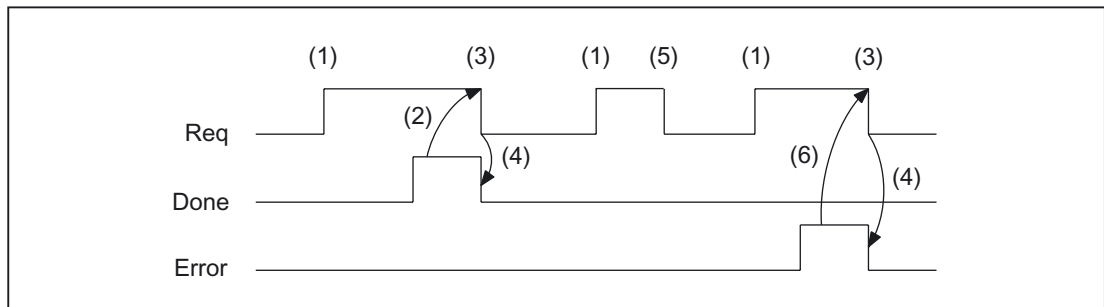
Error identifiers

If it was not possible to execute a job, the failure is indicated by "logic 1" on status parameter error. The error cause is coded at the block output State:

State		Meaning	Note
WORD H	WORD L		
1 to 8	1	Access error	In high byte number of Var in which error occurred
0	2	Error in job	Incorrect compilation of Var in a job
0	3	Negative acknowledgment, job not executable	Internal error, any help: NC RESET
1 to 8	4	Data areas or data types do not match or string is empty	Check data to be written in SD1 to SD8; in high byte number of the Var in which error occurred
0	6	FIFO full	Job must be repeated since queue is full
0	7	Option not set	BP parameter "NCKomm" is not set
1 to 8	8	Incorrect target area (SD)	SD1 to SD8 may not be local data
0	9	Transmission occupied	Job must be repeated
1 to 8	10	Error in variable addressing	Unit or column/line contains value 0
0	11	Variable addr. invalid or var. is read-only	Check Addr (or variable name), area, unit
0	12	NumVar = 0	Check parameter NumVar

Configuration steps

To write NC variables, the same configuration steps are required as for reading NC variables. It is useful to store the address data of all NC variables to be read or written in a DB.

Pulse diagram

(1)	Activation of function
(2)	Positive acknowledgment: variables have been written
(3)	Reset function activation after receipt of acknowledgment
(4)	Signal change by means of FB
(5)	If function activation is reset prior to receipt of acknowledgment, the output signals are not updated without the operational sequence of the activated function being affected.
(6)	Negative acknowledgment: Error has occurred, error code in output parameter state.

Call example

Writing of three channelspecific machine data of channel 1:

Select the three data with NC VAR selector and store in the file DB120.VAR:

Area	Block	Name	Type	Byte	S7 Name
C[1]	RP	rpa[5]	double	4	rpa_5C1RP
C[1]	RP	rpa[11]	double	4	rpa_11C1RP
C[1]	RP	rpa(14)	double	4	rpa_14C1RP

Entry NCVAR for DB 120 with the S7 SYMBOL Editor:

Symbol	Operand	Data type
NCVAR	DB120	DB 120

File DB120.AWL must be compiled and transferred to the PLC.

Call and parameterization of FB 3 with instance DB 111:

```
DATA_BLOCK DB 111          //Unassigned user DB, as instance for FB 3
FB 3
BEGIN
Function FC "VariablenCall": VOID
END_DATA_BLOCK
     U    I 7.7;        //Unassigned machine control panel key
     S    M 100.0;      //Activate req.
     U    M 100.1;      //Done completed message
     R    M 100.0;      //Terminate job
     U    I 7.6;        //Manual error acknowledgment
     U    M 102.0;      //Error pending
     R    M 100.0;      //Terminate job
     CALL FB 3, DB 111 (
          Req :=        M 100.0,
          NumVar :=      3,                      //Write 3 variables
          Addr1 :=      NCVAR.rpa_5C1RP,
          Addr2 :=      NCVAR.rpa_11C1RP,
          Addr3 :=      NCVAR.rpa_14C1RP,
          FMNCno :=     1,
          Error :=      M102.0,
          Done :=       M100.1,
          State :=      MW104,
          SD1 :=        P#DB99.DBX0.0 REAL 1,
          SD2 :=        P#DB99.DBX4.0 REAL 1,
          SD3 :=        P#M110.0 REAL 1);
```

Example: Variable addressing

Writing of two R parameters of channel 1, whose address specifications are stored in DB 120 as the basic type. The R parameter number is parameterized via parameter LineX.

```
DATA_BLOCK DB 120
 VERSION : 0.0
STRUCT
    C1_RP_rpa0_0:
    STRUCT
    SYNTAX_ID :           BYTE := B#16#82;
    area_and_unit :       BYTE := B#16#41;
    column :              WORD := W#16#1;
    line :                WORD := W#16#0;
    block type :          BYTE := B#16#15;
    NO. OF LINES :        BYTE := B#16#1;
    type :                BYTE := B#16#F;
    length :              BYTE := B#16#8;
    END_STRUCT;
END_STRUCT;
BEGIN
END_DATA_BLOCK
CALL FB 3, DB 122 (
        Req :=        M 10.0,
        NumVar :=     2,
        Addr1 :=      "NCVAR".C1_RP_rpa0_0,
        Line1 :=      W#16#1,
        Addr2 :=      "NCVAR".C1_RP_rpa0_0,
        Line3 :=      W#16#2
        Error :=      M 11.0,
        Done :=       M 11.1,
        State :=      MW 12,
        SD1 :=        P#M 4.0 REAL 1,
        SD2 :=        P#M 24.0 REAL 1);
```

2.12.4   FB 4: PI_SERV General PI services

Description of Functions

FB PI_SERV can be used to start program-instance services in the NCK area.

A program section, which carries out a particular function (e.g., with tool management, search for empty location in a magazine), is executed in the NCK by making a request via the PI service.

Every FB 4 call must be assigned a separate instance DB from the user area. The documentation for using the multii-instance capability is contained in the STEP7 descriptions.
The specified service is referenced via the "PIService" parameter. The selected PI service is supplied via the freely assignable additional input variables with varying data types (Addr1 to Addr4 for strings, WVar1 to WVar 10 for integer or word variables).

A job is started when **FB 4 is called** by means of a positive edge change at control input Req. Successful execution of the job is displayed by means of a logical "1" in status parameter "Done". Any errors are indicated by Error and State.
The "PI" data block (DB16) contains internal descriptions of the possible PI services. A name must then be assigned to this DB in the symbol list. On calling the FB 4, "DB-Name.PI-Name" is transferred as the actual parameter for "PIService".

The execution of the PI service extends over several PLC cycles (generally 1 to 2). The block can be called up in cyclic mode only.

Note

The FB 4 can start PI services only
if the basic program parameter NCKomm has been set to "1" (in OB100: FB 1, DB 7). The call is permitted only in cyclic program OB 1.

Declaration of the function

```
FUNCTION_BLOCK FB 4
VAR_INPUT
    Req :               BOOL ;
    PIService :         ANY ;
    Unit :              INT ;
    Addr1 :             ANY ;
    Addr2 :             ANY ;
    Addr3 :             ANY ;
    Addr4 :             ANY ;
    WVar1 :             WORD ;
    WVar2 :             WORD ;
    WVar3 :             WORD ;
    WVar4 :             WORD ;
```

```
    WVar5 :              WORD ;
    WVar6 :              WORD ;
    WVar7 :              WORD ;
    WVar8 :              WORD ;
    WVar9 :              WORD ;
    WVar10 :             WORD ;
    FMNCNo :             INT ;              //(in FMNC only)
END_VAR
VAR_OUTPUT
    Error :              BOOL ;
    Done :               BOOL ;
    State :              WORD ;
END_VAR
```

Description of formal parameters

The following table shows all formal parameters of the function PI_SERV.

Signal	Type	Type	Value range	Remark
Req	I	BOOL		Job request
PIService	I	ANY	[DBName].[VarName] Default is: "PI".[VarName]	PI service description[1]
Unit	I	INT	1...	Area number
Addr1 to Addr4	I	ANY	[DBName].[VarName]	Reference to strings specification according to selected PI service
WVar1 to WVar10	I	WORD	1...	Integer or word variables. Specification according to selected PI service
Error	Q	BOOL		Negative acknowledgment of job or execution of job impossible
Done	Q	BOOL		Job successfully executed
State	Q	WORD		See error identifiers
[1]See README file on basic program diskette supplied				

Error identifiers

If it was not possible to execute a job, the failure is indicated by "logic 1" on status parameter error. The error cause is coded at the block output State:

State	Meaning	Note
3	Negative acknowledgment, job not executable	Internal error, any help: NC RESET
6	FIFO full	Job must be repeated since queue is full
7	Option not set	BP parameter "NCKomm" is not set
9	Transmission occupied	Job must be repeated

Pulse diagram
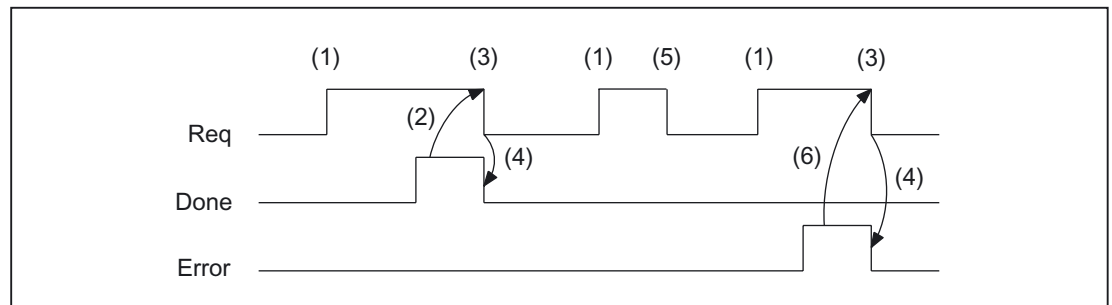
(1)	Activation of function
(2)	Positive acknowledgment: PI service has been executed
(3)	Reset function activation after receipt of acknowledgment
(4)	Signal change by means of FB
(5)	If function activation is reset prior to receipt of acknowledgment, the output signals are not updated without the operational sequence of the activated function being affected.
(6)	Negative acknowledgment: Error has occurred, error code in the output parameter State

2.12.4.1 Overview of available PI services

Overview of PI services

The following section provides an overview of the PI services that can be started from the PLC. The meaning and application of the general FB 4 input variables (Unit, Addr ..., WVar ...) depend on the individual PI service concerned.

PI service	Function	available SINUMERIK	
		FM-NC	810/840D
ASUB	Assign interrupt	*	*
CANCEL	Execute cancel	*	*
CONFIG	Reconfiguration of tagged machine data	*	*
DIGION	Digitizing on		*
DIGIOF	Digitizing off		*
FINDBL	Activate block search	*	*
LOGIN	Activate password	*	*
LOGOUT	Reset password	*	*
NCRES	Trigger NC RESET.		*
SELECT	Select program for processing for one channel	*	*
SETUDT	Sets the current user data to active		*
PI service	Tool management function		
CRCEDN	Create new cutting edge		*
CREACE	Create cutting edge	*	*
CREATO	Generate tool	*	*
SETUFR	Activate user frames	*	*
DELECE	Delete a cutting edge		*
DELETO	Delete tool	*	*
MMCSEM	Semaphores for various PI services		*
TMCRTO	Create tool		*
TMFDPL	Empty location search for loading		*
TMFPBP	Empty location search		*
TMGETT	T-number for the previous tool identifier with Duplo number		*
TMMVTL	Prepare magazine location for loading, unload tool		*
TMPOSM	Position magazine location or tool		*
TMPCIT	Set increment value for workpiece counter		*
TMRASS	Reset active status		*
TRESMO	Reset monitoring values		*
TSEARC	Complex search using search screen forms		*
x: PI service is available			

2.12.4.2   General PI services

The possible services are described in this section.

PI service: ASUB

Allocate function interrupt:

A program stored on the NCK is assigned an interrupt signal for a channel. This is possible only if the file may be executed. The path name and program name must be entered as described in the Programming Manual Job Planning, File and Program Management, Section "Program Memory"). Please also refer to example of FB 4 for notation of path and program names.

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.ASUP	Assign interrupt
Unit	INT	1 to 10	Channel
WVar1	WORD	1 to 8	Interrupt number
WVar2	WORD	1 to 8	Priority
WVar3	WORD	0/1	LIFTFAST
WVar4	WORD	0/1	BLSYNC
Addr1	STRING		Path name
Addr2	STRING		Program name

Note

The SETINT instruction is also used to make the assignment.

The ASUP PI service may only be executed when the channel to be activated is in RESET state.

References:
/PGA/ Programming Manual Job Planning, Flexible NC-Programming Chapter "Interrupt routine (SETINT, DISABLE, ENABLE, CLRINT)".

PI service: CANCEL

Execute the function Cancel:

The CANCEL command activates the Cancel function (in the same way as the key on the HMI).

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.CANCEL	Cancel

PI service: CONFIG

Function Reconfiguration:

The reconfiguration command activates machine data, which have been entered sequentially by the operator or the PLC, almost in parallel.
The command can only be activated when the control is in RESET state or the program is interrupted (NC stop at block limit). An FB 4 error checkback message is output if these conditions are not fulfilled (state = 3).

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.CONFIG	Reconfiguration
Unit	INT	1	
WVar1	INT	1	Classification

PI service: DIGION

Function Digitalize On:

Selecting digitizing in the specified channel.

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.DIGION	Digitizing on
Unit	INT	1 to 10	Channel

PI service: DIGIOF

Function Digitalize Off:

Deactivating digitizing in the specified channel.

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.DIGIOF	Digitizing off
Unit	INT	1 to 10	Channel

PI service: FINDBL

Activate function Search:

A channel is switched to block search mode and the appropriate acknowledgment then transmitted. The block search is then executed immediately by the NCK. The search pointer must already be in the NCK at this point in time. The search can be interrupted at any time by an NC RESET. Once the search is successfully completed, the normal processing mode is reactivated automatically. NC Start then takes effect from the located search target. The operator is responsible for providing a collisionfree approach path.

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.FINDBL	Block search
Unit	INT	1 to 10	Channel
WVar1	WORD	x	Preprocessing mode
x: Describes the preprocessing mode x = 1 without calculation x = 2 with calculation x = 3 with main block observation			

PI service: LOGIN

Create function Keyword:

Transfers the parameterized password to the NCK. The passwords generally consist of 8 characters. If required, blanks must be added to the string of the password.

Example:

Password: STRING[8] := 'SUNRISE';

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.LOGIN	Create password
Unit	INT	1	NCK
Addr1	STRING	8 characters	Password

PI service: LOGOUT

Reset function Keyword:

The password last transferred to the NCK is reset.

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.LOGOUT	Reset password
Unit	INT	1	NCK

PI service: NCRES

Trigger function NC-RESET:

Initiates an NCK RESET. The Unit and WVar1 parameters must be assigned 0.

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.NCRES	Trigger NC-RESET
Unit	INT	0	0
WVar1	WORD	0	0

PI service: SELECT

Select function Processing for a channel:

A program stored on the NCK is selected for processing for one channel. This is possible only if the file may be executed. The path name and program name must be entered as described in the Programming Manual Job Planning, File and Program Management, Section "Program Memory". Please also refer to example of FB 4 for notation of path and program names.

Possible block types

Block types	
Workpiece directory	WPD
Main program	MPF
Subroutine	SPF
cycles	CYC
Asynchronous subprograms	ASP
Binary files	BIN

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.SELECT	Program selection
Unit	INT	1 to 10	Channel
Addr1	STRING		Path name
Addr2	STRING		Program name

PI service: SETUDT

Set function current user data active

The current user data, such as tool offsets, basic frames and settable frames are set to active in the next NC block (only in STOP state).

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.SETUDT	Activate user data
Unit	INT	1 to 10	Channel
WVar1	WORD	1 to 5	User data type 1 = active tool offset 2 = active basic frame 3 = active settable frame 4 = active global basic frame 5 = active global settable frame
WVar2	WORD	0	Standby
WVar3	WORD	0	Standby

PI service: SETUFR

Activate function user frames:

User frames are loaded to the NCK. All necessary frame values must be transferred to the NCK beforehand by writing variables with FB 3.

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.SETUFR	Activate user frames
Unit	INT	1 to 10	Channel

2.12.4.3 Tool management services

Overview of PI services

Available PI services of the function tool management.

PI service	Function	FM-NC	840D
CRCEDN	Create new cutting edge		*
CREACE	Create cutting edge	*	*
CREATO	Generate tool	*	*
SETUFR	Activate user frames	*	*
DELECE	Delete a cutting edge		*
DELETO	Delete tool	*	*
MMCSEM	Semaphores for various PI services		*
TMCRTO	Create tool		*
TMFDPL	Empty location search for loading		*
TMFPBP	Empty location search		*
TMMVTL	Prepare magazine location for loading, unload tool		*
TMPOSM	Position magazine location or tool		*
TMPCIT	Set increment value for workpiece counter		*
TMRASS	Reset active status		*
TRESMO	Reset monitoring values		*
TSEARC	Complex search using search screen forms		*
x: PI service is available			

PI service: CRCEDN

Function Create new cutting edge

Create tool edge by specifying the cutting edge number.

If the T number of an existing tool is specified in parameter "T number" in the PI service, then a cutting edge is set up for this particular tool (in this case, parameter "D number" (number of cutting edge to be created) has a value range of 00001–00009). If a positive T number is specified as a parameter and the tool for the T number entered does not exist, then the PI service is aborted. If a value of 00000 is entered for the T number (model of absolute D numbers), then the D number value range might extend from 00001 to 31999. The new cutting edge is set up with the specified D number. If the specified cutting edge already exists, then the PI service is aborted in both cases.

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.CRCEDN	Create new cutting edge
Unit	INT	1, 2	TOA
WVar1	INT		T number of tool for which cutting edge must be created. A setting of 00000 states that the cutting edge should not refer to any particular tool (absolute D number).
WVar2	INT	1 - 9 or 01 - 31999	Edge number of tool cutting edge

PI service: CREACE

Create function tool cutting edge:

Creation of the cutting edge with the next higher/next unassigned D number for the tool with the transferred T number in TO, TS (if present). The cutting edge for the OEM cutting edge data is set up simultaneously in the TUE block - if present.

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.CREACE	Create cutting edge
Unit	INT	1, 2	TOA
WVar1	INT		T number

PI service: CREATO

Create function tool:

Creation of a tool with specification of a T number. The tool is entered as existing in the tool directory area (TV). The first "cutting edge" D1 (with zero contents) is created for tool offsets in the TO block. D1 (with zero contents) is also created for the OEM "cutting edge" data in the TUE block - if one is present. If a TU block exists, it will contain the data set for the tool.

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.CREATO	Create tool
Unit	INT	1, 2	TOA
WVar1	INT		T number

PI service: DELECE

Function Delete a cutting edge:

If the T number of an existing tool is specified in parameter "T number" in the PI service, then a cutting edge is deleted for this particular tool (in this case, parameter "D number" (number of cutting edge to be created) has a value range of 00001–00009). If a positive T number is specified as a parameter and the tool for the T number entered does not exist, then the PI service is aborted. If a value of 00000 is entered for the T number (model of absolute D numbers), then the D number value range might extend from 00001 to 31999. If the specified cutting edge does not exist, then the PI service is aborted in both cases.

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.DELETE	Delete cutting edge
Unit	INT	1, 2	TOA
WVar1	INT		T number of tool for which cutting edge must be created. A setting of 00000 states that the cutting edge should not refer to any particular tool (absolute D number).
WVar2	INT	1 - 9 or 01 - 31999	Edge number of cutting edge that must be deleted

PI service: DELETO

Delete function tool:

Deletes the tool assigned to the transferred T number with all cutting edges (in TO, in some cases TU, TUE and TG (type 400), TD and TS blocks).

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.DELETO	Delete tool
Unit	INT	1, 2	TOA
WVar1	INT		T number

PI service: MMCSEM

Semaphores for various PI services

For use by MMC and PLC

10 semaphores are provided for each channel. These protect critical functions for the MMC/PLC. By setting the semaphore for the corresponding function number, several MMC/PLC units can be synchronized with it in cases where a function contains a critical section with respect to data to be fetched by the NCK. Semaphores are managed by the MMC/PLC. A semaphore value of 1stipulates a Test & Set operation for the semaphore of the specified function number. The return value of the PI service represents the result of this operation:

- Checkback value Done := TRUE:

  Semaphore has been set, critical function can be called.

- Checkback value Error := TRUE with state = 3: Semaphore was already set, critical function cannot be called. The operation must be repeated later.

Note

On completion of the operation (reading data of this PI service) it is **essential** that the **semaphore** is **enabled again**.

Parameter:

WVar1 = FunctionNumber

This function number represents a PI service:

`1:`	TMCRTO (create tool)
`2:`	TMFDPL (search for empty location for loading):
`3:`	TMMVTL (prepare magazine location for loading, unload tool)
`4:`	TMFPBP (search for location)
`5:`	TMGETT (search for tool number)
`6:`	TSEARC (search for tool)
`7 ...` `10:`	Reserved

`WVar2=SemaphorValue`

`0:`	Reset semaphore
`1:`	Test and set semaphore

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.MMCSEM	Set semaphore
Unit	INT	1, 2 to 10	Channel
WVar1	INT	1 to 10	FunctionNumber
WVar2	WORD	0, 1	SemaphoreValue

PI service: TMCRTO

Create function tool:

Creating a tool under specification

- of an identifier, a Duplo number e.g. with
  $TC_TP1[y] = Duplo number;
  $TC_TP2[y] = "Tool identifer"

- and optionally a T Number e.g. with y = T Number

The tool is entered as existing in the tool directory area (TV). The first cutting edge "D1" (with zero contents) is created for tool offsets in the TO block. "D1" (with zero contents) is also set up for the monitoring data in the TS block, and simultaneously with zero contents for the OEM cutting edge data in the TUE block - if one is present. The TD block contains the identifier, duplo number and number of cutting edges (=1) for the T number that is entered optionally or allocated by the NCK.

If a TU block exists, it will contain the data set for the tool. After execution of the PI, the T number of the tool created is available in the TV block under **TnumWZV**.

Note

Before and after this PI service, the MMCSEM PI service must be called with the associated parameter "WVar1" for this PI service. See PI service MMCSEM for more information.

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.TMCRTO	Create tool
Unit	INT	1, 2	TOA
WVar1	INT		T number
WVar2	INT		Duplo number
Addr1	STRING	max. 32 characters	Tool identifier
T number > 0 means a T number must be specified			
T number = -1 means that the NCK should allocate a T number			
The example shows T number = -1 ⇒ T number assigned by NCK			

PI service: TMFDPL

Function Searching for empty location for loading:
(dependent on parameter assignment)

Location_number_to = -1, Magazine_number_to = -1:

Searches all magazines in the specified area (= channel) for an empty location for the tool specified with a T number. After execution of the PI, the magazine and locations numbers found during the search are listed in the configuration block of the channel (component **magCMCmdPar1** (magazine number) and **magCMCmdPar2** (location number)). Location_number_ID and magazine_number_ID can be set as search criteria or not (= -1). The PI is acknowledged positively or negatively depending on the search result.

Location_number_to = -1, Magazine_number_to = Magazine_number:

An empty location for the tool specified with a T number is searched for in the specified magazine. Location_number_ID and magazine_number_ID can be set as search criteria or not (= -1). The PI is acknowledged positively or negatively depending on the search result.

Location_number_to = Location_number, Magazine_number_to = Magazine_number:

The specified location is checked, to confirm that it is free to be loaded with the specified tool. Location_number_ID and magazine_number_ID can be set as search criteria or not (= -1). The PI is acknowledged positively or negatively depending on the search result.

Command parameters 1 and 2 are located at source.

| Loading: | If source is an internal loading magazine, then the command parameters are located at the target (a real magazine). |
| Unloading : | Source is always a real magazine. |

Note

Before and after this PI service, the MMCSEM PI service must be called with the associated parameter "WVar1" for this PI service. See PI service MMCSEM for more information.

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.TMFDPL	Empty location for loading
Unit	INT	1, 2	TOA
WVar1	INT		T number
WVar2	INT		Location_number _to
WVar3	INT		Magazine_number _to
WVar4	INT		Location_number _Bez
WVar5	INT		Magazine_number _ID

PI service: TMFPBP

Function Searching for empty location

(dependent on parameter assignment):

For more information please see the description of FB 7: PI_SERV2 General PI services):

PI service: TMMVTL

Prepare magazine location for loading, unload tool:

This PI service is used both to load and unload tools. Whether the PI initiates a loading or unloading operation depends on the assignment between the real locations and the "from" parameters and "to" parameters: Loading ⇒ 'From' = Loading point/station, unloading ⇒ 'To' = loading point/station

The TMMVTL PI service is used for all movements.

1. Loading and unloading (loading point ↔ magazine)

2. Loading and unloading (loading point ↔ buffer storage, e.g., spindle)

3. Relocation within a magazine

4. Relocation between different magazines

5. Relocation between magazine and buffer storage

6. Relocation within buffer storage

The following variables from the TM block are used to monitor case 1, 3, 4, 5:

magCmd (area no. = TO unit, line = magazine number)

magCmdState <- "acknowledgment"

The following variables from the TMC block are used to monitor case 2), 6):

magCBCmd (area no. = TO unit)

magCBCmdState <- "acknowledgment"

Load function

Prepares the specified real magazine for the specified channel for loading, i.e., traverses the magazine to the selected location for loading at the specified loading point/station (location_number_from, magazine_number_from) and inserts the tool.

When location_number_to = -1, an empty location for the tool specified by a T number is first sought in the specified magazine and the magazine then traversed. After execution of the PI, the number of the location found is listed in the TM area in component **magCMCmdPar2** for the **real** magazine of the channel.

With location_number_to = -2 and a valid magazine number, loading takes place into the currently queued magazine position of the specified magazine. After execution of the PI, the number of the location for tool loading is listed in the TM area in component **magCMCmdPar2** for the real magazine of the channel.

Unload function

The tool specified by the tool number is unloaded at the specified loading point/station (location_number_to, magazine_number_to), i.e., the magazine is traversed to the position for unloading and the tool is then removed. The magazine location for the tool is marked as being free in the TP block. The tool can be specified either via a T number or by means of the location and magazine numbers. The value -1 is entered at unused specification points.

Note

Before and after this PI service, the MMCSEM PI service must be called with the associated parameter "WVar1" for this PI service. See PI service MMCSEM for more information.

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.TMMVTL	Make magazine location ready for loading, unload tool
Unit	INT	1, 2	TOA
WVar1	INT		T number
WVar2	INT		Location_number_from
WVar3	INT		Magazine_number_from
WVar4	INT		Location_number_to
WVar5	INT		Magazine_number_to

PI service: TMPOSM

Function Position magazine location or tool:
(dependent on parameter assignment)

A magazine location, which has either been specified directly or qualified via a tool located on it, is traversed to a specified position (e.g., in front of a load location) via the PI service.

The PI service makes a magazine location, which can be qualified in various ways, traverse in front of a specified load location. The load location must be specified in the PI parameters "location number_from" and "magazine number_from" (obligatory!).

The magazine location to be traversed can be qualified by the following:

- T number of the tool

  The location where the tool is positioned traverses;
  the parameter "tool identifier", "duplo number", "location number_from" and magazine number_from" parameters are irrelevant (i.e., values "" , "-0001", "-0001", "-0001").

Or

- Tool identifier and duplo number

  The location where the tool is positioned traverses;
  the parameters "T number", "location number_from" and "magazine number_from" are irrelevant (i.e., value "-0001" each).

Or

- Direct specification of the location in the location_number_from and magazine_number_from parameters

  The tool-qualifying
  parameters "T number", "tool identifier" and "duplo number" are irrelevant
  (i.e., values "-0001", "", "-0001").

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.TMPOSM	Position magazine location or tool
Unit	INT	1, 2	TOA
Addr1	STRING	max. 32 characters	Tool identifier
WVar1	INT		T number
WVar2	INT		Duplo number
WVar3	INT		Location_number_from
WVar4	INT		Magazine_number_from
WVar5	INT		Location number_ref
WVar6	INT		Magazine number_ref

PI service: TMPCIT

Function Set increment value for workpiece counter:

Incrementing the workpiece counter of the spindle tool

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.TMPCIT	Set increment value for workpiece counter
Unit	INT	1 to 10	TOA
WVar1	WORD	0 ... max.	Spindle number; corresponds to the type index in the location data with spindle location type of the buffer magazine in channel.000 = main spindle
WVar2	WORD	0 ... max.	Increment value; indicates the number of spindle revolutions after which the workpiece counter is incremented

PI service: TMRASS

Function Reset the active status:

Resetting the active status on worn tools
This PI service is used to search for all tools with the tool status active and disabled. The active status is then canceled for these tools. Potentially appropriate times for this PI service are the negative edge of VDI signal "tool disable ineffective", an end of program, or a channel RESET. This PI service is intended mainly for the PLC, since it knows when the disabled tool is finally no longer to be used.

Parameterization			
Signal	Type	Value range	Description
PIService	ANY	PI. TMRASS	Reset active status
Unit	INT	1 to 10	TO area

PI service: TRESMO

Function Reset monitoring values:

This PI service resets the monitoring values of the designated edges of the designated tools to their setpoint (initial) values.
This only relates to tools with active monitoring.

Compare this with the `RESETMON` NC command.

Parameterization			
Signal	Type	Value range	Description
PIService	ANY	PI. TRESMO	Reset monitoring values
Unit	INT	1 to 10	TO area
WVar1	WORD	-max ..max	ToolNumber 0: Applies to all tools >0: Applies only to this tool <0: Applies to all sister tools of the specified T No.
WVar2	WORD	0 ... max.	D number >0: Monitoring of specified edge of specified tools is reset. 0: Monitoring of all edges of specified tools is reset.
WVar3	WORD	0 ...15	Monitoring types Type of monitoring to be reset. This parameter is binary-coded. 1: Tool-life monitoring is reset. 2: Count monitoring is reset. 4: Wear monitoring is reset. 8: Sum offset monitoring is reset. Combinations of monitoring types can be reset by adding the values above. 0: All active tool-monitoring functions ($TC_TP9) are reset.

PI service: TSEARC

Function complex search via search screen forms:
(dependent on parameter assignment)

The PI service allows you to search for tools with specified properties within a search domain (in one or more magazines starting and ending at a specific location). The specified properties refer only to data of the tools and their cutting edges.

The PI service is only available if tool management is activated.

You can define a search direction and the number of hits for the PI service (e.g., one tool for the next tool with matching properties or all tools with the specified properties).

As a result of this service, the user who made the call receives a list of the internal T numbers of the tools found.

The search criteria can only be specified as AND operations. If an application needs to define an OR operation for the search criteria, it must first execute a series of queries with AND criteria and then combine/evaluate the results of the individual queries.

To assign the parameters of the PI service, the properties of the required tools are first defined via variable service in the TF block. To do this, in the block TF in the operand screens (parMaskT...), the relevant

- comparison criteria (which tool data are to be compared?) are highlighted,
- Comparison operator data (parDataT...) are filled with the corresponding comparison types (==, <, >, <=, >=, &&) to be done
- and the comparison values are entered in the operand data.

The PI service is then initiated and, after its successful return, the variable service from the TF block is used to read out the number of hits in the variable resultNrOfTools and the result list in the variable resultToolNr (i.e., the list of internal T numbers of the tools found in the search - resultNrOfTools quantity).

The PI service must be encapsulated with a semaphore from its preparation until the successful return of the result. This is the only way to ensure exclusive access and the exclusive use of the TF block in conjunction with the TSEARC PI service. The function number provided for the semaphore feature (PI service MMCSEM) is the function number for TSEARC.

If the service is configured incorrectly, a malfunction occurs. In all other cases, it will return a result, even if no tools are found (resultNrOfTools = 0).

The search domain can be defined as follows in the parameters "MagNrFrom", "PlaceNrFrom", "MagNrTo", "PlaceNrTo":

MagNr From	PlaceNr From	MagNr To	PlaceNr To	Search area
WVar1	WVar2	WVar3	WVar4	
#M1	#P1	#M2	#P2	Locations starting at magazine #M1, location #P1 up to magazine #M2, location #P2 are searched
#M1	-1	#M1	-1	All locations in magazine #M1 - and no others - are searched
#M1	-1	-1	-1	All locations starting at magazine #M1 are searched
#M1	#P1	-1	-1	All locations starting at magazine #M1 and location #P1 are searched
#M1	#P1	#M1	-1	Locations in magazine #M1 starting at magazine #M1 and location #P1 in this magazine are searched
#M1	#P1	#M2	-1	Locations starting at magazine #M1, location #P1 up to magazine #M2, location #P2 are searched
#M1	-1	#M2	#P2	Locations starting at magazine #M1 up to magazine #M2, and in that location #P2 are searched
#M1	-1	#M2	-1	Locations starting at magazine #M1 up to and including magazine #M2 are searched
-1	-1	-1	-1	All magazine locations are searched

For a symmetric search (see parameter "SearchDirection")

- the search domain may stretch only over a single magasine (cases 2 and 5 from the table given above). If another search domain is specified, the service will malfunction.

- a reference location must be specified in the parameters "MagNrRef" and "PlaceNrRef", with respect to which the symmetric search is done.

The reference location is a buffer location (a location from the magazine buffer, i.e., change position, gripper, etc.) or a load point (a location from the internal loading magazine). The search is executed symmetrically with reference to the magazine location before the specified reference location. A multiple assignment to the magazine being searched must be configured in the TPM block for the reference location. If this is not the case, a malfunction occurs. If the magazine location in front of the reference location is outside the search domain, the service responds as if it has not found a matching location.

Note

Before and after this PI service, the MMCSEM PI service must be called with the associated parameter "WVar1" for this PI service. See PI service MMCSEM for more information.

Parameterization:

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.TSEARC	Complex search using search screen forms
Unit	INT	1, 2	TOA
WVar1	INT		MagNrFrom Magazine number of magazine from which search must begin
WVar2	INT		PlaceNrFrom Location number of location in magazine MagNrFrom, at which search must begin
WVar3	INT		MagNrTo Magazine number of magazine at which search must end
WVar4	INT		PlaceNrTo Location number of location in magazine MagNrTo, at which search must end
WVar5	INT		MagNrRef Magazine number of (internal) magazine, with reference to which the symmetrical search is to be performed. (this parameter is only relevant with a "symmetrical" search direction)
WVar6	INT		PlaceNrRef Location number of location in magazine MagNrRef, with reference to which the symmetrical search is to be performed. This parameter is only relevant with a "symmetrical" search direction
WVar7	INT	1, 2, 3	SearchDirection specifies the required search direction. 1: Forwards from the first location of the search domain 2: Backwards from the last location of the search domain 3: Symmetrical with real magazine location positioned in front of the location specified by MagNrRef and PlaceNrRef
WVar8	INT	0, 1, 2, 3	KindofSearch 0: Find all tool with this property cutting edge specifically 1: Search for the first tool found with this property (cutting edge specifically) 2: Browse all cutting edges to find all tool with this property 3: Browse all tools to search for the first tool found with this property

Call example

Program selection in channel 1 (main program and workpiece program)

Entry of PI for DB 16 and STR for DB 124 with the S7 SYMBOL editor:

Parameterization		
Symbol	Operand	Data type
PI	DB 16	DB 16
STR	DB 124	DB 124

```
DATA_BLOCK DB 126            //Unassigned user DB, as instance for FB 4
FB 4
BEGIN
END_DATA_BLOCK


DATA_BLOCK DB 124
   struct
       PName:         string[32]:= '_N_TEST_MPF
                      ';
       Path:          string[32]:=               //Main program
                      '/_N_MPF_DIR/';
       PName_WST:     string[32]:= '_N_ABC_MPF';
       Path_WST:      string[32]:=               //Workpiece program
                      '/_N_WKS_DIR/_N_ZYL_WPD';
   end_struct
BEGIN
END_DATA_BLOCK
Function FC "PICall" : VOID
   U  I 7.7;        //Unassigned machine control panel key
   S  M 0.0;        //Activate req.
   U  M 1.1;        //Done completed message
   R  M 0.0;        //Terminate job
   U  I 7.6;        //Manual error acknowledgment
   U  M 1.0;        //Error pending
   R  M 0.0;        //Terminate job


   CALL FB 4, DB 126 (
       Req :=       M0.0,
       PIService := PI.SELECT,
       Unit :=      1,                        // CHAN 1
       Addr1 :=     STR.Path,
       Addr2 :=     STR.PName,                //Main-program selection
                    //Addr1:=STR.Path_WST,
                    //Addr2:=STR.PName_WST,    //Workpiece-program
                                              selection
       FMNCNo :=    1,                        //(in FMNC only)
       Error :=     M1.0,
       Done :=      M1.1,
       State :=     MW2);
```

2.12.5   FB 5: GETGUD read GUD variable

Description of Functions

The PLC user program can read a GUD variable (GUD = Global User Data) from the NCK or channel area using the FB GETGUD. The FB is multi-instance-capable. Capital letters must be used for the names of GUD variables. Every FB 5 call must be assigned a separate instance DB from the user area.

A job is started when **FB 5 is called** by means of a positive edge change at control input "Req". This job includes the name of the GUD variable to be read in parameter "Addr" with data type "STRING". The pointer to the name of the GUD variables is assigned symbolically to the "Addr" parameter with <DataBlockName>.<VariableName>. Additional information about this variable is specified in parameters "Area", "Unit", "Index1" and "Index2" (see table of block parameters).

When parameter "CnvtToken" is activated, a variable pointer (token) can be generated for this GUD variable as an option. This pointer is generated via the VAR selector for system variables of the NC. Only this method of generating pointers is available for GUD variables. Once a pointer has been generated for the GUD variable, then it is possible to read and write via FB 2 and FB 3 (GET, PUT) with reference to the pointer. This is the only method by which GUD variables can be read. When FB 2 or FB 3 is parameterized, only parameter Addr1 ... Addr8 needs to be parameterized for this GUD variable pointer. GUD variable fields are an exception. In these, Line1 .. Line8 must also be parameterized with the field index of this variable.

Successful completion of the read process is indicated by a logical "1" in status parameter "Done".

The read process extends over several PLC cycles generally 1 to 2).
The block can be called up in cyclic mode only.

Any errors are indicated by Error and State.

Note

FB 5 can read GUD variables only if basic program parameter NCKomm has been set to "1" (in OB 100: FB 1, DB 7).

Declaration of the function

```
FUNCTION_BLOCK FB 5                              //Server name
    KNOW_HOW_PROTECT
     VERSION : 3.0

VAR_INPUT
    Req :           BOOL ;
    Addr:           ANY ;                        //Variables name string
    Area:           BYTE ;                       //Area: NCK = 0, channel = 2
    Unit :          BYTE ;
    Index1:         INT ;                        //Field index 1
    Index2:         INT ;                        //Field index 2
    CnvtToken:      BOOL ;                       //Conversion into 10-byte token
VarToken:           ANY ;                        //Struct with 10 bytes for the variable token
FMNCNo :            INT ;                         //(in FMNC only)


END_VAR


VAR_OUTPUT
    Error :         BOOL ;
    Done :          BOOL ;
    State :         WORD ;

END_VAR


VAR_IN_OUT
    RD:             ANY ;

END_VAR


BEGIN
END_FUNCTION_BLOCK
```

Description of formal parameters

The table below lists all formal parameters of the GETGUD function.

Signal	I/O	Type	Value range	Remark
Req	I	BOOL		Job start with positive signal edge
Addr	I	ANY	[DBName].[VarName]	GUD variable name in a variable of data type STRING
Area	I	BYTE		Area address: 0: NCK variable 2: Channel variables
Unit	I	BYTE		NCK area: Unit:=1 Channel area: Channel no.
Index1	I	INT		Field index 1 of variable Variable has the value 0 if no field index is used.
Index2	I	INT		Field index 2 of variable Variable has the value 0 if no field index is used.
CnvtToken	I	BOOL		Activate generation of a variable token
VarToken	I	ANY		Address to a 10byte token (see example)
FMNCNo (on FMNC only)	I	INT	0, 1, 2	0, 1=1 NCU, 2=2 NCUs
Error	Q	BOOL		Job negatively acknowledged or not executable
Done	Q	BOOL		Job successfully executed.
State	Q	WORD		See error identifiers
RD	I/O	ANY	P#Mm.n BYTE x... P#DBnr.dbxm.n BYTE x	Data to be written

Error identifiers

If it was not possible to execute a job, the failure is indicated by "logic 1" on status parameter error. The error cause is coded at the block output State:

State		Meaning	Note
WORD H	WORD L		
0	1	Access error	
0	2	Error in job	Incorrect compilation of Var. in a job
0	3	Negative acknowledgment, job not executable	Internal error, any help: NC RESET
0	4	Data areas or data types do not tally	Check data to be read in RD
0	6	FIFO full	Job must be repeated, since queue is full
0	7	Option not set	BP parameter "NCKomm" is not set
0	8	Incorrect target area (SD)	RD may not be local data
0	9	Transmission occupied	Job must be repeated
0	10	Error in addressing	Unit contains value 0
0	11	Address of variable invalid	Check Addr (or variable name), area, unit

Configuration steps

To be able to read a GUD variable, its name must be stored in a string variable. The data block with this string variable must be defined in the symbol table so that the "Addr" parameter can be assigned symbolically for FB GETGUD. A structure variable can be defined optionally in any data area of the PLC to receive the variable pointer (see specification in following example).

Pulse diagram

(1)	Activation of function
(2)	Positive acknowledgment: variables have been written
(3)	Reset function activation after receipt of acknowledgment
(4)	Signal change by means of FB
(5)	If function activation is reset prior to receipt of acknowledgment, the output signals are not updated without the operational sequence of the activated function being affected.
(6)	Negative acknowledgment: Error has occurred, error code in output parameter state.

Call example

Reading of a GUD variable with the name "GUDVAR1" as an integer variable
(see also table in FB 2: Assignment of NC data type in SIMATIC data type).

Call and parameterization of FB 5 with instance DB 111:

```
DATA_BLOCK DB GUDVAR                          //Assignment to symbol table
    STRUCT
    GUDVar1 : STRING[32] := 'GUDVAR1';        //Name is defined by user
    GUDVar1T :
    STRUCT
    SYNTAX_ID :              BYTE ;
    area_and_unit :          BYTE ;
    column :                 WORD ;
    line :                   WORD ;
    block type :             BYTE ;
    NO. OF LINES :           BYTE ;
    type :                   BYTE ;
    length :                 BYTE ;
    END_STRUCT;
END_STRUCT;


BEGIN
END_DATA_BLOCK
DATA_BLOCK DB 111                             //Unassigned user DB, as instance for FB 5
```

```
FB 5
BEGIN
END_DATA_BLOCK
//A user-defined channel variable from channel 1 must be read
//with conversion into a variable pointer to allow subsequent
//writing of a variable.
Function FC "VariablenCall" : VOID
    U   I 7.7;
    S   M 100.0;
    U   M 100.1;
    R   M 100.0;
    U   I 7.6;
    U   M 102.0;
    R   M 100.0;


    CALL FB 5, DB 111 (
        Req :=          M 100.0,
        Addr :=         GUDVAR.GUDVar1,
        Area :=         B#16#2,
        Unit :=         B#16#1,
        Index1 :=       0,
        Index2 :=       0,
        CnvtToken :=    TRUE,
        FMNCNo :=       1
        VarToken :=     GUDVAR.GUDVar1T,
        Error :=        M102.0,
        Done :=         M100.1,
        State :=        MW104,
        RD :=           P#DB99.DBX0.0 DINT 1);
```

//Unassigned machine control panel key

//Activate req.

//Done completed message

//Terminate job

//Manual error acknowledgment

//Error pending

//Terminate job

//Starting edge for reading

//Channel variable

//Channel 1

//No field index

//No field index

//Conversion into 10-byte token

//(in FMNC only)

2.12.6   FB 7: PI_SERV2 General PI services

Description of Functions

A detailed description of the FB 7 is contained in the description of FB 4.
The call is permitted only in cyclic program OB 1.

The only difference to FB 4 is the number of WVar1 and subsequent parameters. In FB 7,
WVar1 to WVar16 are defined in VAR_INPUT (FB4 has WVar1 to WVar10). All other
parameters are identical to FB 4.
This PI server can be used for all PI services previously implemented with FB 4. In addition,
the PI services listed below can only be handled with FB 7.

Declaration of the function

```
FUNCTION_BLOCK FB 7

Var_INPUT
        Req :           BOOL ;
        PIService :     ANY ;
        Unit :          INT ;
        Addr1 :         ANY ;
        Addr2 :         ANY ;
        Addr3 :         ANY ;
        Addr4 :         ANY ;
        WVar1 :         WORD ;
        WVar2 :         WORD ;
        WVar3 :         WORD ;
        WVar4 :         WORD ;
        WVar5 :         WORD ;
        WVar6 :         WORD ;
        WVar7 :         WORD ;
        WVar8 :         WORD ;
        WVar9 :         WORD ;
        WVar10 :        WORD ;
        WVar11 :        WORD ;
        WVar12 :        WORD ;
        WVar13 :        WORD ;
        WVar14 :        WORD ;
        WVar15 :        WORD ;
        WVar16 :        WORD ;
        FMNCNo :        INT ;                          //(in FMNC only)
END_VAR

VAR_OUTPUT
        Error :         BOOL ;
        Done :          BOOL ;
        State :         WORD ;
END_VAR
```

Description of formal parameters

The following table shows all formal parameters of the function PI_SERV2.

Signal	Type	Type	Value range	Remark
Req	I	BOOL		Job request
PIService	I	ANY	[DBName].[VarName] Default is: "PI".[VarName]	PI service description
Unit	I	INT	1...	Area number
Addr1 to Addr4	I	ANY	[DBName].[VarName]	Reference to strings specification according to selected PI service
WVar1 to WVar16	I	WORD	1...	Integer or word variables. Specification according to selected PI service,
FMNCNo (FMNC only)	I	INT	0, 1, 2	0, 1=1 NCU, 2=2 NCUs
Error	A	BOOL		Negative acknowledgment of job or execution of job impossible
Done	Q	BOOL		Job successfully executed
State	Q	WORD		See error identifiers

Overview of additional PI services

The following additional PI-services for those of FB 4 can be started from PLC. The meaning and application of the general FB 7 input variables (Unit, Addr ..., WVar ...) depend on the individual PI service concerned.

PI service	Function	Available in
TMFPBP	Empty location search	**SINUMERIK 840D/810D** *(with SW 4 and higher)

PI service: TMFPBP

Function: Search for empty location
(dependent on parameter assignment)

This service searches the magazine(s) named in the relevant parameters for an empty location, which meets the specified criteria (tool size and location type). The result of the empty location search can be fetched from variables magCMCmdPar1 (magazine number) and magCMCmdPar2 (location number) in block TMC when the service has functioned correctly. As the PI service stores a result in variables magCMCmdPar1 and magCMCmdPar2, the service must be protected by the semaphore mechanism (PI service MMCSEM) with the function number for _N_TMFDPL in cases where several control units or PLCs are operating on one NC. The search area can be predefined in the following way by setting parameters "MagazineNumber_From", "LocationNumber_From", "MagazineNumber_To", "LocationNumber_To":

MagazinNumber_From	LocationNumber_From	MagazineNumber_To	LocationNumber_To	Search area
WVar1	WVar2	WVar3	WVar4	
#M1	#P1	#M1	#P1	Only location #P1 in magazine #M1 is checked
#M1	#P1	#M2	#P2	Locations starting at magazine #M1, location #P1 up to magazine #M2, location #P2 are searched
#M1	-1	#M1	-1	All locations in magazine #M1 - and no others - are searched
#M1	-1	-1	-1	All locations starting at magazine #M1 are searched
#M1	#P1	-1	-1	All locations starting at magazine #M1 and location #P1 are searched
#M1	#P1	#M1	-1	Locations in magazine #M1 starting at magazine #M1 and location #P1 in this magazine are searched
#M1	#P1	#M2	-1	Locations starting at magazine #M1, location #P1 up to magazine #M2, location #P2 are searched
#M1	-1	#M2	#P2	Locations starting at magazine #M1 up to magazine #M2, and in that location #P2 are searched
#M1	-1	#M2	-1	Locations starting at magazine #M1 up to and including magazine #M2 are searched
-1	-1	-1	-1	All magazine locations are searched

Note

Before and after this PI service, the MMCSEM PI service must be called with the associated parameter "WVar1" for this PI service. See PI service MMCSEM for more information.

Parameterization			
Signal	Type	Value range	Meaning
PIService	ANY	PI.TMFPBP	Empty location search
Unit	INT	1 ... max. TOA	TOA
WVar1	INT		MagazinNumber_From: Magazine number of magazine from which search must begin
WVar2	INT		LocationNumber_From: Location number of location in magazine MagazineNumber_From at which search must begin
WVar3	INT		MagazineNumber_To: Magazine number of magazine at which search must end
WVar4	INT		LocationNumber_To: Location number of location in magazine MagazineNumber_To at which search must end
WVar5	INT		MagazineNumber_Ref:
WVar6	INT		LocationNumber_Ref:
WVar7	INT	0, 1 .. 7	Number of required half locations to left
WVar8	INT	0, 1 .. 7	Number of required half locations to right
WVar9	INT	0, 1 .. 7	Number of required half locations in upward direction
WVar10	INT	0, 1 .. 7	Number of required half locations in downward direction
WVar11	INT		Number of required location type
WVar12	INT	0: default 1: forwards 2: backwards 3: Symmetrical	Specifies the required search direction 0: Empty location search strategy is set in $TC_MAMP2

2.12.7   FB 9: MzuN Control unit switchover

Description of Functions

This block allows switchover between several **control units** (operator panels, MMC handheld units and/or MCP machine control panels), which are connected to one or more NCU control modules via a bus system.

References:
/FB2/ Function Manual, Expansion Functions; Several Control Panels on Multiple NCUs, Decentralized Systems (B3)

The **interface** between the individual control units and the NCU (PLC) is the M : N Interface in the data block DB 19; see
/FB2/ Function Manual Extended Functions, Chapter "Data lists, Signal description".

FB 9 uses the signals of these interfaces.

Apart from initialization, sign-of-life monitoring and error routines, the following **basic functions** are also performed by the block for operating-unit switchover:

Tabulated overview of functions:	
Basic function	**Meaning**
MMC call waiting	MMC wants to go online to an NCU
MMC coming	MMC is connecting to an NCU
MMC going	MMC is disconnecting from an NCU
Forced break	MMC must break connection to an NCU
Operating focus changeover to server mode	Change operating focus from one NCU to the other
Active/passive operating mode:	Operator control and monitoring/monitoring only
MCP switchover	As an option, MCP can be switched over with the MMC

Brief description of a few important functions

Active/passive operating mode

An online MMC can operate in two different modes:

Active mode:	Operator can control and monitor
Passive mode:	Operator can monitor (MMC header only)

After switchover to an NCU, this initially requests active operating mode in the PLC of the online NCU. If two MMCs are simultaneously connected online to one NCU, one of the two is always in active and the other in passive operating mode. The operator can request active mode on the passive control unit at the press of a button.

MCP switchover

As an option, an MCP assigned to the MMC can be switched over at the same time. This can be done on condition that the MSTT address is entered in parameter "mstt_adress" of configuration file NETNAMES.INI in the operator panel and MCPEnable is set to TRUE. The MSTT of the passive MMC is deactivated. so that there is only ever one active MCP on an NCU at one time.

Boot condition

To prevent the previously selected MCP being reactivated when the NCU is restarted, input parameters
MCP1BusAdr = **255** (address of 1st MCP) and **MCP1STOP =TRUE** (deactivate 1st MCP) must be set when FB1 is called in OB 100.

Releases

When one MCP is switched over to another, any active feedrate or axis enabling signals may be transferred at the same time.

Note

Keys actuated at the moment of switchover remain operative until the new MCP is activated (by the MMC which is subsequently activated). The override settings for feedrate and spindle also remain valid. To deactivate the activated keys, in case of the falling edge of the signal DB10, ... DBX104.0 (MSTT 1 ready) the input image of the machine control signal is to be laid on the non-actuated signal level. The override settings should remain unchanged. Measures for deactivating keys must be implemented in the PLC user program, see example "Override switchover".

The call is permitted only in cyclic program OB 1.

Declaration of function

```
FUNCTION_BLOCK FB 9
VAR_INPUT
     Ack :                BOOL ;              //Acknowledge interrupts
     OPMixedMode:         BOOL:= FALSE;       //Mixed operation with non-M-to-N-enabled OP
                                              //deactivated
     ActivEnable:         BOOL:= TRUE;        // Not supported.
                                              //Control panel switchover Interlocking via
                                              //MMCx_SHIFT_LOCK in DB 19
     MCPEnable :          BOOL:= TRUE;        // Activate MCP switchover
END_VAR
VAR_OUTPUT
     Alarm1 :             BOOL ;              // Interrupt: Error in HMI bus address, bus type!
     Alarm2 :             BOOL ;              // Interrupt: No confirmation MMC_1 offline!
     Alarm3 :             BOOL ;              // Interrupt: MMC_1 is not going offline!
     Alarm4 :             BOOL ;              // Interrupt: No confirmation MMC_2 offline!
     Alarm5 :             BOOL ;              // Interrupt: MMC_2 is not going offline!
     Alarm6 :             BOOL ;              // Interrupt: Queuing MMC is not going online!
     Report :             BOOL ;              // Message: Sign-of-life monitoring MMC
     ErrorMMC :           BOOL ;              // Error detection MMC
END_VAR
```

Description of formal parameters

The table below lists all formal parameters of the M:N function.

Formal parameters of M:N function			
Signal	I/O	Type	Remark
Ack	I	BOOL	Acknowledge alarms
OPMixedMode	I	BOOL	Mixed mode deactivated for OP without M to N capability
ActivEnable	I	BOOL	// Function not supported. Control panel switchover Interlocking via MMCx_SHIFT_LOCK in DB 19
MCPEnable	I	BOOL	Activate MSTT switchover **TRUE =** MSTT is switched over with operator panel front. **FALSE: =** MCP is not switched over with operator panel front. This can be used to permanently link an MCP. See also MMCx_MCP_SHIFT_LOCK in DB 19
Alarm1	Q	BOOL	Alarm: Error in MMC bus address, bus type!
Alarm2	Q	BOOL	Alarm: No confirmation MMC 1 offline!
Alarm3	Q	BOOL	Alarm: MMC 1 is not going offline!
Alarm4	Q	BOOL	Alarm: No confirmation MMC 2 offline!
Alarm5	Q	BOOL	Alarm: MMC 2 is not going offline!
Alarm6	Q	BOOL	Alarm: Queuing MMC is not going online!
Report	Q	BOOL	Message: Sign-of-life monitoring MMC
ErrorMMC	Q	BOOL	Error detection MMC

Note

The block must be called by the user program. The user must provide an instance DB with any number for this purpose. The call is multi-instance-capable.

Example of a call for FB 9

```
CALL    FB 9,  DB 109  (
     Ack               := Error_ack,           //e.g. MSTT-RESET
     OPMixedMode       := FALSE,
     ActivEnable       := TRUE,                 //
     MCPEnable         := TRUE,                 // Enable MCP switchover
     Alarm1            := DB2.dbx188.0,         // Error message 700.100
     Alarm2            := DB2.dbx188.1,         // Error message 700.101
     Alarm3            := DB2.dbx188.2,         // Error message 700.102
     Alarm4            := DB2.dbx188.3,         // Error message 700.103
     Alarm5            := DB2.dbx188.4,         // Error message 700.104
     Alarm6            := DB2.dbx188.5,         // Error message 700.105
     Report            :=DB2.dbx192.0);         //Operational message 700,132
```

Note

Input parameter "MCPEnable" must also be set to TRUE to enable MSTT switchover. The default value of these parameters is set in this way and need not be specially assigned when the function is called.

Alarm, error

The output parameters "Alarm1" to "Alarm6" and "Report" can be transferred to the DB2 areas for MMC alarm and error messages.

If execution of an MMC function has failed (for which an appropriate error message cannot be displayed), status parameter ErrorMMC is set to 'logical 1' (e.g., booting error when no connection is made).

Call example for FB 1 (Call in OB 100)

```
CALL "RUN_UP", "gp_par" (
     MCPNum               := 1,
     MCP1In               := P#I 0.0,
     MCP1Out              := P#Q 0.0,
     MCP1StatSend         := P#Q 8.0,
     MCP1StatRec          := P#Q 12.0,
     MCP1BusAdr           := 255,                 // Address of first MCP
     MCP1Timeout          := S5T#700MS,
     MCP1Cycl             := S5T#200MS,
     MCP1Stop             := TRUE,                // MCP disabled
     NCCyclTimeout        := S5T#200MS,
     NCRunupTimeout        := S5T#50S);
```

Example of override switchover

```
// Auxiliary flags used M100.0, M100.1, M100.2, M100.3
// Positive edge of MCP1Ready must check override and actions for activation
// Initiate MCP block
//This example applies to the feedrate override;
//The interface and input bytes must be exchanged for spindle override.
```

U	DB10.DBX	104.0;	//MCP1Ready
EN	M	100.0;	//Edge trigger flag 1
JCN smth1;			
S	M	100.2;	//Set auxiliary flag 1
R	M	100.3;	//Reset auxiliary flag 2

```
// Save override
```

	L DB21.DBB 4;		//Feed override interface
	T EB 28;		//Buffer storage (freely assignable input or memory byte)
wei1:			
U	M	100.2;	//Switchover takes place
O	DB10.DBX	104.0;	//MCP1Ready
JCN smth2;			
U	DB10.DBX	104.0;	//MCP1Ready
FP	M	100.1;	//Edge trigger flag 2
JC smth2;			
U	M	100.2;	//Switchover takes place
R	M	100.2;	//Reset auxiliary flag 1
JC smth2;			
U	M (GND)	100.3;	//Comparison has taken place
JC MCP;			//Call MCP program

```
// Guide the stored override to the interface of the switched MCP
// until the override values match
```

L	EB28;	//Buffer storage open
T DB21.DBB 4;		//Guide override interface
L EB 3;		//Override input byte for feed
<>i;		//Match?
JC smth2;		//No, jump
S	M100.3;	//Yes, set auxiliary flag 2

```
// When override values match, call the MCP program again
```

MCP: CALL "MCP_IFM"(		//FC 19
BAGNo	:= B#16#1,	
ChanNo	:= B#16#1,	
SpindleIFNo	:= B#16#0,	
FeedHold	:= M 101.0,	
SpindleHold	:= M 101.1);	
wei2: NOP	0;	

2.12.8   FB 10: Safety relay (SI relay)

Description of Functions

The SPL block "Safety relay" for "Safety Integrated" is the PLC equivalent of the NC function of the same name. The standard SPL "Safety relay" block is designed to support the implementation of an emergency stop function with safe programmable logic. However, it can also be used to implement other similar safety functions, e.g., control of a protective door. The function contains 3 input parameters (In1, In2, In3). On switchover of one of these parameters to the value 0, the output Out0 is deactivated without delay and outputs Out1, Out2 and Out3 deactivated via the parameterized timer values (parameters TimeValue1, TimeValue2, TimeValue3). The outputs are activated again without delay, if inputs In1 to In3 take the value 1 and a positive edge change is detected at one of the acknowledgement inputs Ack1, Ack2. To bring the outputs to their basic setting (values = 0) after booting, the parameter "FirstRun" must be configured as follows. The parameter "FirstRun" must be switched to the value TRUE via a retentive data (memory bit, bit in data block) on the first run after control booting. The data can be preset, e.g., in OB 100. The parameter is reset to FALSE when FB 10 is executed for the first time. Separate data must be used for parameter "FirstRun" for each call with its own instance.

The corresponding NCK-SPL-component is described in:
References:
/FBSI/Description of Functions, Safety Integrated

Simplified block diagram in CSF

The figure below shows only one acknowledgment input Ack1 and one delayed deactivation output Out1. The circuit for Ack2 and the other delayed outputs are identical. The parameter FirstRun is also missing in the function diagram. The mode of operation is described above.

Netzwerk 2: Titel:

Vereinfachter Logikplan des Sicherheitsrelais. In der Darstellung ist der Timer
T10 für die Ausschaltverzögerung verwendet.

Netzwerk 3: Titel:

Nach Ablauf der Zeit TimeValue1 wird Ausgang Out1 = False
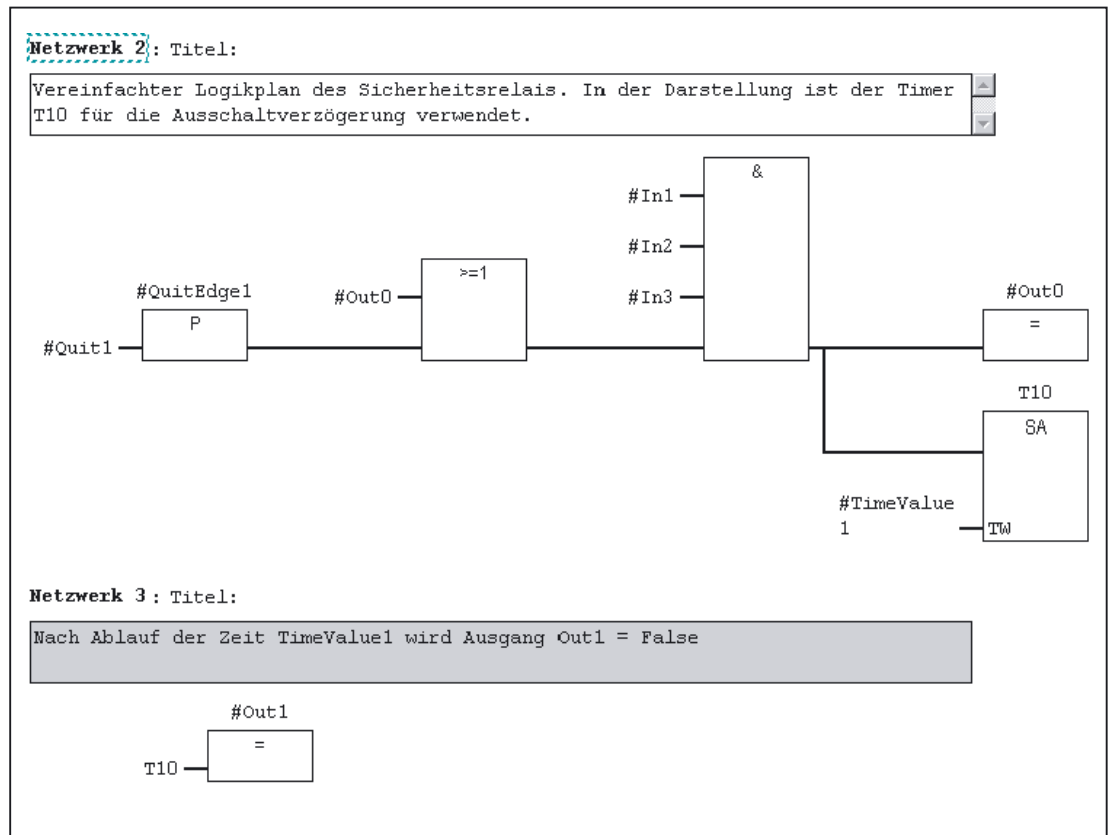
Declaration of the function

```
FUNCTION_BLOCK FB 10
VAR_INPUT
    In1 : BOOL          := TRUE;              //Input 1
    In2 : BOOL          := TRUE;              //Input 2
    In3 : BOOL          := TRUE;              //Input 3
    Ackn1 :             : BOOL ;              //Ack 1 signal
    Ackn2 :             :BOOL;                //Ack 2 signal
    TimeValue1 :        TIME := T#0ms ;       //TimeValue for output 1
    TimeValue2 :        TIME := T#0ms ;       //TimeValue for output 2
    TimeValue3 :        TIME := T#0ms ;       //TimeValue for output 3
END_VAR
VAR_OUTPUT
    Out0                : BOOL ;              //Output without delay
    Out1                : BOOL ;              //Delayed output to false by timer 1
    Out2                : BOOL ;              //Delayed output to false by timer 2
    Out3                : BOOL ;              //Delayed output to false by timer 3
END_VAR
VAR_INOUT
    FirstRun            : BOOL ;              //TRUE by user after initial start of SPL
END_VAR
```

Description of formal parameters

The following table shows all formal parameters of the SI relay function.

Formal parameters of SI relay function			
Signal	Type	Type	Remark
In1	I	BOOL	Input 1
In2	I	BOOL	Input 2
In3	I	BOOL	Input 3
Ackn1	I	BOOL	Acknowledge input 1
Ackn2	I	BOOL	Acknowledge input 2
TimeValue1	I	TIME	Time value 1 for OFF delay
TimeValue2	I	TIME	Time value 2 for OFF delay
TimeValue3	I	TIME	Time value 3 for OFF delay
Out0	A	BOOL	Output, instantaneous (no delay)
Out1	A	BOOL	Output, delayed by TimeValue1
Out2	A	BOOL	Output, delayed by TimeValue2
Out3	A	BOOL	Output, delayed by TimeValue3
FirstRun	I/O	BOOL	Activation of initial state

Note

The block must be called cyclically by the user program once following SPL program startup. The user must provide an instance DB with any number for this purpose. The call is multi-instance-capable.

2.12.9   FB 11: Brake test

Description of Functions

The braking operation check should be used for all axes, which must be prevented from moving in an uncontrolled manner by a holding brake. This check function is primarily intended for the socalled "vertical axes".

The machine manufacturer can use his PLC user program to close the brake at a suitable moment in time (guide value every 8 hours, similar to the SI test stop) and allow the drive to produce an additional torque/additional force equivalent to the weight of the axis. In errorfree operation, the brake can produce the necessary braking torque/braking force and keep the axis at a virtual standstill. When an error occurs, the actual position value exits the parameterizable monitoring window. In this instance, the position controller prevents the axis from sagging and negatively acknowledges the mechanical brake test.

The necessary parameterization of NC and the drive is described in:

References:

/FBSI/ Function Description Safety Integrated

The brake test must always be started when the axis is at a standstill. For the entire duration of the brake test, the enable signals of the parameterized axis must be set to Enable (e.g., the servo disable, feedrate enable signals). Further, the signal to axis/spindle DB31, ... DBX28.7 (PLC controls axis) is to be set to state 1 by the user program during the complete test.

Before activating the signal DB31, ... DBX28.7 (PLC controls axis) the axis is to be switched as "neutral axis" e.g. the DB31 is, ... DB8.0-8.3 (assign NC-axis channel) is to be set to channel 0 as well as DB31, ... DB8.4 (activation signal when the byte is changed) is to be set.

The checkback signal

- about the current state can be queried in the byte DB31, ... DBB68.

- the NC about the signal DB31, ... DBX63.1 (PLC controls axis) is to be awaited before the block is started. The direction in which the drive must produce its torque/force is specified by the PLC in the form of a "traversing motion" (e.g., via FC 18).

The axis must be able to reach the destination of this movement without risk of collision if the brake is unable to produce the necessary torque/force.

Note

Instructions for FC 18

In case FC 15, FC 16 or FC 18 are called for the same axis in the remainder of the user programs, the calls must be mutually interlocked. For example, this can be achieved via a common call of this function with an interlocked common data interface for the FC 18 parameters. A second option is to call the FC18 repeatedly, in which case the inactive FC18 will not be processed by the program. A multiple-use interlock must be provided.

The brake test is divided into the following steps:

Brake test sequence		
Step	Expected checkback	Monitoring time value
Start brake test	DBX 71.0 = 1	TV_BTactiv
Close brake	Bclosed = 1	TV_Bclose
Issue travel command	DBX 64.6 Or DBX 64.7	TV_FeedCommand
Issue test travel command	DBX62.5 = 1	TV_FXSreached
Wait for hold time	DBX62.5 = 1	TV_FXShold
Deselect brake test/ open brake	DBX71.0 = 0	TV_BTactiv
Issue Test O.K.		

Declaration of the function

```
Function_BLOCK FB 11
VAR_INPUT
    Start           : BOOL ;              //Start of brake test
    Ack             : BOOL ;              //Acknowledge error
    Bclosed         : BOOL ;              //Brake closed input (single channel - PLC)
    Axis            : INT ;               //Testing axis no.
    TimerNo         : TIMER ;             //Timer from user
    TV_BTactiv      : S5TIME ;            //TimeValue -> brake test active
    TV_Bclose       : S5TIME ;            //TimeValue -> close brake
    TV_FeedComm     : S5TIME ;            //TimeValue -> force FeedCommand
    and
    TV_FXSreache    : S5TIME ;            //TimeValue -> fixed stop reached
    d
    TV_FXShold      : S5TIME ;            //TimeValue -> test brake
END_VAR
VAR_OUTPUT
    CloseBrake      : BOOL ;              //Signal close brake
    MoveAxis        : BOOL ;              //Do move axis
    Done            : BOOL ;
    Error           : BOOL ;
    State           : BYTE ;              //Error byte
END_VAR
```

Description of formal parameters

The following table lists all of the formal parameters of the brake test function

Formal parameters of brake test function			
Signal	**I/O**	**Type**	**Remark**
Start	I	BOOL	Start brake test
Ack	I	BOOL	Acknowledge error
Bclosed	I	BOOL	Checkback input whether Close Brake is activated (singlechannel - PLC)
Axis	I	INT	**Axis number of axis to be tested**
TimerNo	I	TIMER	Timer from user program
TV_BTactiv	I	S5TIME	Monitoring time value -> brake test active, check of axis signal DBX71.0
TV_Bclose	I	S5TIME	Monitoring time value -> close brake Check of input signal Bclosed after output CloseBrake has been set.
TV_FeedCommand	I	S5TIME	Monitoring time value -> issue travel command Check travel command after MoveAxis has been set.
TV_FXSreched	I	S5TIME	Monitoring time value -> fixed stop reached
TV_FXShold	I	S5TIME	Monitoring time value -> test brake
CloseBrake	Q	BOOL	Request, close brake
MoveAxis	Q	BOOL	Request, initiate traversing motion
Done	Q	BOOL	Test successfully completed
Error	Q	BOOL	Error occurred.
State	Q	BYTE	Error status

Error identifiers

State	Meaning
0	No error
1	Start conditions not fulfilled, e.g., axis not under closedloop control/brake closed/axis disabled
2	No NC checkback in "Brake test active" signal on selection of brake test
3	No "Brake applied" checkback by input signal BClosed
4	No travel command output (e.g., axis motion has not been started)
5	Fixed end stop will not be reached -> axis reset was initiated
6	Traversing inhibit/approach too slow -> fixed end stop cannot be reached. TV FXSreached monitoring timeout
7	Brake is not holding at all (end position is reached)/approach velocity too high
8	Brake opens during the holding period
9	Error in brake test deselection
10	Internal error
11	"PLC monitoring axis" signal not activated by the user program

Note

The block must be called by the user program. The user must provide an instance DB with any number for this purpose. The call is multi-instance-capable.

Example of a call for FB 11:

UN	M	111.1;	//Request to close brake, Z axis of FB
=	Q	85.0;	//Brake control, Z axis
OPEN		"Axis3";	//Brake test, Z axis
O	I	73.0;	//Brake test trigger, Z axis
O	M	110.7;	//Brake test is running
FP	M	110.0;	
UN	M	111.4;	//Error has occurred
S	M	110.7;	//Brake test is running
S	M	110.6;	//Next step
S	DBX	8.4;	//Request neutral axis
U	DBX	68.6;	//Checkback signal, axis is neutral
U	M	110.6;	
FP	M	110.1;	
R	M (GND)	110.6;	
S	M	110.5;	//Next step
R	DBX	8.4;	
S	DBX	28.7;	//Request PLC-monitored axis
U	DBX	63.1;	//Checkback signal, axis monitored by PLC
U	M	110.5;	
FP	M	110.2;	
R	M	110.5;	
S	M	111.0;	//Start brake test for FB

```
CALL FB 11, DB 211 (//Brake test block
        Start           :=M     111.0,    //Start brake test
        Ack             :=I     3.7,      //Acknowledge error with RESET key
        Bclosed         :=I     54.0,     //Checkback message close brake
                                          //controlled
        Axis            := 3,             //Axis number of axis to be tested
                                          //Axis Z axis
        TimerNo         := T    110,      //Timer number
        TV_BTactiv      := S5T#200MS,     //Monitoring time value:
                                          //Brake test active DBX71.0
        TV_Bclose       := S5T#1S,        //Monitoring time value:
                                          //Brake closed
        TV_FeedCommand  := S5T#1S,        //Monitoring time value:
                                          //Traversing command output
        TV_FXSreache    := S5T#1S,        //Monitoring time value:
                                          //Fixed stop reached
        TV_FXShold      := S5T#2S,        //Monitoring time value: Test time Brake
        CloseBrake      :=M     111.1,    //Request to close brake
```

```
           MoveAxis             :=M      111.2,    Request, initiate traversing motion
           Done                 :=M      111.3,    //Test successfully completed
           Error                :=M      111.4,    //Error has occurred
           State                := MB    112);     //Error status


OPEN                 "Axis3";   //Brake test, Z axis
O          M         111.3;     //Test successfully completed
O          M         111.4;     //Error has occurred
FP         M         110.3;
R          DBX       28.7;      //Request, PLC-monitored axis
UN         DBX       63.1;      //Checkback signal, axis monitored by PLC
U          M (GND)   111.0;     //Start brake test for FB
U          M (GND)   110.7;     //Brake test is running
FP         M (GND)   110.4;
R          M (GND)   111.0;     //Start brake test for FB
R          M         110.7;     //Brake test is running


CALL "SpinCtrl" (//Traverse Z axis
           Start                :=M      111.2,    //Start traversing motion
           Stop                 := FALSE,
           Funct                := B#16#5,         //Mode: Axis mode
           Mode                 := B#16#1,         //Procedure: Incremental
           AxisNo               := 3,              //Axis number of axis to be traversed
                                                   //Z axis
           Pos                  := -5.000000e+000, //Traversing distance: Minus 5 mm
           FRate                := 1.000000e+003,  //Feedrate: 1000 mm/min
           InPos                :=M      113.0,    //Position reached
           Error                :=M      113.1,    //Error has occurred
           State                := MB    114);     //Error status
```

2.12.10   FB 29: Signal recorder and data trigger diagnostics

Signal recorder

The "diagnostics" FB allows various diagnostic routines to be performed on the PLC user program. A diagnostic routine logs signal states and signal changes. In this diagnostic routine, function number 1 is assigned to the "Func" parameter. Up to 8 signals of the parameters "Signal_1" to "Signal_8" are recorded in a ring buffer each time one of the signals changes. The current information of parameters "Var1" as BYTE value, and "Var2" and "Var3" as INTEGER values are also stored in the ring buffer.

The number of past OB 1 cycles is also stored in the buffer as additional information. This information enables the graphical evaluation of signals and values in OB 1 cycle grid. The first time the "diagnostics" FB is called in a new PLC cycle, the "NewCycle" parameter must be set to TRUE. If the "diagnostics" FB is called several times in the same OB 1 cycle, the "NewCycle" parameter must be set to FALSE for the second and subsequent calls. This prevents a new number of OB 1 cycles from being calculated.

The ring buffer, specified by the user, must have an ARRAY structure specified as in the source code. The array can have any number of elements. A size of 250 elements is recommended. The "ClearBuf" parameter is used to clear the ring buffer and set the BufAddr pointer (I/O parameter) to the start. The instance DB related to the FB is a DB from the user area and is to be transferred to the FB "Diagnostics" with the parameter "BufDB".

Data trigger

The data trigger function is intended to allow triggering on specific values (or bits) at any permissible memory cell. The cell to be triggered is "rounded" with a bit mask ("AndMask" parameter) before the "TestVal" parameter is compared in the diagnostic block.

Note

The source code for the function is available in the source container of the basic-program library under the name Diagnose.awl.. The instance DB and the ring buffer DB are also defined in this source block. The function call is also described in the function. The DB numbers and the call must be modified.

Declaration of the function

```
FUNCTION_BLOCK FB 29
VAR_INPUT
Func                    : INT ;          //Function number: 0 = No function,
                                         //1 = Signal recorder, 2 = Data trigger
     Signal_1           : BOOL ;         //Start of brake test
     Signal_2           : BOOL ;
     Signal_3           : BOOL ;
     Signal_4           : BOOL ;
     Signal_5           : BOOL ;
     Signal_6           : BOOL ;
     Signal_7           : BOOL ;
     Signal_8           : BOOL ;
     NewCycle           : BOOL ;
     Var1               : BYTE ;
     Var2               : INT ;
     VAR                : INT ;
     BufDB              : INT ;
     ClearBuf           : BOOL ;
     DataAdr            : POINTER;       //Area pointer to testing word
     TestVal            : WORD ;         //Value for triggering
     AndMask            : WORD ;         //AND mask to the testing word
END_VAR
VAR_OUTPUT
     TestIsTrue         : BOOL ;
END_VAR
VAR_IN_OUT
     BufAddr            : INT ;
END_VAR
```

Structure for ring buffer

```
TITLE =
                                        //Ring buffer DB for FB 29

VERSION : 1.0


STRUCT
     Field: ARRAY [0 .. 249 ] OF STRUCT      //can be any size of this struct

     Cycle : INT ;                          //Delta cycle to last storage in buffer
     Signal_1 : BOOL ;                      //Signal names same as FB 29
     Signal_2 : BOOL ;
     Signal_3 : BOOL ;
     Signal_4 : BOOL ;
     Signal_5 : BOOL ;
     Signal_6 : BOOL ;
     Signal_7 : BOOL ;
     Signal_8 : BOOL ;
     Var1 : BYTE ;
     Var2 : WORD ;
     Var3 : WORD ;
     END_STRUCT;
END_STRUCT;
BEGIN
END_DATA_BLOCK
```

Description of formal parameters

The table below lists all formal parameters of the Diagnostics function:

Formal parameters of diagnostics function				
Signal	Type	Type	Value range	Remark
Func	I	INT	0, 1, 2	Function 0: Deactivate 1: Signal recorder 2: Data trigger
Parameters for function 1				
Signal_1 to Signal_8	I	BOOL		Bit signals checked for change
NewCycle	I	BOOL		See the "Signal recorder" description above
Var1	I	BYTE		Additional value
Var2	I	INT		Additional value
VAR	I	INT		Additional value
BufDB	I	INT		Ring buffer DB no.
ClearBuf	I	BOOL		Delete ring buffer DB and reset pointer BufAddr
BufAddr	I/O	INT		Target area for read data
Parameters for function 2				
DataAdr	I	POINTER		Pointer to word to be tested
TestVal	I	WORD		Comparison value
AndMask	I	WORD		See description
TestIsTrue	A	BOOL		Result of comparison

Configuration steps

- Select function of diagnostics block.
- Define suitable data for the recording as signal recorder or data triggering.
- Find a suitable point or points in the user program for calling the diagnostics FB.
- Create a data block for the ring buffer, see call example.
- Call the diagnostics FB with parameters in the user program.

In function 1, it is advisable to clear the ring buffer with the "ClearBuf" parameter. When the recording phase with function 1 is completed, read out the ring buffer DB in STEP7 with the function "opening the data block in the data view". The content of the ring buffer DB can now be analyzed.

Call example

```
FUNCTION FC 99: VOID
TITLE =
VERSION : 0.0

BEGIN
NETWORK
TITLE = NETWORK

CALL FB 29, DB 80 (
Func                := 1,
        Signal_1    :=M          100.0,
        Signal_2    :=M          100.1,
        Signal_3    :=M          100.2,
        Signal_4    :=M          100.3,
        Signal_5    :=M          10.4,
        Signal_6    :=M          100.5,
        Signal_7    :=M          100.6,
        Signal_8    :=M          100.7,
        NewCycle    := TRUE,
        Var1        := MB        100,
        BufDB       := 81,
        ClearBuf    :=M          50.0);
END_FUNCTION
```

2.12.11   FC 2: GP_HP Basic program, cyclic section

Description of Functions

The complete processing of the NCKPLC interface is carried out in cyclic mode. In order to minimize the execution time of the basic program, only the control/status signals are transmitted cyclically; transfer of the auxiliary functions and G functions only takes place when requested by the NCK.

Declaration

FUNCTION FC 2: VOID
//No parameters

Call example

As far as the time is concerned, the basic program must be executed **before** the user program. It is, therefore, called first in OB 1.

The following example contains the standard declarations for OB 1 and the calls for the basic program (FC2), the transfer of the MCP signals (FC19), and the acquisition of error and operating messages (FC10).

```
ORGANIZATION_BLOCK OB 1
VAR_TEMP
    OB1_EV_CLASS :        BYTE ;
    OB1_SCAN_1 :          BYTE ;
    OB1_PRIORITY :        BYTE ;
    OB1_OB_NUMBR :        BYTE ;
    OB1_RESERVED_1 :      BYTE ;
    OB1_RESERVED_2 :      BYTE ;
    OB1_PREV_CYCLE :      INT ;
    OB1_MIN_CYCLE :       INT ;
    OB1_MAX_CYCLE :       INT ;
    OB1_DATE_TIME :       DATE_AND_TIME;
END_VAR
BEGIN
CALL FC 2;                          //Call basic program as first FC
//INSERT USER PROGRAM HERE
CALL FC 19 (                        //MCP signals to interface
BAGNo :=              B#16#1,       //Mode group no. 1
ChanNo :=            B#16#1,        //Channel no. 1
SpindleIFNo :=       B#16#4,        //Spindle interface number = 4
FeedHold :=          m22.0,         //Feed stop signal
                                    //modal
SpindleHold :=  db2.dbx151.0);      //Spindle stop modal
                                    //in message DB
CALL FC 10 (                        //Error and operational messages
             ToUserIF :=  TRUE,     //Signals transferred from DB2
                                    //to interface
             Ack :=       I6.1);    //Acknowledgment of error messages
                                    //via I 6.1
END_ORGANIZATION_BLOCK
```

2.12.12   FC 3: GP_PRAL Basic program, interruptcontrolled section

Description of Functions

Blocksynchronized transfers from the NCK to the PLC (auxiliary and G functions) are carried out in the interruptdriven part of the basic program. **Auxiliary functions** are subdivided into normal and highspeed auxiliary functions.

Highspeed auxiliary functions

- The highspeed functions of an NC block are buffered and the transfer acknowledged to the NC. These are passed to the application interface at the start of the next OB1 cycle.

- Highspeed auxiliary functions programmed immediately one after the other, are not lost for the user program. This is ensured by a mechanism in the basic program.

Normal auxiliary functions
are acknowledged to the NC only when one cycle duration has passed on it. This allows the application to issue a read disable to the NC.

The **G Functions** are evaluated immediately and passed to the application interface.

NC process interrupts

If the interrupt is triggered by the NC (possible in each IPO cycle), a bit in the local data of OB 40 ("GP_IRFromNCK") is set by the basic program only when the FB 1 parameter "UserIR": is = TRUE. This data is not set on other events (process interrupts through I/Os). This information makes it possible to branch into the associated interrupt routine in the user program in order to initiate the necessary action.

To be able to implement highspeed, jobcontrolled processing of the user program for the machine, the following NC functions are available in the interrupt processing routine (OB 40 program section) for the PLC user program with SW 3.2 and higher:

- Selected **auxiliary functions**

- **Tool-change function** for tool-management option

- **Position reached** for positioning axes, indexing axes and spindles with activation via PLC

- Block transfer to FM (function available soon)

The functions listed above can or must be evaluated by the user program in OB 40 in order to initiate reactions on the machine. As an example, the revolver switching mechanism can be activated when a T command is programmed on a turning machine.

For further details on programming hardware interrupts (time delay, interruptibility, etc.) refer to the corresponding SIMATIC documentation.

Auxiliary functions

Generally, high-speed or acknowledging auxiliary functions are processed with or without interrupt control independently of any assignment.
Basic-program parameters in FB 1 can be set to define which auxiliary functions (T, H, DL) must be processed solely on an interruptdriven basis by the user program.
Functions which are not assigned via interrupts are only made available by the cyclic basic program as in earlier versions. The change signals of these functions are available in a PLC cycle.
Even if the selection for the auxiliary function groups (T, H, DL) is made using interrupt control, only one interrupt can be processed by the user program for the selected functions.
A bit is set channelspecifically in the local data "GP_AuxFunction" for the user program (if "GP_AuxFunction[1]" is set, then an auxiliary function is available for the 1st channel).
The change signals and function value are available to the user in the associated channel DB. The change signal for this interruptdriven function is reset to zero in the cyclic basic program section after the execution of at least one full OB1 cycle (max. approx. two OB1 cycles).

Tool change

With the tool-management option, the tool-change command for revolver and the tool change in the spindle is supported by an interrupt. The local data bit "GP_TM" in OB 40 is set for this purpose. The PLC user program can thus check the tool management DB (DB 72 or DB 73) for the tool change function and initiate the tool change operation.

Position reached

In the bit structure, "GP_InPosition" of the local data of OB 40 is specific to the machine axis (each bit corresponds to an axis/spindle, e.g., GP_InPosition[5] corresponds to axis 5).
If a function has been activated via FC 15 (positioning axis), FC 16 (indexing axis) or FC 18 (spindle control) for an axis or spindle, the associated "GP_InPosition" bit makes it possible to implement instantaneous evaluation of the "InPos" signal of the FCs listed above. This feature can be used, for example, to obtain immediate activation of clamps for an indexing axis.

Declaration

FUNCTION FC 3 : VOID
//No parameters

Call example

As far as the time is concerned, the basic program must be executed **before** other interrupt-driven user programs. It is, therefore, called first in OB 40.

The following example contains the standard declarations for OB 40 and the call for the basic program.

```
ORGANIZATION_BLOCK OB 40
VAR_TEMP
     OB40_EV_CLASS :              BYTE ;
     OB40_STRT_INF :              BYTE ;
     OB40_PRIORITY :              BYTE ;
     OB40_OB_NUMBR :              BYTE ;
     OB40_RESERVED_1 :            BYTE ;
     OB40_MDL_ID :                BYTE ;
     OB40_MDL_ADDR :              INT ;
     OB40_POINT_ADDR :            DWORD;
     OB40_DATE_TIME :             DATE_AND_TIME;


//Assigned to basic program
GP_IRFromNCK : BOOL ;                     //Interrupt by NCK for user

GP_TM : BOOL ;                            //Tool management

GP_InPosition : ARRAY [1..3] OF BOOL;     //Axis-oriented for positioning,
                                          //indexing axes, spindles

GP_AuxFunction : ARRAY [1..10] OF BOOL;   //Channel-oriented for auxiliary functions

GP_FMBlock : ARRAY [1..10] OF BOOL;       //Channel-oriented for block transfer for FM
                                          // (in preparation)


//Further local user data may be defined from this point onwards
END_VAR
BEGIN
     CALL FC 3;
     //INSERT USER PROGRAM HERE
END_ORGANIZATION_BLOCK
```

2.12.13   FC 7: TM_REV Transfer block for tool change with revolver

Description of Functions

After a revolver has been changed, the user will call this block FC TM_REV. The revolver number (corresponding to interface number in DB 73) must be specified in parameter "ChgdRevNo" for this purpose. As this block is called, the associated "Interface active" bit in data block DB 73, word 0 of FC 7 is reset after parameter "Ready" := TRUE is returned.

Block FC TM_REV may be started (with "Start" parameter = "TRUE") only if an activation signal for the appropriate interface (DB 73, word 0) for this transfer has been supplied by the tool management function.

Output parameter "Ready" is set to the value TRUE when the job has been executed correctly.

The user must then set the
"Start" parameter to FALSE
or not call the block again.

If the
"Ready" parameter is set to FALSE,
the error code in the
"Error" parameter must be interpreted.

If the error code = 0, then this job must be repeated in the next PLC cycle (e.g., "Start" remains set to "TRUE"). This means that the transfer job has not yet been completed (see example FC 7 call and timing diagram).
The "Start" parameter does not need a signal edge for a subsequent job.

⚠ **Warning**

It is not permissible to abort the transfer (e.g., by an external signal RESET). The "Start" parameter must always retain the 1 signal until the "Ready" and/or "Error" parameters <> 0.

An error code <> 0 indicates incorrect parameterization.

Note

For further details on tool management (also with regard to PLC) refer to the Description of Functions Tool Management. In addition, PI services for tool management via FB 4, FC 8 and FC 22 are available.

Manual revolver switching

If a manual action is used to rotate the revolver, this information must be forwarded to the tool management. The asynchronous transfer function of FC 8 must be used to transfer the modified positions of the revolver. This must only occur once on the first manual rotation in the sequence. In this case, the following parameterization of the asynchronous transfer is needed via FC 8:

TaskIdent = 4

TaskIdentNo = channel

NewToolMag = Magazine number of the revolver

NewToolLoc = Original location of the tool

OldToolMag = Magazine no. buffer storage (spindle) = 9998

OldToolLoc = Buffer storage number of the spindle

Status = 1

This action also causes the same T command to be resent to the tool-management interface if the previous T is programmed again.

Declaration of the function

STL representation

```
FUNCTION FC 7 :          VOID
//NAME :TM_REV
VAR_INPUT
    Start:               BOOL ;
    ChgdRevNo:           BYTE ;
END_VAR
VAR_OUTPUT
    Ready:               BOOL ;
    Error :              INT ;
END_VAR
BEGIN
END_FUNCTION
```

Description of formal parameters

The table below lists all formal parameters of the TM_REV function.

Signal	Type	Type	Value range	Remark
Start	I	BOOL		1 = Start of transfer
ChgdRevNo	I	BYTE	1..	Number of revolver interface
Ready	A	BOOL		1 = Transfer complete
Error	A	INT	0..3	Error checkback
				0 : No error has occurred
				1: No revolver present
				2: Illegal revolver number in parameter "ChgdRevNo"
				3: Illegal job ("interface active" signal for selected revolver = "FALSE")

Pulse diagram
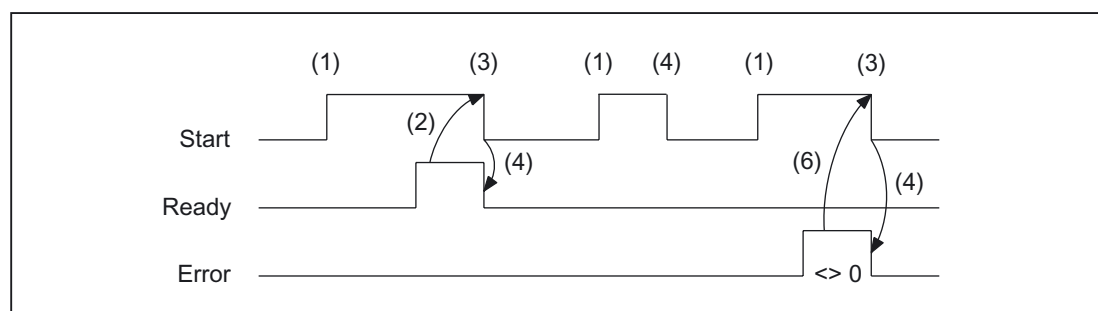
(1) Activation of function by means of a positive edge

(2) Positive acknowledgment: Tool management has been transferred

(3) Reset function activation after receipt of acknowledgment

(4) Signal change using FC

(5) This signal chart is not permissible. The job must generally be terminated since the new tool positions must be conveyed to the tool management in the NCK.

(6) Negative acknowledgment: Error has occurred, error code in the output parameter "Error".

Call example

`CALL FC 7 (`	//Tool management transfer of block for revolver
`Start :=`	m 20.5, //Start := "1 " => transfer trigger
`ChgdRevNo :=`	DB61.DBB 1,
`Ready :=`	m 20.6,
`Error :=`	DB61.DBW 12);
`u m 20.6;`	//Poll ready
`r m 20.5;`	//Reset start
`spb m001;`	//Jumps, if everything OK
`l db61.dbw 12;`	//Error information
`ow w#16#0;`	//Evaluate error
`JC error;`	//Jumps to troubleshooting, if <> 0
`m001:`	// Start of another program
`error:`	
`r m 20.5;`	//Reset start, if an error has occurred

2.12.14　FC 8: TM_TRANS transfer block for tool management

Description of Functions

The user calls this block FC TM-TRANS when the position of the tool or the status of the transfer operation changes. The parameter "TaskIdent" specifies the transfer job for the block FC 8 at the tool management interface:

1. For loading/unloading positions,

2. For spindle change positions,

3. For revolver change positions as transfer identifier

4. Asynchronous transfer,

5. Asynchronous transfer with location reservation

The interface number is indicated in parameter "TaskIdentNo".

Example for loading point 5:

Parameter "TaskIdent":= 1 and "TaskIdentNo":= 5.

Furthermore, the **current** tool positions and status data (list of "Status" parameter in the following text) are also transmitted for this transfer function.

Note

FC8 informs the NCK of the current positions of the old tool.

The NCK knows where the old and the new tool have been located until the position change.

In the case of a transfer without a socalled "old tool" (e.g., on loading), the value 0 is assigned to parameters "OldToolMag", "OldToolLoc".

Block FC TM_TRANS may be started only with "Start" parameter = "TRUE" if an activation signal for the appropriate interface (DB 71, DB 72, DB 73 in word 0) for this transfer has been supplied by the tool management function.

Output parameter "Ready" is set to the value TRUE when the job has been executed correctly.

The user must then set the
"Start" parameter to FALSE
or not call the block again.

If the
"Ready" parameter = FALSE,
the error code in the
"Error" parameter must be interpreted (see Call example FC 8 and timing diagram).

If the error code = 0, then this job must be repeated in the next PLC cycle (e.g., "Start" remains set to "TRUE"). This means that the transfer operation has not yet been completed.

If the user assigns a value of less than 100 to the **Parameter "Status"**, then the associated interface in data block DB 71 or DB 72 or DB 73, word 0 is deactivated (process completed). The appropriate bit for the interface is set to 0 by FC 8.

The "Start" parameter does not need a signal edge for a subsequent job. This means that new parameters can be assigned with "Start = TRUE" immediately when "Ready = TRUE" is received.

Asynchronous transfer

To ensure that changes in the position of a tool are automatically signaled from PLC to tool management (e.g., power failure during an active command or independent changes in the position by the PLC), block FC 8 TM_TRANS with "TaskIdent": = 4 or 5 is called. This call does not require interface activation by tool management.

If parameter "TaskIdent" = 5
the tool management reserves the location in addition to changing the position. The location is only reserved if the tool has been transported from a real magazine to a buffer storage.

A relevant NC channel must be parameterized in the "TaskIdentNo" parameter.

The previous position of the tool is specified in parameters "OldToolMag", "OldToolLoc"
; the current position of the tool is specified
in parameters "NewToolMag", "NewToolLoc". Status = 1 must be specified.

With status 5, the specified tool remains at location "OldToolMag", "OldToolLoc". This location must be a buffer (e.g., spindle). The real magazine and location must be specified in the parameters "NewToolMag", "NewToolLoc"; the location is at the position of the buffer. This procedure must always be used if the tool management is to be informed of the position of a specific magazine location. This procedure is used for alignment in search strategies.

Note

It is not permissible to abort the transfer (e.g., by an external signal RESET). The "Start" parameter must always retain the 1 signal until the "Ready" and/or "Error" parameters <> 0.

An error code <> 0 indicates incorrect parameterization.

Note

For further details on tool management (also with regard to PLC) refer to the Description of Functions Tool Management. In addition, PI services for tool management via FB 4, FC 7 and FC 22 are available.

Declaration of the function

STL representation

```
FUNCTION FC 8 : VOID
//NAME :TM_TRANS
VAR_INPUT
     Start:                  BOOL ;
     TaskIdent:              BYTE ;
     TaskIdentNo:            BYTE ;
     NewToolMag:             INT ;
     NewToolLoc:             INT ;
     OldToolMag:             INT ;
     OldToolLoc:             INT ;
     Status:                 INT ;
END_VAR
VAR_OUTPUT
     Ready:                  BOOL ;
     Error :                 INT ;
END_VAR
BEGIN
END_FUNCTION
```

Description of formal parameters

The table below lists all formal parameters of the TM_TRANS function:

Signal	Type	Type	Value range	Remark
Start	I	BOOL		1 = Start of transfer
TaskIdent	I	BYTE	1..5	Interface or tank identifier 1: Loading/unloading location 2: Spindle change position 3: Revolver change position 4: Asynchronous transfer 5: Asynchronous transfer with location reservation
TaskIdentNo	I	BYTE	1..	Number of associated interface or channel number. The upper nibble can specify the interface number for asynchronous transfer (e.g., B#16#12, 1st interface, 2nd channel).
NewToolMag	I	INT	1, 0..	Current magazine number of new tool -1: Tool remains at its location NewToolLoc = any value. Only permissible if TaskIdent = 2
NewToolLoc	I	INT	0 to max. location number	Current location number of new tool

Signal	Type	Type	Value range	Remark
OldToolMag	I	INT	-1, 0..	Current magazine number of new tool -1: Tool remains at its location OldToolLoc = any value. Only permissible if TaskIdent = 2
OldToolLoc	I	INT	Max. location number	Current location number of tool to be replaced
Status	I	INT	1...7, 103...105	Status information about transfer operation
Ready	A	BOOL		1= transfer completed
Error	A	INT	0..65535	Error checkback 0: No error has occurred 1: Unknown "TaskIdent" 2: Unknown "TaskIdentNo" 3: Illegal task ("signal "Interface (SS) active" of selected revolver = "FALSE") Other values: The number corresponds to the error message of the tool management function in the NCK caused by this transfer.

Pulse diagram
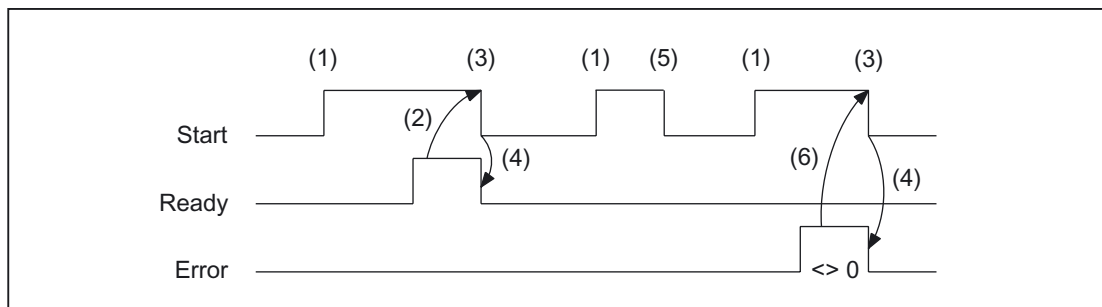
(1)   Activation of function by means of a positive edge

(2)   Positive acknowledgment: Tool management has been transferred

(3)   Reset function activation after receipt of acknowledgment

(4)   Signal change using FC

(5)   This signal chart is not permissible. The job must generally be terminated since the new tool positions must be conveyed to the tool management in the NCK.

(6)   Negative acknowledgment: Error has occurred, error code in the output parameter "Error".

Status list

Status = 1:
The WZV operation is completed (loading/unloading/reloading, prepare change, change).

The parameters "NewToolMag", "NewToolLoc", "OldToolMag", "OldToolLoc" of the FC 8 block are to be parameterized to the actual positions of the tools involved. Except in the case of preparing change these are normally the specified target position of the tools of the associated WZV interface, see also "Explanations of the formal parameters".

1. In the case of loading/unloading/reloading, the tool has arrived at the required target address. If the bit in the interface in DB 71.DBX (n+0).3 "position at loading point" is enabled, status 1 cannot be used for the function termination. Status 5 must be used for correct termination.

2. In the case of "Prepare change", the new tool is now available. The tool may, for example, be positioned in a buffer (gripper). In some cases, the target (magazine, location) of the old tool has been moved to the toolchange position after placement of the new tool in a buffer. However, the old tool still remains in the spindle. The preparations for a tool change are thus complete. After this acknowledgment, the "Change" command can be received. The positions in parameters "NewToolMag", "NewToolLoc", "OldToolMag" and "OldToolLoc" correspond to the current tool positions.

3. In the case of "Change" (spindle or revolver), the tools addressed in the interface have now reached the required target addresses.
The tool-change operation is thus completed.

Status = 2:
The "new" tool cannot be made available.

This status is only admissible in conjunction with the "Prepare Change" command. When this status is applied, the PLC must be prevented from making a change with the proposed tool. The proposed (new) tool is disabled by the tool management function in the NCK. A new command is then output by the tool management with a duplo tool. The positions in parameters "NewToolMag", "NewToolLoc", "OldToolMag", and "OldToolLoc" correspond to the original tool positions.

Status = 3:
An error has occurred.

The tool positions must not have been changed. Any changes to the magazine positions which have taken place in the meantime must be notified beforehand, for example, with status = 105 via FC 8 transfer block Only then will the tool positions be taken into account by the tool management function.

Status = 4:
It would be better to position the "old" tool in the magazine position specified in parameters "OldToolMag" and "OldToolLoc".

This status is permissible only in conjunction with preparation for tool change (change into spindle). After this status has been transferred to the tool management in the NCK, the tool management tries to consider the specified magazine position in the next command. But this is done only when this position is free. Parameters "NewToolMag" and "NewToolLoc" are not taken into account.

Status = 5:
The operation is complete.

The "new" tool is **in** the position specified in parameters "NewToolMag", "NewToolLoc". In this case, the specified tool is not really in this position, but is still in the same magazine location. However, this magazine location has been moved to the position set in the parameters (e.g., tool change position). This status may be used only for revolvers, chain-type magazines and disk magazines. Status 5 enables the tool management function to adjust the current position of a magazine and to improve the search strategy for subsequent commands. This status is permissible only in conjunction with loading, unloading, and reloading operations and with preparations for a tool change.

The "OldToolMag" and "OldToolLoc" parameters must be parameterized with the data of a buffer.

- **Loading, reloading**:

  On loading or reloading, a location for the tool is already reserved in the NCK. The machine operator must then insert the tool at the target location. Note: The location reservation is canceled when the control system is switched on again.

- **Tool-change preparation**:

  Tool motions still to be executed are not carried out until after the tool has been changed.

- **Positioning to load point:**

  If the bit in the interface in DB 71.DBX (n+0).3 "positioning to load point" is enabled, only status 5 (not status 1) may be used for terminating the function.

Status = 6:
The WZV job has been completed.

This status has the same function as status 1, but, in addition, a reservation of the source location is carried out. This status is only permitted when reloading. The command is ended and the source location of the tool is reserved if the target location is in a buffer magazine.

Status = 7:
Initiate repetition of the command "Prepare Tool".

This status is only admissible in conjunction with the "Change tool" command. This status is intended for use when the "new" tool has changed its position (e.g., via an asynchronous command of the "new" tool). After "Ready = 1" has been provided by FC 8, the "Prepare Change" command is repeated automatically with the same tool. For the automatic repetition, a new tool search is carried out. The positions in parameters "NewToolMag", "NewToolLoc", "OldToolMag", and "OldToolLoc" correspond to the original tool positions.

Status = 103:
The "new" tool can be inserted.

This status is permitted only in the tool change preparation, when the PLC may reject the new tool (e.g. in case of MD20310 $MC_TOOL_MANAGEMENT_MASK, bit 4=1 for the possibility, request changed parameter from PLC once again). The tool positions have remained unchanged. This status is thus necessary, when the processing is to be continued in the NCK without an unnecessary stop.

References:
/FBW/, Description of Functions, Tool Management.

Status = 104:
The "new" tool is in the position specified in parameters "NewToolMag", "NewToolLoc".

This status is only permissible if the tool is still in the magazine in the same location. The "old" tool is in the position (buffer) specified in parameters "OldToolMag", "OldToolLoc". In this case, however, the new tool is not really in this position, but is still in the same magazine location. However, this magazine location has been moved to the position set in the parameters (e.g., tool change position). This status may be used only in conjunction with revolvers, chaintype magazines and disk magazines for the "Tool change preparation" phase. Status 104 enables the tool management to adjust the current position of a magazine and to improve the search strategy for subsequent commands.

Status = 105:
The specified buffer has been reached by all tools involved
 (standard case if the operation has not yet been completed).

The tools are in the specified tool positions (parameters "NewToolMag", "NewToolLoc", "OldToolMag", "OldToolLoc").

Status definition

A general rule for the acknowledgment status is that the state information 1 to 7 leads to the termination of the command. If FC 8 receives one of the statuses, the "Interface active bit" of the interface specified in FC 8 is reset to "0" (see also interface lists DB 71 to DB 73), thus completing the operation. The response if different in the case of status information 103 to 105. When the FC 8 receives one of these items of status information, the "Interface active bit" of this interface remains at "1". Further processing is required by the user program in the PLC (e.g., continuation of magazine positioning). This item of status information is generally used to transfer changes in position of one or both tools while the operation is still in progress.

Call example

```
CALL FC 8 (                        //Tool-management transfer block
      Start :=         m 20.5,      //Start := "1 " => transfer trigger
      TaskIdent :=     DB61.DBB 0,
      TaskIdentNo :=   DB61.DBB 1,
      NewToolMag :=    DB61.DBW 2,  //Current position of new tool
      NewToolLoc :=    DB61.DBW 4,
      OldToolMag :=    DB61.DBW 6,  //Current position of old tool
      OldToolLoc :=    DB61.DBW 8,
      Status :=        DB61.DBW 10, //Status
      Ready :=         m 20.6,
      Error :=         DB61.DBW 12);
u m 20.6;                           //Poll ready
r m 20.5;                           //Reset start
spb m001;                           //Jumps, if everything OK
l DB61.dbw 12;                      //Error information
ow w#16#0;                          //Evaluate error
JC error;                           //Jumps to troubleshooting


m001:                               //Normal branch


error:                              //Troubleshooting
r m 20.5:                           //Reset start
```

2.12.15   FC 9: ASUB startup of asynchronous subprograms

Description of functions

The FC ASUB can be used to trigger any functions in the NC. Before an ASUB can be started from the PLC, it must have been selected and parameterized by an NC program or by FB 4 (PI service ASUB). The channel and interrupt number must tally with the parameters in FC 9.

An ASUB prepared in this way can be started by the PLC at any point in time. The NC program running on the channel in question is interrupted by the asynchronous subprogram. Only one ASUB can be started in the same channel at a time. If the start parameter is set to logical 1 for two FC 9s within **one** PLC cycle, the ASUBs are started in the sequence in which they are called.

The start parameter must be set to logic 0 by the user once the ASUB has been terminated (Done) or if an error has occurred.

For the purpose of job processing, every FC ASUB requires its own WORD parameter "Ref" from the global user memory area. This parameter is for internal use only and must not be changed by the user. The parameter **Ref** is initialized with the value 0 in the first OB 1 cycle and, for this reason, **every FC 9 must be called absolutely**. Alternatively, the user can initialize parameter "Ref" with a value of 0 during startup. This option makes conditional calls possible. When a conditional call is activated by parameter "Start" = 1, it must remain active until parameter "Done" has made the transition from 1 to 0.

Note

The FB 4 call must be terminated before the FC 9 can be started. FC 9 cannot be started if "Emergency off" is set. Neither can FC 9 be started if the channel reset is active.

Declaration of the function

```
FUNCTION FC 9 : VOID
//NAME :ASUP
VAR_INPUT
      Start:        BOOL ;
      ChanNo:    INT ;
      IntNo:        INT ;
END_VAR
VAR_OUTPUT
      Active:       BOOL ;
      Done :        BOOL ;
      Error :        BOOL ;
      StartErr:     BOOL ;
END_VAR
VAR_IN_OUT
      Ref:          WORD ;
END_VAR
```

Description of formal parameters

The table below lists all formal parameters of the ASUB function.

Signal	Type	Type	Value range	Remark
Start	I	BOOL		
ChanNo	I	INT	1 - 10	No. of the NC channel
IntNo	I	INT	1 - 8	Interrupt No.
Active	A	BOOL		1 = Active
Done	A	BOOL		1 = ASUB completed
Error	A	BOOL		1 = Interrupt switched off
StartErr	A	BOOL		1 = Interrupt number not assigned or deleted
Ref	I/O	WORD	Global variable (MW, DBW,..)	1 word per FC 9 (for internal use)

Pulse diagram
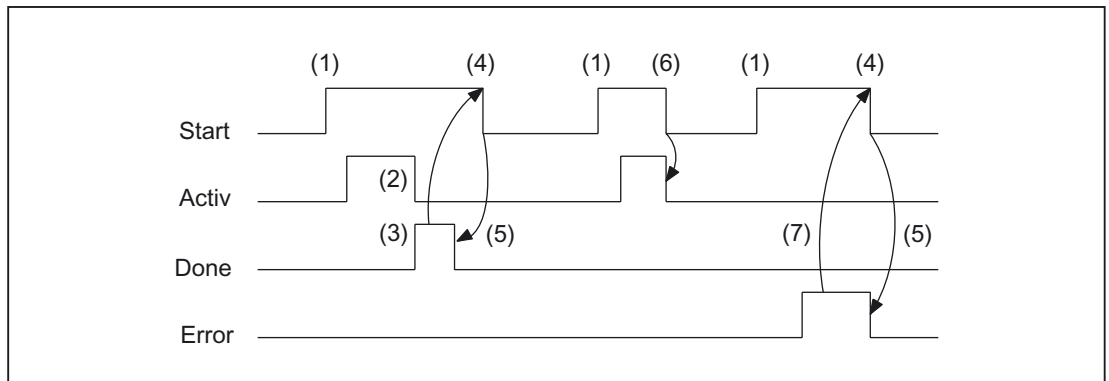
(1)	Activation of function
(2)	ASUP active
(3)	Positive acknowledgment: ASUB ended
(4)	Reset function activation after receipt of acknowledgment
(5)	Signal change using FC
(6)	Not permitted If function activation is reset prior to receipt of acknowledgment, the output signals are not updated without the operational sequence of the activated function being affected.
(7)	Negative acknowledgment: Error has occurred

Call example

```
CALL FC 9 (          //Start an asynchronous subprogram
                     //in channel 1 interrupt number 1
          Start :=        I 45.7,
          ChanNo :=       1,
          IntNo :=        1,
          Active :=       M 204.0,
          Done :=         M204.1,
          Error :=        M 204.4,
          StartErr :=     M 204.5,
          Ref :=          MW 200);
```

2.12.16   FC 10: AL_MSG error and operating messages

Description of Functions

FC AL_MSG evaluates the signals entered in DB 2 and displays them as incoming and outgoing error and operational messages on the MMC.

The incoming signals (positive edge)
are displayed immediately in the case of both error and operating messages.

Outgoing signals (negative edge)
are only canceled immediately in the case of operating messages. In case of error messages, the messages not needed any more are deleted only with the parameter "Quit" i.e. error messages remain stored on the MMC - even if the signals no longer exist - until the "acknowledge" parameter is issued, i.e., until the user acknowledges the messages.

The "ToUserIF" parameter can be used to transfer the group signals for the feed, read and NC start disabling signals and feed stop signal to the existing axis, spindle and channel interfaces. The group signals are transferred to the user interface directly from the status information in DB 2 irrespective of an alarm acknowledgment.

1. If parameter "ToUserIF": is set to FALSE, signals are not transferred to the user interface. In this case, the user must take measures in his PLC program to ensure that these signals are influenced in the interface.

2. If parameter "ToUserIF": is set to TRUE, all signals listed above are sent to the user interface as a group signal in each case. The user PLC program can therefore influence these signals only via DB 2 in connection with a message or alarm output. The appropriate information is overwritten in the user interface.

As an alternative to the procedure described under paragraph 2, the user can influence the disable and stop signals without a message output by applying a disable or stop signal state to the interface signals after FC AL_MSG has been called.

The following program illustrates this method:

```
CALL FC 10 (
       ToUserIF :=    TRUE,
       Ack :=         I 6.1);


u m 50.0;                          //Feed disable for channel 1
to DB 21;
s dbx 6.0;                         //Set disable condition, reset via
                                   //FC AL_MSG, if M 50.0 outputs the signal "0".

```

The error and operational messages are stored by the user in the data block DB 2, so that a message can also be displayed at MMC.

Note

In DB2, a "1" signal must be present for several OB1 cycles to ensure that a message can also be displayed on the operator interface.

Declaration of the function

STL representation

```
FUNCTION FC 10:        VOID
     // NAME:          AL_MSG
VAR_INPUT
     ToUserIF :        BOOL ;
     Ack :             BOOL ;
END_VAR
END_FUNCTION
```

Description of formal parameters

The table below lists all formal parameters of the AL-MSG function.

Signal	I/O	Type	Value range	Remark
ToUserIF	I	BOOL		1 = Transfer signals to user interface every cycle
Ack	I	BOOL		1 = Acknowledge error messages.

Call example

```
CALL FC 10 (                         //Error and operational messages
     ToUserIF :=      TRUE,          //Signals from DB 2 are transferred to
                                     //interface
     Ack :=           I6.1);         //Acknowledging the error message is done via
                                     //Input E6.1
```

2.12.17   FC 12: AUXFU call interface for user with auxiliary functions

Description of functions

FC AUXFU is generally called on an eventdriven basis in the basic program if the channel transferred in the input parameter contains new auxiliary functions. The PLC user can extend FC AUXFU with program instructions for processing his auxiliary function to avoid cyclic polling of the channel DBs. This mechanism permits auxiliary functions to be processed on a jobdriven basis. FC AUXFU is supplied as a compiled empty block in the basic program. In this case, the basic program supplies parameter "Chan" with the channel number. The PLC user knows which channel has new auxiliary functions available. The new auxiliary functions can be determined by the auxiliary-function change signals in the channel concerned.

Declaration of the function

```
FUNCTION FC 12: VOID            //Event control of auxiliary functions
VAR_INPUT
    Chan:       BYTE ;
END_VAR
BEGIN
    BE;
END_FUNCTION
```

Explanation of formal parameters

The table below lists all formal parameters of the AUXFU function:

Signal	Type	Type	Value range	Remark
Chan	I	BYTE	0 to 9	No. of NC channel -1

Example

```
FUNCTION FC 12: VOID                       //Event control of auxiliary functions
VAR_INPUT
    Chan:                   BYTE ;         //Parameter is supplied by basic program
END_VAR
VAR_TEMP
    ChanDB:                 INT ;
END_VAR

BEGIN
L Chan;                                    //Channel index no., (0,1,2,..)
+ 21;                                      //Channel DB offset
T ChanDB;                                  //Save channel DB no.
TO DB[ChanDB];                             //Channel DB is opened indirectly
```

```
// Auxiliary-function change signals are now scanned, etc.
     BE;

END_FUNCTION
```

2.12.18   FC 13: BHGDisp display control for handheld unit

Description of Functions

This module carries out the display control of the handheld unit (HHU). The information to be output on the display is stored as 32 characters in string data ChrArray. A fixed text assignment of 32 characters is therefore required for this string when the data block is created. Variable components within this string can be inserted by means of the optional numerical converter, For the numeric converter the parameter "Convert" must be set to TRUE. The variable to be displayed is referenced via the pointer Addr. The Parameter "DataType" contains the format description of this parameter (see parameter table). The number of bytes of the variable is linked to the format description. The address justified to the right within the string is specified by parameter "StringAddr". The number of written characters is shown in the parameter table. By setting parameter Row to 0, it is possible to suppress the display (e.g., if several variables in one or several PLC cycles need to be entered in the string without any display output).

Signals

Byte 1 is supplied by the output signals of the HHU and the character specifications are supplied by the module. These may not be written by the PLC user program.

Additional parameters

The pointer parameters for the input and output data of the handheld unit must be parameterized in the start OB 100 in FB 1, DB 7. Parameter BHGIn corresponds to the input data of the PLC from the handheld unit (data received by PLC). Parameter BHGOut corresponds to the output data of the PLC to the handheld unit (data transmitted by PLC). These two pointers must be set to the starting point of the relevant data area, which is also parameterized in SDB210 with an MPI link.

Note

If the numerical converter is used to display information, then it is better to avoid performing a conversion in every PLC cycle for the sake of reducing the PLC cycle time. In this case, it is advisable to use the input signal from the HHU to the PLC "Acknowledgment digital display" (DB m+5.7) for parameter "Convert".

In this way it can be ensured that the most recent numerical information is displayed.

Declaration of the function

STL representation

```
DATA_BLOCK "strdat"                      //The data block number is defined in the symbol file
    STRUCT
    disp:          STRING [32]:= 'line 1 line 2 ';          //32 characters are defined
    END_STRUCT;
BEGIN
END_DATA_BLOCK


FUNCTION FC 13: VOID
VAR INPUT
    Row :          BYTE ;          //Display line (see table)
    ChrArray :     STRING ;        //Transfer at least string[32]
    Convert :      BOOL ;          //Activate numerical conversion
    Addr:          POINTER;        //Points to the variable being converted
    DataType :     BYTE ;          //Data type of the variables
    StringAddr :   INT ;           //Right-justified string address (1 to 32)
    Digits :       BYTE ;          //Number of decimal places (1 to 3)
END VAR
VAR OUTPUT
    Error :        BOOL ;          //Conversion or string error
END VAR
```

Description of formal parameters

The table below lists all formal parameters of the BHGDisp function:

Signal	I/O	Type	Value range	Remark
Row	I	BYTE	0-3	Display line 0: No display output 1: Line 1 2: Line 2 3: Line 1 and line 2
ChrArray	I	STRING	>= string[32]	This string contains the entire display content
Convert	I	BOOL		Activation of numerical conversion
Addr	I	POINTER		Points to the variable to be converted
DataType	I	BYTE	1-8	Data type of variable 1: BOOL, 1 character 2: BYTE, 3 characters 3: CHAR, 1 character 4: WORD, 5 characters 5: INT, 6 characters 6: DWORD, 7 characters 7: DINT, 8 characters 8: REAL, 9 characters (see parameter Digits)
StringAddr	I	INT	1-32	Address within variable ChrArray
Digits	I	BYTE	1-4	Relevant only for data type REAL with sign 1: 6.1 digits unsigned 2: 5.2 digits unsigned 3: 4.3 digits unsigned 4: 3,4 digits without sign indicate number of digits without the sign
Error	A	BOOL		Conversion error, numerical overflow or error in StringAddr

Ranges of values

Value ranges of data types	
Data type	Representable numerical range
BOOL	0, 1
BYTE	0 to 255
WORD	0 to 65535
INT	- 32768 to + 32767
DWORD	0 to 9999999
DINT	- 9999999 to + 9999999
REAL (Digits := 1)	- 999999.9 to + 999999.9
REAL (Digits := 2)	- 99999.99 to + 99999.99
REAL (Digits := 3)	- 9999.999 to + 9999.999
REAL (Digits := 4)	- 999.9999 to + 999.9999

Call example

```
CALL FC 13 (
        Row :=              MB 26,
        ChrArray :=         "strdat".disp,    //DB with name strdat in the symbol table,
                                              //data element disp is declared as String[32]
                                              //and completely assigned with sign
        Convert :=          M 90.1,
        Addr :=             P#M 20.0,         //Number to be converted
        DataType :=         MB 28,            //Data type of the variables
        StringAddr :=       MW 30,
        Digits :=           B#16#3,           //3 decimal places
        Error :=            M 90.2);
```

2.12.19 FC 15: POS_AX positioning of linear and rotary axes

Description of Functions

(Do not use for new applications, function is integrated in FC 18 in SW 3.6 and higher.)

FC POS_AX can be used to traverse axes in any operating mode, also from the PLC.
References:
/FB2/ Function Manual, Extended Functions; Positioning Axes (P2)

In order to traverse the NC axes via the PLC, the traversing check must be activated for the PLC. This can be achieved, for example, by calling FC "POS_AX" with activation of the "Start" parameter. FC "POS_AX" then requests an axis check by the NC. The NC feeds back the status of this axis in byte 68 in the associated axis interface DB 31, ... see lists/ (Book 2), Interface signals power line.

Once the axis check has been completed ("InPos" is True, "Start" changes to zero), it is switched to a neutral state by FC POS AX. Alternatively, the PLC user program can also request the check for the PLC prior to calling FC "POS_AX".

By calling this function several times in succession, a better response reaction by the axes can be obtained as the changeover process in the FC can be omitted.

Activation through the PLC user program is executed in the corresponding axis interface in byte 8.

After return of the check, the axis can again be programmed by the NC program.

Note

Rotary axes can be positioned by the shortest possible route through the programming of a negative feed value in absolute programming mode. In incremental mode (parameter "IC" := TRUE), the traversing direction can be defined by means of the parameter "Pos" sign:
Positive sign indicates travel in a positive direction
Negative sign indicates travel in a negative direction

After the FC has been called, ACCU1 contains an error statement by the NCK (but not if the output parameters are assigned to a data block). This is generally the value 0 (signifies: No error has occurred). The interpretation of other numerical values is shown in the following table.

FC 15 must be called cyclically until the "Active" signal produces an edge transition from 1 to 0. Only when the "Active" signal has a 0 state can the axis concerned be restarted (the next start must be delayed by at least one PLC cycle). This also applies when the assignment in data byte 8 has changed.

The function cannot be aborted by means of parameter "Start", but only by means of the axial interface signals (e.g., delete distancetogo). The axial interface also returns status signals of the axis that may need to be evaluated (e.g., exact stop, traverse command).

⚠ **Warning**

If several block calls (FC 15, FC 16, FC 18) are programmed for the same axis/spindle in the PLC user program, then the functions concerned must be interlocked by conditional calls in the user program.
The conditional call of a started block (parameter Start or Stop = TRUE) must be called cyclically until the signal state of output parameter "Active" or "InPos" changes from 1 to 0.

Error identifiers

If a function could not be executed, this is indicated by the "Error" status parameter being set to 'logical 1'. The error cause is coded at block output "State".

For listing of error identifiers see table in the chapter Block description, "FC 18: SpinCtrl Spindle control".

Declaration of the function

```
FUNCTION FC 15: VOID                          //POS_AX
VAR_INPUT
     Start:              BOOL ;
     AxisNo:             INT ;
     IC:                 BOOL ;
     Inch:               BOOL ;
     HWheelOv:           BOOL ;
     Pos:                REAL;
     FRate:              REAL;
END_VAR
VAR_OUTPUT
     InPos:              BOOL ;
     Active:             BOOL ;
     StartErr:           BOOL ;
     Error :             BOOL ;
END_VAR
```

Description of formal parameters

The table below lists all formal parameters of the POS_AX function:

Signal	I/O	Type	Value range	Remark
Start	I	BOOL		
AxisNo	I	BYTE	1 - 31	No. of axis to be traversed
IC	I	BOOL		0 = Absolute 1 = Incremental
Inch	I	BOOL		0 = mm 1 = Inch
HWheelOv	I	BOOL		1 = Handwheel override
Pos	I	REAL	∓ 0,1469368 I -38 to ∓ 0,1701412 I +39	Position of linear axis: mm Rotary axis: Degr.
FRate	I	REAL	∓ 0,1469368 I -38 to ∓ 0,1701412 I +39	Feedrate of linear axis: mm/Min Rotary axis: deg/min
InPos	A	BOOL		1 = In position
Active	A	BOOL		1 = Active
StartErr	A	BOOL		Axis cannot be started.
Error	A	BOOL		Error during traversing [1]
[1]Error evaluation by user in the PLC				

Timing diagram

(1)   Activation of function

(2)   Positioning axes active

(3)   Positive acknowledgment: Position reached

(4)   Reset function activation after receipt of acknowledgment

(5)   Signal change using FC

(6)   Reset function activation after receipt of active signal

Timing diagram (fault scenario)
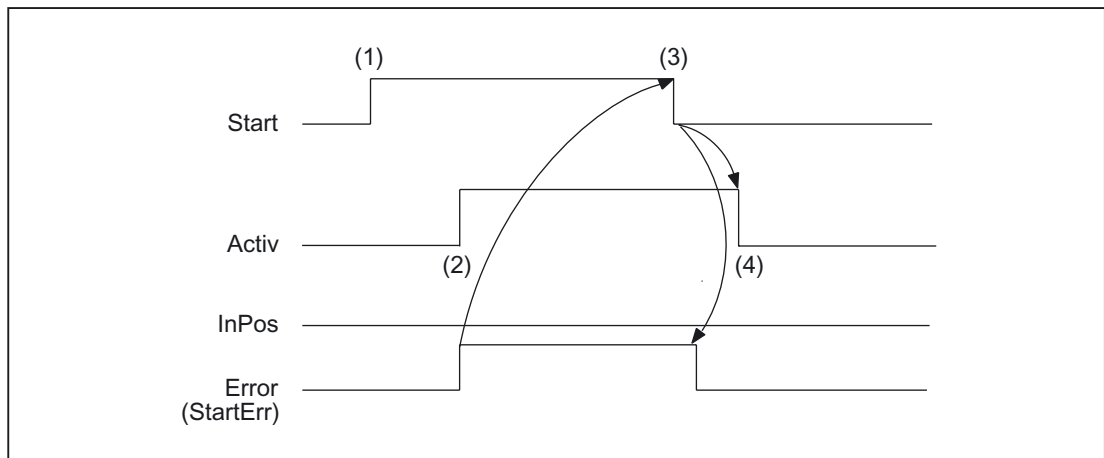
(1)   Activation of function by means of a positive edge

(2)   Negative acknowledgment: Error has occurred

(3)   Reset function activation after receipt of acknowledgment

(4)   Signal change using FC

Call example

```
CALL FC 15 (
            Start :=          TRUE,
            AxisNo :=         5,
            IC :=             #incr,           //e.g., local variable
            Inch :=           FALSE,
            HWheelOv :=       FALSE,
            Pos :=            MD160,
            FRate :=          MD164,
            InPos :=          Q 36.0,
            Active :=         Q 36.1,
            StartErr :=       Q 36.2,
            Error :=          Q 36.3);
```

2.12.20   FC 16: PART_AX positioning of indexing axes

Description of Functions

(Do not use for new applications, function is integrated in FC 18 in SW 3.6 and higher.)

FC PART_AX can be used to traverse NC axes defined via machine data as "indexing axes", also from the PLC.

References:
/FB2/Function Manual Expanded Functions; Indexing Axes (T1)

In order to traverse the indexing axes via the PLC, the traversing check must be activated for the PLC. This can be achieved, for example, by calling FC "PART_AX" with activation of the "Start" parameter. The FC "PART_AX" then requests checking of the axes from the NC. The NC feeds back the status of this axis in byte 68 in the associated axis interface DB 31, ... see lists/ (Book 2), Interface signals power line.

Upon completion ("InPos" is True, "Start" changes to zero), the axis/spindle check function is switched to a neutral status by FC PART_AX. Alternatively, the PLC user program can also request the check for the PLC prior to calling FC "PART_AX".

By calling this function several times in succession, a better response reaction by the spindle/axis can be obtained as the changeover process in the FC can be omitted.

Activation through the PLC user program is executed in the corresponding axis interface in byte 8.

After return of the check, the axis can again be programmed by the NC program.

Note

After the FC has been called, ACCU1 contains an error statement by the NCK (but not if the output parameters are assigned to a data block). This is generally the value 0 (signifies: No error has occurred). See table for the interpretation of other numerical values for error codes in FC 15.

FC 16 must be called cyclically until the "Active" signal produces an edge transition from 1 to 0. Only when the "Active" signal has a 0 state can the axis concerned be restarted (the next start must be delayed by at least one PLC cycle). This also applies when the assignment in data byte 8 has changed.

The function cannot be aborted by means of parameter "Start", but only by means of the axial interface signals (e.g., delete distancetogo).
The axial interface also returns status signals of the axis that may need to be evaluated (e.g., exact stop, traverse command).

> **Warning**
>
> If several block calls (FC 15, FC 16, FC 18) are programmed for the same axis/spindle in the PLC user program, then the functions concerned must be interlocked by conditional calls in the user program. The conditional call of a started block (parameter "Start" or "Stop" = TRUE) must be called cyclically until the signal state of output parameter "Active" or "InPos" changes from 1 to 0.

Declaration of the function

```
FUNCTION FC 16: VOID                          //PART_AX
VAR_INPUT
        Start:              BOOL ;
        AxisNo:             INT ;
        IC:                 BOOL ;
        DC:                 BOOL
        Minus:              BOOL ;             //Movement in the negative direction
        Plus:               BOOL ;             //Movement in the positive direction


        Pos:                INT ;
        FRate:              REAL;
END_VAR
VAR_OUTPUT
        InPos:              BOOL ;
        Active:             BOOL ;
        StartErr:           BOOL ;
        Error :             BOOL ;
END_VAR
```

Description of formal parameters

The table below lists all formal parameters of the PART_AX function:

Signal	I/O	Type	Value range	Remark
Start	I	BOOL		
AxisNo	I	INT	1 - 31	No. of axis to be traversed
IC	I	BOOL		Directional data 0 = absolute 1 = incremental
DC	I	BOOL		0 = specified direction 1 = shortest path when DC = 1 the parameters IC, Minus, Plus must be = 0
Minus	I	BOOL		0 : Rotary axis motion as for linear axis 1: Motion in negative direction with rotary axes
Plus	I	BOOL		0 : Rotary axis motion as for linear axis 1: Motion in positive direction with rotary axes
Pos	I	INT	0 to +32767	No. of indexing position
FRate	I	REAL	± 0,1469368 E -38 to ± 0,1701412 E +39	Feedrate of linear axis: mm/min Rotary axis: deg/min
InPos	A	BOOL		1 = In position
Active	A	BOOL		1 = Active
StartErr	A	BOOL		Axis cannot be started.
Error	A	BOOL		Error during traversing [1]
[1] Error evaluation by user in the PLC				

Timing diagram

(1)   Activation of function by means of a positive edge

(2)   Positioning axes active

(3)   Positive acknowledgment: Position reached

(4)   Reset function activation after receipt of acknowledgment

(5)   Signal change using FC

(6)   Reset function activation after receipt of active signal

Timing diagram (fault scenario)
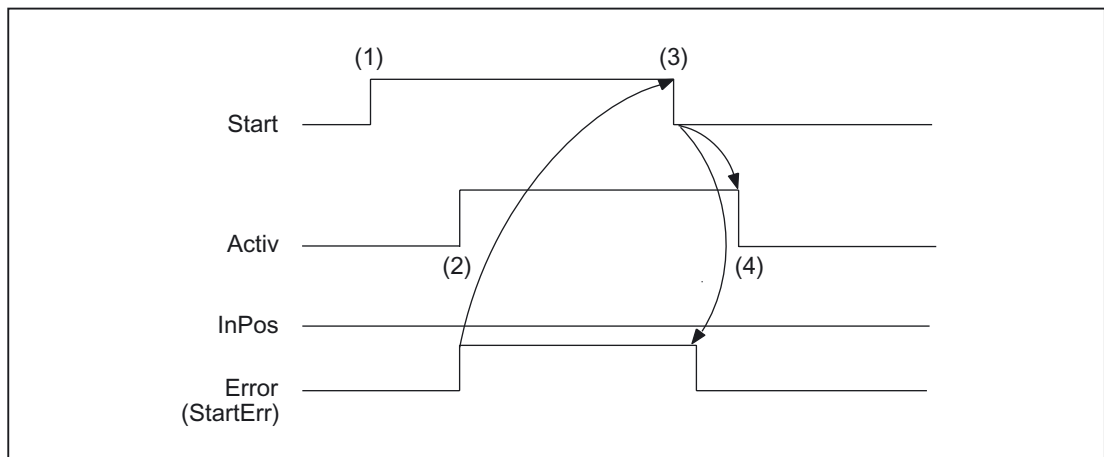
(1)   Activation of function by means of a positive edge

(2)   Negative acknowledgment: Error has occurred

(3)   Reset function activation after receipt of acknowledgment

(4)   Signal change using FC

Call example

```
CALL FC 16 (                      //Position an indexing axis
            Start :=        I72.4,
            AxisNo :=       6,
            IC :=           FALSE,
            DC :=           #short,              //e.g., local variable
            Minus :=        FALSE,
            Plus :=         FALSE,
            Pos :=          MW 168,
            FRate :=        MD164,
            InPos :=        Q 36.4,
            Active :=       Q 36.5,
            StartErr :=     Q 36.6,
            Error :=        Q 36.7);
```

2.12.21   FC 17: YDelta star/delta changeover

Description of Functions

The block for star/delta changeover controls the timing of the defined switching logic such that the changeover can be performed in either direction even when the spindle is running. This block may be used only for digital main spindle drives and must be called separately for each spindle.

The changeover operation is implemented via 2 separate contactors in a sequence involving 4 steps:

Step 1:	Deleting the interface signal DB31, ... DBX21.5 (Motor selection done) in the related axis-DB and register the changeover process via A with DB31, ... DBX21.3 (Motor selection).
Step 2:	As soon as the return message NST DB31, ... DBX93.7 (Pulse enabled) and the acknowledgment of the announced motor selection from the drive have appeared, the currently energized contactor drops out.
Step 3:	The other contactor is energized after the time period set by the user in parameter "TimeVal" has elapsed.
Step 4:	After a further delay, the changeover is signaled to the drive with NST DB31, ... DBX21.5 (Motor selection done) reported.
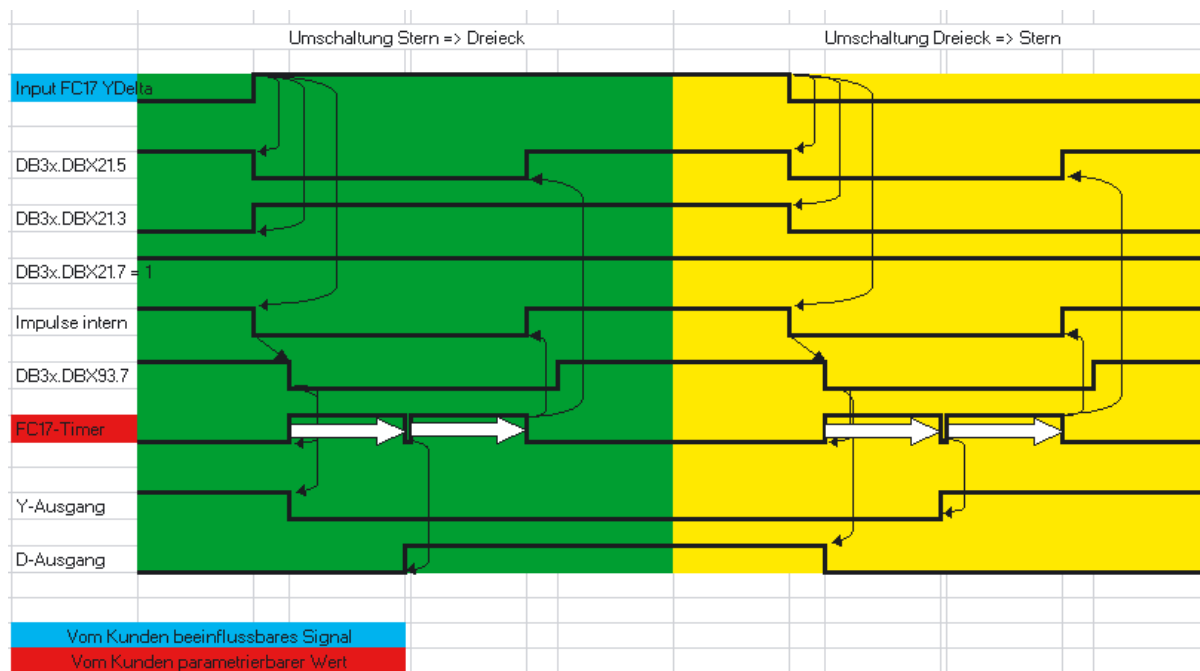
Figure 2-29   Star/delta changeover

For more detailed information about motor speed adjustments, please see:
References:
/FB1/Function Manual, Basic Functions; Spindles (S1);
Chapter "Configurable gear adjustments"
/FB1/Function Manual Basic functions; Velocities, Actual/Set point system, Regulation (G2)

Error message

If the parameter "SpindleIFNo" is not in the permissible range, the PLC is stopped with output of interrupt message number 401702.

Special features

When parameterizing "TimeVal" with the value 0, a default value of 100 ms is used. With a value of less than 50 ms, the minimum setting of 50 ms is applied.

The block must be called unconditionally.

Note

Switchover does not take place if the spindle is in an axis mode such as M70, SPOS.

Restrictions

Star/delta changeover on digital main spindle drives initiates a process, which contains closed-loop control sequences. Since the closed-loop control system supports automatic star/delta changeover, certain restrictions should be noted:

- Owing to the automatic deactivation of the pulse in the drive, simultaneously with the NST DB31, ... DBX93.7 (Pulse enabled) the NST DB31, ... DBX61.7 (current regulator active) and DB31, ... DBX61.6 (speed regulator active) are deactivated.

- In case of rotating spindle and activated position controller the spindle with NST DB31, ... DBX61.5 (position controller active) is changed over from star to delta, then it leads to the alarm 25050 "Contour monitoring".

- Once the star/delta changeover has been initiated with FC17, it cannot be delayed by the user, e.g., by waiting until the star/delta contactors change over during the course of operation. The user can implement this signal interaction with PLC logic.

Declaration of the function

STL representation

```
VAR_INPUT
      YDelta:           BOOL ;                  //Star = 0, delta = 1
      SpindleIFNo:      INT ;                   //Machine axis number
      TimeVal:          S5TIME ;                //Time value
      TimerNo :         INT ;                   //User timer for changeover time
END_VAR
VAR_OUTPUT
      Y:                BOOL ;                  //Star contactor
      Delta:            BOOL ;                  //Delta contactor
END_VAR
VAR_IN_OUT
      Ref:              WORD ;                  //Block status word (instance)
END_VAR
```

Description of formal parameters

The table below lists all formal parameters of the YDelta function:

Signal	Type	Type	Value range	Remark
YDelta	I	BOOL		= star = delta. The changeover edge of the signal initiates the changeover operation.
SpindleIFNo	I	INT	1..	Number of the axis interface declared as a spindle
TimeVal	I	S5time	0..	Switchover time
TimerNo	I	INT	10..	Timer for programming the wait time
Y	Q	BOOL		Energizing of star contactor
Delta	Q	BOOL		Energizing of delta contactor
Ref	I/O	WORD		Instance for status information Internal use

Call example

```
CALL FC 17 (
    YDelta :=        I 45.7,              //Star delta
    SpindleIFNo :=   4,
    TimeVal :=       S5T#150ms,
    TimerNo :=       10,                  //Timer 10
    Y :=             Q 52.3,              //Star contactor
    Delta :=         Q 52.4,              //Delta contactor
    Ref :=           mw 50);             //Instance
```

2.12.22   FC 18: SpinCtrl spindle control

Description of Functions

FC SpinCtrl can be used to control spindles and axes from the PLC.

References
/FB1/Function Manual, Basic Functions; Spindles (S1)
/FB2/Function Manual, Extended Functions; Positioning Axes (P2)
/FB/Function Manual, Extended Functions; Indexing Axes (T1)

This block supports the following functions:

- Position spindle

- Rotate spindle

- Oscillate spindle

- Indexing axes

- Positioning axes

Each function is activated by the positive edge of the appropriate initiation signal (start, stop). This signal must remain in the logic "1" state until the function has been acknowledged positively or negatively by InPos="1" or Error = "1". The output parameters are deleted when the relevant trigger signal is reset and the function terminated.

To be able to control an axis or spindle via the PLC, it must be activated for the PLC. This can, for example, be achieved by calling the FC "SpinCtrl" with activation of the "Start" or "Stop" parameter. In this case, the FC "SpinCtrl" requests control over the spindle/axis from the NC.

The NC feeds back the status of this axis in byte 68 in the associated spindle/axis interface DB 31, ... see lists/ (Book 2), Interface signals power line.

Once the axis/spindle is operating under PLC control, the travel command for the active state can be evaluated via the relevant axis interface.

On completion ("InPos" is TRUE, "Start" changes to zero), the axis/spindle check function is switched to a neutral status by FC "SpinCtrl". Alternatively, the PLC user program can also request the check for the PLC prior to calling FC "SpinCtrl".

By calling this function several times in succession, a better response reaction by the spindle/axis can be obtained as the changeover process in the FC can be omitted.

Activation through the PLC user program is executed in the corresponding spindle interface in byte 8.

After return of the check, the spindle can again be programmed by the NC program.

Note

Please note:

FC 18 must be called cyclically until signal "InPos" or, in the case of an error "Error", produces an edge transition of 1 to 0. Only when the "InPos"/"Error" signal has a 0 state, another "Start" or "Stop" is possible for this spindle/axis (the next "Start" or "Stop" must be delayed by at least one PLC cycle). This also applies when the assignment in data byte 8 on the axial interface has been changed.

Abort:

The function cannot be aborted by means of parameter "Start" or "Stop", but only by means of the axial interface signals (e.g., delete distancetogo).
The axial interface also returns status signals of the axis that may need to be evaluated (e.g., exact stop, traverse command).

InPos on spindle - rotation/oscillation:

For the function "Rotate spindle" as also for "Oscillate spindle" the meaning of the parameter "InPos" is defined as follows:
Set speed is output --> Function started without error.
Reaching the desired spindle speed must be evalated via the spindle interface.

Simultaneity:

Several axes can be traversed simultaneously or subject to a delay by the blocks FC 15, 16 and 18. The upper limit is defined by the maximum number of axes. The NCK handles the PLC function request (FC 15, 16, 18) via independent interfaces for each axis/spindle.

Axis disable:

With the axis disabled, an axis controlled via FC 18 will not move. Only a simulated actual value is generated. (Behavior as with NC programming).

Warning

If several block calls (FC 15, FC 16, FC 18) are programmed for the same axis/spindle in the PLC user program, then the functions concerned must be interlocked by conditional calls in the user program. The conditional call of a started block (parameter "Start" or "Stop" = TRUE) must be called cyclically until the state change of output parameter Active or InPos takes place from 1 to 0.

Functions

1. Position spindle:

The following signals are relevant:

Start:	Initiation signal
Funct:	"1" = Position spindle
Mode:	Positioning mode 1, 2, 3, 4
AxisNo:	Number of machine axis
Pos:	Position
FRate:	Positioning speed, if FRate = 0, value from MD35300: SPIND_POSCTRL_VELO (position control activation speed) is used
InPos:	Is set to "1" when position is reached with "Exact stop fine".
Error :	With positioning error = "1"
State :	Error code

2. Rotate spindle:

The following signals are relevant:

Start:	Initiation signal for start rotation
Stop:	Initiation signal for stop rotation
Funct:	"2" = Rotate spindle
Mode:	Positioning mode 5 (direction of rotation M4)
	Positioning mode < >5 (direction of rotation M3)
AxisNo:	Number of machine axis
FRate:	Spindle speed
InPos:	Function has started without an error
Error :	With positioning error = "1"
State :	Error code

3. Oscillate spindle:

The following signals are relevant:

Start:	Initiation signal for start oscillation
Stop:	Initiation signal for stop oscillation
Funct:	"3" = Oscillate spindle
AxisNo:	Number of machine axis
Pos:	Set gear stage
InPos:	Setpoint speed is output
Error :	With positioning error = "1"
State :	Error code

The oscillation speed originates from the machine data:
MD35400 SPIND_OSCILL_DES_VELO

MD35010 GEAR_STEP_ CHANGE_ENABLE = 0	Function	MD35010 GEAR_STEP_ CHANGE_ENABLE = 1	Function
Pos = 0	Oscillation	Pos = 0	
Pos = 1	Oscillation	Pos = 1	Oscillation with gear stage change M41
Pos = 2	Oscillation	Pos = 2	Oscillation with gear stage change M42
Pos = 3	Oscillation	Pos = 3	Oscillation with gear stage change M43
Pos = 4	Oscillation	Pos = 4	Oscillation with gear stage change M44
Pos = 5	Oscillation	Pos = 5	Oscillation with gear stage change M45

4. Traverse indexing axis:

The following signals are relevant:

Start:	Initiation signal
Funct:	"4" = Indexing axis

Note

With
Funct: "4" = Indexing axis

The modulo conversion can be compared with approaching the indexing position via POS[AX] = CIC (value) in the part program.

Mode:	Positioning mode 0, 1, 2, 3, 4
AxisNo:	Number of machine axis
Pos:	Indexing position
FRate:	Positioning speed; if FRate = 0, the value is taken from machine data POS_AX_VELO (unit as set in machine data).
InPos:	Is set to "1" when position is reached with "Exact stop fine".
Error :	With positioning error = "1"
State :	Error code

5 to 8. Position axes:

The following signals are relevant:

Start:	Initiation signal
Funct:	"5 to 8" = Position axes
Mode:	Positioning mode 0, 1, 2, 3, 4
AxisNo:	Number of machine axis
Pos:	Position
FRate:	Positioning speed; if FRate = 0, the value is taken from machine data POS_AX_VELO (unit as set in machine data).
InPos:	Is set to "1" when position is reached with "Exact stop fine".
Error :	With positioning error = "1"
State :	Error code

9. Rotate spindle with automatic gear stage selection:

The following signals are relevant:

Start:	Initiation signal for start rotation
Stop:	Initiation signal for stop rotation
Funct:	"9" = Rotate spindle with gear stage selection
Mode:	Positioning mode 5 (direction of rotation M4)
	Positioning mode < >5 (direction of rotation M3)
AxisNo:	Number of machine axis
FRate:	Spindle speed
InPos:	Setpoint speed is output
Error :	With positioning error = "1"
State :	Error code

10/11. Rotate spindle with constant cutting rate:

The "Constant cutting rate" function must be activated by the NC program in order for this to be executed.

The following signals are relevant:

Start:	Initiation signal for start rotation
Stop:	Initiation signal for stop rotation
Funct:	"B#16#0A = Rotate spindle with constant cutting rate (m/min)
Funct:	"B#16#0B = Rotate spindle with constant cutting rate (feet/min)
Mode:	Positioning mode 5 (direction of rotation M4)
	Positioning mode < >5 (direction of rotation M3)
AxisNo:	Number of machine axis
FRate:	cutting rate
InPos:	Setpoint speed is output
Error :	With position error = "1"
State :	Error code

Declaration of the function

```
FUNCTION FC 18: VOID                            //SpinCtrl
VAR_INPUT
      Start:                BOOL ;
      Stop:                 BOOL ;
      Funct:                BYTE ;
      Mode:                 BYTE ;
      AxisNo:               INT ;
      Pos:                  REAL;
      FRate:                REAL;
END_VAR
VAR_OUTPUT
      InPos:                BOOL ;
      Error :               BOOL ;
      State :               BYTE ;
END_VAR
```

Description of formal parameters

The table below lists all formal parameters of the SpinCtrl function.

Signal	I/O	Type	Value range	Remark
Start	I	BOOL		Start spindle control from PLC
Stop	I	BOOL		Stop spindle control from PLC
Funct	I	BYTE	1 to B#16#0B	1: Position spindle
				2: Rotate spindle
				3: Oscillate spindle
				4: Indexing axis
				5: PosAxis metric
				6: PosAxis inch
				7: PosAxis metric with handwheel override
				8: PosAxis inch with handwheel override
				9: Rotate spindle with automatic gear stage selection
				A: Rotate spindle with constant cutting rate (m/min)
				B: Rotate spindle with constant cutting rate (feet/min)
Mode	I	BYTE	0 to 5	0: Pos to absolute pos
				1: Pos incrementally
				2: Pos shortest path
				3: Pos absolute, positive approach direction
				4: Pos absolute, negative approach direction
				5: Rotational direction as for M4
AxisNo	I	INT	1 - 31	No. of axis/spindle to be traversed
Pos	I	REAL	∓ 0,1469368 I -38 to ∓ 0,1701412 I +39	Rotary axis: Degrees Indexing axis: Indexing position Linear axis: mm or inches
FRate	I	REAL	∓ 0,1469368 I -38 to ∓ 0,1701412 I +39	Rotary axis and spindle: rev/min See under table containing info about FRate
InPos	Q	BOOL		1 = Position reached, or function executed
Error	Q	BOOL		1 = Error
State	Q	BYTE	0 to 255	Error code

FRate

The feed rate in FC 18 can also be specified as:

1. Cutting rate with unit m/min or feet/min

2. Constant grinding wheel surface speed in m/s or feet/s

These alternative velocity settings can be used only if this function is activated by the NC program. Checkback signals for successful activation can be found in byte 84 on the axis interface.

Error identifiers

If a function could not be executed, this is indicated by the "Error" status parameter being set to 'logical 1'. The error cause is coded at block output "State".

State	Description
Errors caused by PLC handling	
1	Several axis/spindle functions have been activated simultaneously
20	A function has been started without the position being reached
30	The axis/spindle has been transferred to the NC while still in motion
40	NCK internal error
Errors that occur due to handling of the NCK. The alarm numbers are described in the Diagnostics Manual SINUMERIK 840D/840Di/810D.	
100	corresponds to alarm number: 16830
105	corresponds to alarm number: 16770
106	corresponds to alarm number: 22052
107	corresponds to alarm number: 22051
108	corresponds to alarm number: 22050
109	corresponds to alarm number: 22055
110	Velocity/speed is negative
111	Setpoint speed is zero
112	Invalid gear stage
115	Programmed position has not been reached
117	G96/G961 is **not active** in the NC
118	G96/G961 is **still active** in the NC
120	Not an indexing axis
121	Indexing position error
125	DC (shortest path) not possible
126	Minus absolute value not possible
127	Plus absolute value not possible
130	Software limit switch plus
131	Software limit switch minus
132	Working area limitation plus
133	Working area limitation minus
System or other serious interrupts:	
200	corresponds to system alarm number: 450007

Timing diagram
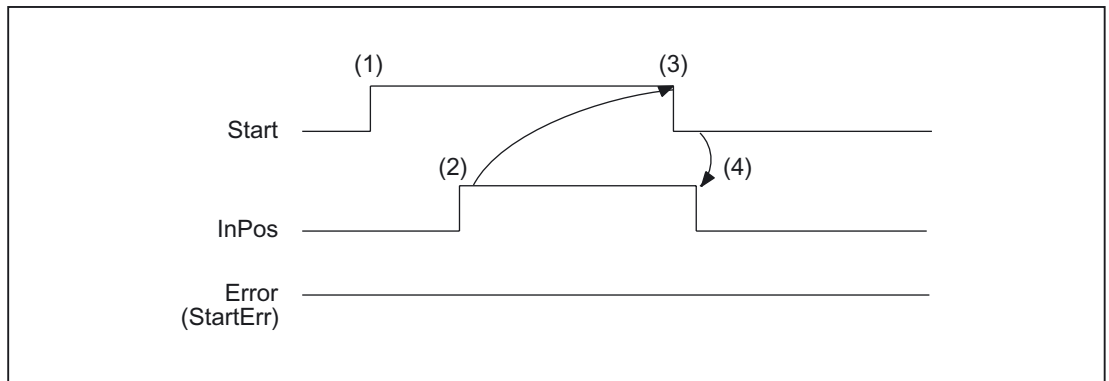
(1)	Activation of function by means of a positive signal edge with start or stop
(2)	Positive acknowledgment: Function executed/Position reached
(3)	Reset function activation after receipt of acknowledgment
(4)	Signal change using FC

Timing diagram (fault scenario)
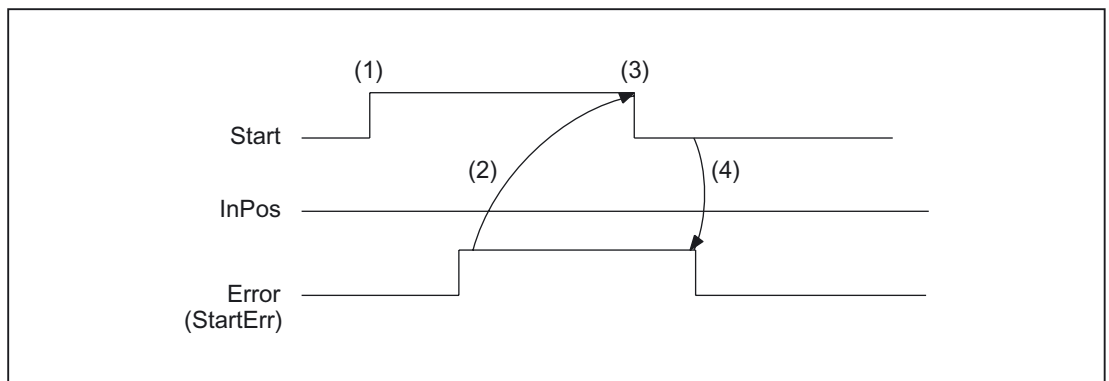
(1)	Activation of function by means of a positive signal edge with start or stop
(2)	Negative acknowledgment: Error has occurred
(3)	Reset function activation after receipt of acknowledgment
(4)	Signal change using FC

Call examples

1. Position spindle:

```
//Positive acknowledgment resets Start:
U M112.0;          //InPos
R M 100.0;         //Start
//Negative acknowledgment, after error evaluation (state: MB114) reset with T12
start
U M113.0;          // Error
U I 6.4;           //Key T12
R M 100.0;         //Start
//Start with T13
U I 6.3;           //Key T13
AN F 112.0;        //Restart only when InPos or Error = 0
AN F 113.0;
S M 100..0;

CALL FC 18 (
        Start :=          M100.0,
        Stop :=           FALSE,
        Funct :=          B#16#1,          //Position spindle
        Mode :=           B#16#2,          //Shortest path
        AxisNo :=         5,
        Pos :=            MD104,
        FRate :=          MD108,
        InPos :=          M112.0,
        Error :=          M113.0,
        State :=          MB114);
```

2. Start spindle rotation:

```
CALL FC 18 (
        Start :=          M100.0,
        Stop :=           FALSE,
        Funct :=          B#16#2,          //Rotate spindle
        Mode :=           B#16#5,          //Rotational direction as for M4
        AxisNo :=         5,
        Pos :=            0.0,
        FRate :=          MD108,
        InPos :=          M112.0,
        Error :=          M113.0,
        State :=          MB114);
```

3. Start spindle oscillation

```
CALL FC 18 (
         Start :=              M100.0,
         Stop :=               FALSE,
         Funct :=              B#16#3,              //Oscillate spindle
         Mode :=               B#16#0,
         AxisNo :=             5,
         Pos :=                0.0,
         FRate :=              MD108,
         InPos :=              M112.0,
         Error :=              M113.0,
         State :=              MB114);
```

4. Traverse indexing axis

```
CALL FC 18 (
         Start :=              M100.0,
         Stop :=               FALSE,               //Not used
         Funct :=              B#16#4,              //Traverse indexing axis
         Mode :=               B#16#0,              //Position absolutely
         AxisNo :=             4,
         Pos :=                MD104,               //Default setting in REAL: 1.0;2.0;..
         FRate :=              MD108,
         InPos :=              M112.0,
         Error :=              M113.0,
         State :=              MB114);
```

5. Position axes

```
CALL FC 18 (
         Start :=              M100.0,
         Stop :=               FALSE,               //Not used
         Funct :=              B#16#5,              //Position axes
         Mode :=               B#16#1,              //Position incrementally
         AxisNo :=             6,
         Pos :=                MD104,
         FRate :=              MD108,
         InPos :=              M112.0,
         Error :=              M113.0,
         State :=              MB114);
```

2.12.23   FC 19: MCP_IFM transmission of MCP signals to interface

Description of Functions

With FC MCP_IFM (M variants), the following are transferred from the machine control panel (MCP) to the corresponding signals of the NCK/PLC interface:

- Mode groups
- Axis selections
- WCS/MCS switchover commands
- Traversing keys
- Overrides
- Key switch

In the basic program (FC 2), the handwheel selections, modes and other operation signals are transmitted from the operator panel or MMC to the NCK/PLC interface in such a way as to allow the modes to be selected from the machine control panel or the operator panel.

Transfer of HMI signals to the interface can be deactivated by setting the value of the parameter "MMCToIF" to "FALSE" in FB 1 (DB 7).

The following specifications apply to the **feed override, axis travel keys** and **INC keys** depending on the active mode or on the coordinate system selected:

- **Feed override**:
  - The feed override is transferred to the interface of the selected channel and to the interface of the axes.
  - The feed override signals are transferred in addition to interface byte "Rapid traverse" override (DBB 5) to the NC channel if MMC signal "Feed override for rapid traverse effective" is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this MMC signal.

- **Machine functions for INC and axis travel keys:**
  - When the MCS is selected, the signals are transferred to the interface of the selected machine axis.
  - When the WCS is selected, the signals are transferred to the geoaxis interface of the parameterized channel.
  - When the system is switched between MCS and WCS, the active axes are generally deselected.

The **handwheel selection signals from the MMC** are decoded and activated in the machine-axis or the Geo-axis interface of the handwheel selected (only if parameter "HWheelMMC := TRUE" in FB 1).

The LEDs on the machine control panel derived from the selections in the acknowledgment.

Feedrate and spindle Start/Stop are not transferred to the interface, but output modally as a "FeedHold" or "SpindleHold" signal. The user can link these signals to other signals leading to a feed or spindle stop (this can be implemented, e.g., using the appropriate input signals in FC 10: AL_MSG). The associated LEDs are activated at the same time.

If the machine control panel fails, the signals it outputs are preset to zero; this also applies to "FeedHold" and "SpindleHold" output signals.

Multiple calls of FC 19 or FC 24, FC 25, FC 26 are permitted in a single PLC cycle. In this case, the first call in the cycle drives the LED displays. Furthermore, all actions of the parameterized block are carried out in the first call. In the following calls, only a reduced level of processing of the channel and mode group interface takes place. The geometry axes are supplied with directional data only in the first block call in the cycle.

Single block processing can be selected/deselected only in the first call in the cycle.

The second machine control panel can be processed if parameter "BAGNo" has been increased by B#16#10. When parameterizing, the HHU number is contained in the lower nibble (lower 4 bits).
"BAGNo" = 0 or B#16#10 means that the mode group signals are not processed.
"ChanNo" = 0 means that the channel signals are not processed.

The INC selections are transferred to the mode group interface. This results in runtime improvements. This specification is activated via the DB10 ... DBX57.0 (INC-inputs active in the BAG range) through this block once after power up.

Machine control panels can still be handled in parallel by this module. The module 2 call for the 2nd machine control panel in OB1 cycle must come after the call of the 1st A support for 2 MSTT is still provided within certain restrictions in the control panel blocks (support is not provided for standard axis numbers 10 to 31, mutual interlocking of axis selections with 2 MSTT).

Flexible axis configuration

With SW V6 and higher, it is possible to be flexible in the assignment of axis selections or direction keys of machine axis numbers.

Better support is now provided by the MSTT blocks for use of two MCPs, which are operated simultaneously, in particular for an application using two channels and two mode groups. Note that the axis-numbers are also specified in the parameterized mode group number of the MCP block in the axis tables of the relevant MCP.

For this flexibility there are tables for axis numbers in DB 10.
For **1.** Machine control panel (MCP) the table starts from the byte 8
(symbolic name: MCP1AxisTbl[1..22]) and
for **2.** Machine control panel (MCP) the table starts from the byte 32
(symbolic name: MCP2AxisTbl[1..22]) for the second MCP. The machine axis numbers must be entered byte by byte here.

It is permissible to enter a value of 0 in the axis table. Checks are not made to find impermissible axis numbers, meaning that false entries can lead to a PLC Stop.

For **FC 19**, the **maximum possible number of axis selections** can also be restricted.

This upper limit is set for the
1. Machine Control Panel in DB10, ... DBW30 (symbolic Name: MCP1MaxAxis) or for the
2. Machine Control Panel in DB10, ... DBW54 (symbolic Name: MCP2MaxAxis) for the
respective MSTT.

The default setting is 0, corresponding to the maximum number of configured axes. The axis
numbers and the limit can also be adapted dynamically. Afterwards, a new axis must be
selected on FC 19. Axis numbers may not be switched over while the axes are traversing the
relevant direction keys.
The compatibility mode is preset with axis numbers **1 to 9** for both MCPs and the restriction
for the configured number of axes.

Declaration of the function

```
FUNCTION FC 19: VOID
// NAME:                              MCP_IFM


VAR_INPUT
    BAGNo :                          BYTE ;
    ChanNo:                          BYTE ;
    SpindleIFNo:                     BYTE ;
END_VAR


VAR_OUTPUT
    FeedHold :                       BOOL ;
    SpindleHold :                    BOOL ;
END_VAR



BEGIN
END_FUNCTION
```

Description of the formal parameters

The table below shows all formal parameters of the "MCP_IFM" function:

Signal	I/O	Type	Value range	Remark
BAGNo	I	BYTE	0 - b#16# and b#16#10 - b#16#1A	No. of mode group to which the mode signals are transferred. BAGNo >= b#16#10 means access to the second machine control panel.
ChanNo	I	BYTE	0 - B#16#0A	Channel no. for the channel signals
SpindleIFNo	I	BYTE	0 - 31 (B#16#1F)	Number of the axis interface declared as a spindle
FeedHold	Q	BOOL		Feed stop from machine control panel, modal
SpindleHold	Q	BOOL		Spindle stop from machine control panel, modal

MCP selection signals to the user interface

Key switch

Source: MCP-Switch	Destination: Interface DB
Position 0	DB10.DBX56.4
Position 1	DB10.DBX56.5
Position 2	DB10.DBX56.6
Position 3	DB10.DBX56.7

Operating modes and machine functions

Source: MCP key	Destination: Interface DB (parameter ModeGroupNo)
AUTOMATIC	DB11, ... DBX0.0
MDA	DB11, ... DBX0.1
JOG	DB11, ... DBX0.2
REPOS	DB11, ... DBX1.1
REF	DB11, ... DBX1.2
TEACH IN	DB11, ... DBX1.0
INC 1 ... 10 000, INC Var. (starting from SW 5)	DB11, ... DBB2, Bit 0 to 5

Source: MCP key	Destination: Interface DB (parameter ChanNo)
INC 1 ... 10 000, INC Var. (till SW 4)	DB21, ... DBB13, Bit 0 to 5
INC 1 ... 10 000, INC Var. (till SW 4)	DB21, ... DBB17, Bit 0 to 5
INC 1 ... 10 000, INC Var. (till SW 4)	DB21, ... DBB21, Bit 0 to 5

Source: MCP key	Destination: Interface DB (all axis DBs)
INC 1 ... 10 000, INC Var. (till SW 4)	DB31, ... DBB5, Bit 0 to 5

Direction keys rapid traverse override

The transfer is dependent upon the selected axis. The associated interface bits are deleted for axes, which are not selected.

Source: MCP button	Destination: Interface DB (Parameter ChanNo)
Direction key +	DB21, ... DBX12.7
Direction key -	DB21, ... DBX12.6
Rapid traverse override	DB21, ... DBX12.5
Direction key +	DB21, ... DBX16.7
Direction key -	DB21, ... DBX16.6
Rapid traverse override	DB21, ... DBX16.5
Direction key +	DB21, ... DBX20.7
Direction key -	DB21, ... DBX20.6
Rapid traverse override	DB21, ... DBX20.5

Source: MCP key	Destination: Interface DB (all axis DBs)
Direction key +	DB31, ... DBX4.7
Direction key -	DB31, ... DBX4.6
Rapid traverse override	DB31, ... DBX4.5

Override

Source: MCP-Switch	Destination: Interface DB (parameter ChanNo)
Feedrate override	DB21, ... DBB4

Source: MCP-Switch	Destination: Interface DB (all axis DBs)
Feedrate override	DB31, ... DBB0 (selected axis number) The feed override of the 1st MCP applies to all axes.
Spindle override	DB31, ... DBB19 (parameter SpindleIFNo)

Channel signals

Source: MCP - keys	Destination: Interface DB (parameter ChanNo)
NC Start	DB21, ... DBX7.1
NC stop	DB21, ... DBX7.3
RESET	DB21, ... DBX7.7
Single block	DB21, ... DBX0.4

Feedrate, spindle signals

Source: MCP - keys	Destination: FC output parameters
Feed stop Feed enable	Parameter: "FeedHold" latched, LEDs are driven
Spindle stop Spindle enable	Parameter: "SpindleHold" latched, LEDs are driven

Checkback signals from user interface for controlling displays

Operating modes and machine functions

Destination: MCP - LED	Source: Interface DB (parameter ModeGroupNo)
AUTOMATIC	DB11, ... DBX6.0
MDA	DB11, ... DBX6.1
JOG	DB11, ... DBX6.2
REPOS	DB11, ... DBX7.1
REF	DB11, ... DBX7.2
TEACH IN	DB11, ... DBX7.0

Destination: MCP - LED	Source: Interface DB (parameter ModeGroupNo)
INC 1..10000, INC Var.	DB11, ... DBB8, Bit0 to Bit5

Channel signals

Destination: MCP - LED	Source: Interface DB (parameter ChanNo)
NC Start	DB21, ... DBX35.0
NC stop	DB21, ... DBX35.2 or DB21, ... DBX35.3
Single block	DB21, ... DBX0.4

Note

Direction key LEDs are controlled by operating the direction keys.
Axis selection and WCS/MCS LEDs are controlled by operating the relevant pushbutton switch.

Call example

```
CALL FC 19 (              //Machine control panel M variants signals to interface
        BAGNo :=         B#16#1,          //Mode group no. 1
        ChanNo :=        B#16#1,          //Channel no. 1
        SpindleIFNo :=   B#16#4,          //Spindle interface number = 4
        FeedHold :=      m22.0,           //Feed stop signal modal
        SpindleHold :=   db2.dbx151.0);   //Spindle stop modal in
                                          //message DB
```

With these parameter settings, the signals are sent to the 1st mode group, the 1st channel and all axes. In addition, the spindle override is transferred to the 4th axis/spindle interface. The feed hold signal is passed to bit memory 22.0 and the spindle stop signal to data block DB2, data bit 151.0.

Reconnecting the axis selections

To ensure a flexible assignment of the axis selection keys to the appropriate axis or spindle, FC 19 **needs not be modified or reprogrammed**.

The required flexibility can be obtained by applying the solution described below.

1. Before FC 19 is called, the information (RLO) relating to the newly defined axis selection key is transferred to the key selection identified by an axis number.

2. After FC 19 has been called, the information (RLO) relating to the LED identified by the axis number is transferred to the LED of the new axis selection key and the RLO of the previous axis LED is then deleted.

Example:

The spindle is defined as the 4th axis and must be selected via axis key 9.

```
STL extract:


u i                                      //Selection of ninth axis
5.2;

= e                                      //Selection of fourth axis
4.2;


CALL FC 19 (              //signals to interface
        BAGNo :=         B#16#1,         //Mode group no. 1
        ChanNo :=        B#16#1,         //Channel no. 1
        SpindleIFNo :=   B#16#4,         //Spindle interface number = 4

        FeedHold :=      m30.0,          //Feed stop signal modal
        SpindleHold :=   m30.1);         //Spindle stop modal


u a                                      //LED fourth axis
2.5;

= a                                      //LED ninth axis
3.3;


= a                                      //Switch off LED on fourth axis
2.5;
```

2.12.24   FC 21: transfer PLC NCK data exchange

Description of functions

When the Transfer block is called, data are exchanged between the PLC and NCK according to the selected function code. Data are transferred as soon as FC 21 is called, not only at the start of the cycle.

The "Enable" signal activates the block.
FC 21 is processed only when "Enable" = "1".

The following functions for the data exchange between PLC and NCK are supported:

1. Signal synchronized actions at the NCK - channel

2. Signals synchronized actions from NCK - channel

3. Fast data exchange PLC-NCK (Read function in NCK)

4. Fast data exchange PLC-NCK (Write function in NCK)

5. Update control signals to NCK - channel

6. Update control signals to axes (data byte 2 of the user interface)

7. Update control signals to axes (data byte 4 of the user interface)

Declaration of the function

STL representation

```
VAR_INPUT
      Enable :              BOOL ;
      Funct:                BYTE ;
      S7Var :               ANY ;
      IVar1 :               INT ;
      IVar2 :               INT ;
      END_VAR
      VAR_OUTPUT
      Error :               BOOL ;
      ErrCode :             INT ;
END_VAR
```

Explanation of formal parameters

The table below shows all formal parameters of the "Transfer" function.

Signal	Type	Type	Value range	Remark
Enable	I	BOOL		1 = FC 21 active
Funct	I	BYTE	1.. 7	1: Synchronized actions to channel 2: Synchronized actions from channel 3: Read data 4: Write data 5: Control signals to channel 6, 7: Control signals to axis
S7Var	I	ANY	S7 data storage area	Depends on "Funct"
IVAR1	I	INT	0..	Depends on "Funct"
IVAR2	I	INT	1..	Depends on "Funct"
Error	A	BOOL		
ErrCode	A	INT		Depends on "Funct"

Functions

1: Signals for synchronized actions to channel:

2: Signals for synchronized actions from channel:

Synchronized actions can be disabled or enabled by the PLC.
The data area is stored on the user interface in DB21 to DB30.DBB 300..307 (to channel) and DBB 308..315 (from channel). The parameter "S7Var" is not evaluated for this function, but must be assigned an actual parameter (see call example). The data are transferred to/from the NC as soon as FC 21 is processed.

The following signals are relevant:

Signal	Type	Type	Value range	Remark
Enable	I	BOOL		1 = FC 21 active
Funct	I	BYTE	1, 2	1: Synchronized actions to channel 2: Synchronized actions from channel
S7Var	I	ANY	S7 data storage area	Not used
IVAR1	I	INT	1..MaxChannel	Channel number
Error	A	BOOL		
ErrCode	A	INT		1 : "Funct" invalid 10: Channel no. invalid

Call example

```
FUNCTION FC 100 : VOID
VAR_TEMP
    myAny:              ANY ;
END_VAR
BEGIN
NETWORK
...
// Deactivate synchronized actions with ID3, ID10 and ID31 in NC channel 1 :
    SYAK:              TO DB21;
    SET;
    S DBX300.2;        //ID3
    S DBX301.1;        //ID10
    S DBX303.6;        //ID31
    L B#16#1;
    T MB11;
    SPA TRAN;


// Synchronized actions from NCK channel:
SYVK:                  L B#16#2;
                       T MB11;
TRAN: CALL FC 21 (
                   Enable :=     M 10.0,        // if True, FC 21 is active
                   Funct :=      MB 11,
                   S7Var :=      #myAny,         //Not used
                   IVAR1 :=      1,              //Channel no.
                   IVAR2 :=      0,
                   Error :=      M 10.1,
                   ErrCode :=    MW 12);
...
END_FUNCTION
```

Functions

3, 4: Rapid PLC NCK data exchange

General

A separate, internal data area is provided to allow the highspeed exchange of data between the PLC and NCK. The size of the internal data field is preset to 1024 bytes. The accesses (read/write) from PLC take place via the FC 21. The occupation of this range (structure) must be defined identically in the NC part program and the PLC user program.

These data can be accessed from the NC parts program by commands $A_DBB[x], $A_DBW[x], $A_DBD[x], $A_DBR[x] (see /PGA1/ Parameter Manual System variables)).

The concrete address is the data field is specified by a byte offset (0 to 1023) in parameter IVAR1. In this case, the alignment must be selected according to the data format, i.e., a Dword starts at a 4byte limit and a word at a 2byte limit. Bytes can be positioned on any chosen offset within the data field, singlebit access operations are not supported and converted to a byte access operation by FC 21. Data type information and quantity of data are taken from the ANY parameter, transferred via S7Var.

Without additional programming actions, data consistency is only ensured for 1 and 2 byte access in the NCK and in the PLC. For the 2-byte consistency this is true only for the data type WORD or INT, but not for the data type BYTE.

In the case of longer data types or transfer of fields, which should be transferred consistently, a semaphore byte must be programmed in parameter IVAR2 that can be used by FC 21 to determine the validity or consistency of a block. This handling must be supported by the NC, i.e., in the part program, by writing or deleting the semaphore byte. The semaphore byte is stored in the same data field as the actual user data.

The semaphore byte is identified by a value between 0 and 1023 in IVAR2.

The PLC reads and describes the semaphore byte via FC 21 in the same call, which should transfer the user data. The PLC programmer only needs to set up a semaphore variable. For access from the NC via the parts program, the semaphore feature must be programmed using individual instructions according to the flow chart shown below. The sequence is different for reading and writing variables.

Only individual variables or ARRAYs can be supported directly by the semaphore technique. Structure transfers must be subdivided into individual jobs. The user must ensure data consistency of this structure by programming a semaphore system.

If IVAR2 is set to -1, data are transferred without a semaphore.

Data exchange with semaphore in PLC (schematic of FC 21)

```
          ┌─────────────┐                      ┌─────────────┐
          │   Funct = 3 │                      │   Funct = 4 │
          │    (read)   │                      │    (write)  │
          └─────────────┘                      └─────────────┘
                 │                                     │
                 ▼                                     ▼
           ╱──────────╲      n                  ╱──────────╲      n
          ╱ Semaphore=1?╲──────┐               ╱ Semaphore=0?╲──────┐
          ╲            ╱       │               ╲            ╱       │
           ╲──────────╱        │                ╲──────────╱        │
                 │ j           │                      │ j           │
                 ▼             ▼                      ▼             ▼
          ┌──────────┐  ┌──────────┐          ┌──────────┐  ┌──────────┐
          │ Transfer │  │ Error=1  │          │ Transfer │  │ Error=1  │
          │data from │  │ErrCode=23│          │data from │  │ErrCode=24│
          │ NC to PLC│  └──────────┘          │ PLC to NC│  └──────────┘
          └──────────┘        │               └──────────┘        │
                 │            │                      │            │
                 ▼            │                      ▼            │
          ┌──────────┐        │               ┌──────────┐        │
          │Semaphore=0│       │               │Semaphore=1│       │
          │Error=0   │        │               │Error=0   │        │
          │ErrCode=0 │        │               │ErrCode=0 │        │
          └──────────┘        │               └──────────┘        │
                 │◄───────────┘                      │◄───────────┘
                 ▼                                     ▼
                ╱ ╲                                   ╱ ╲
               ╱───╲                                 ╱───╲
```
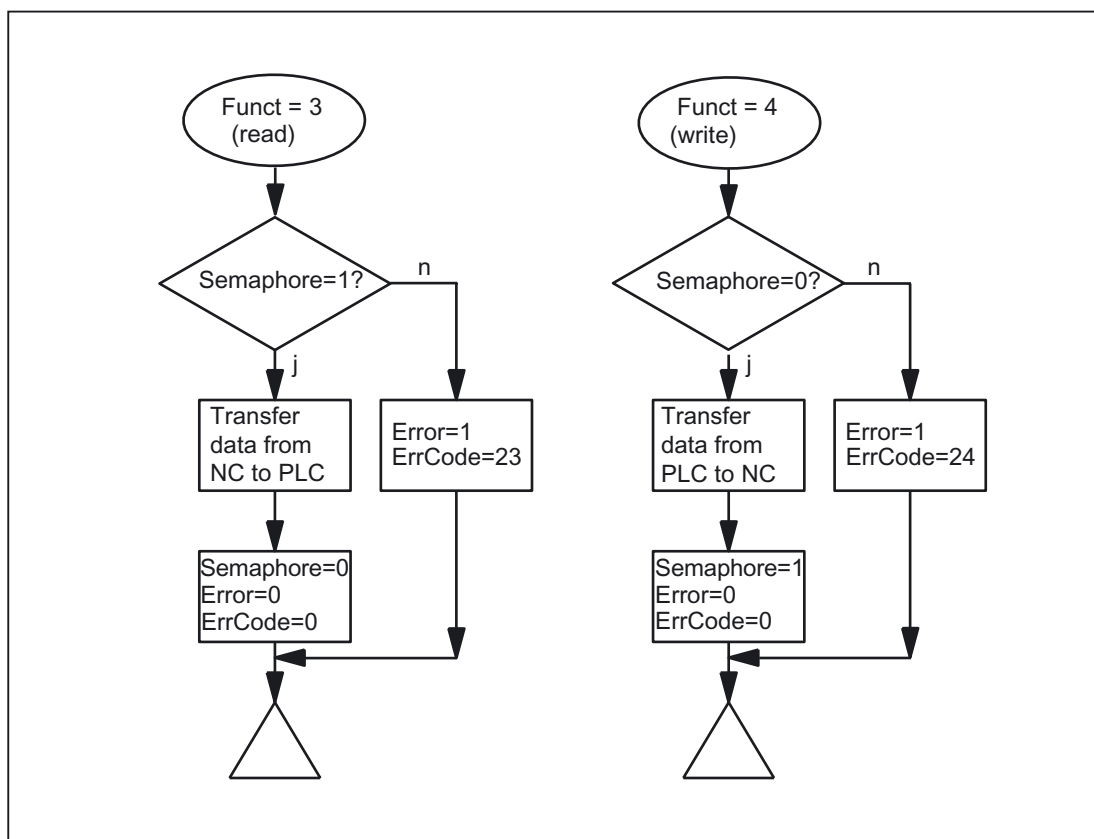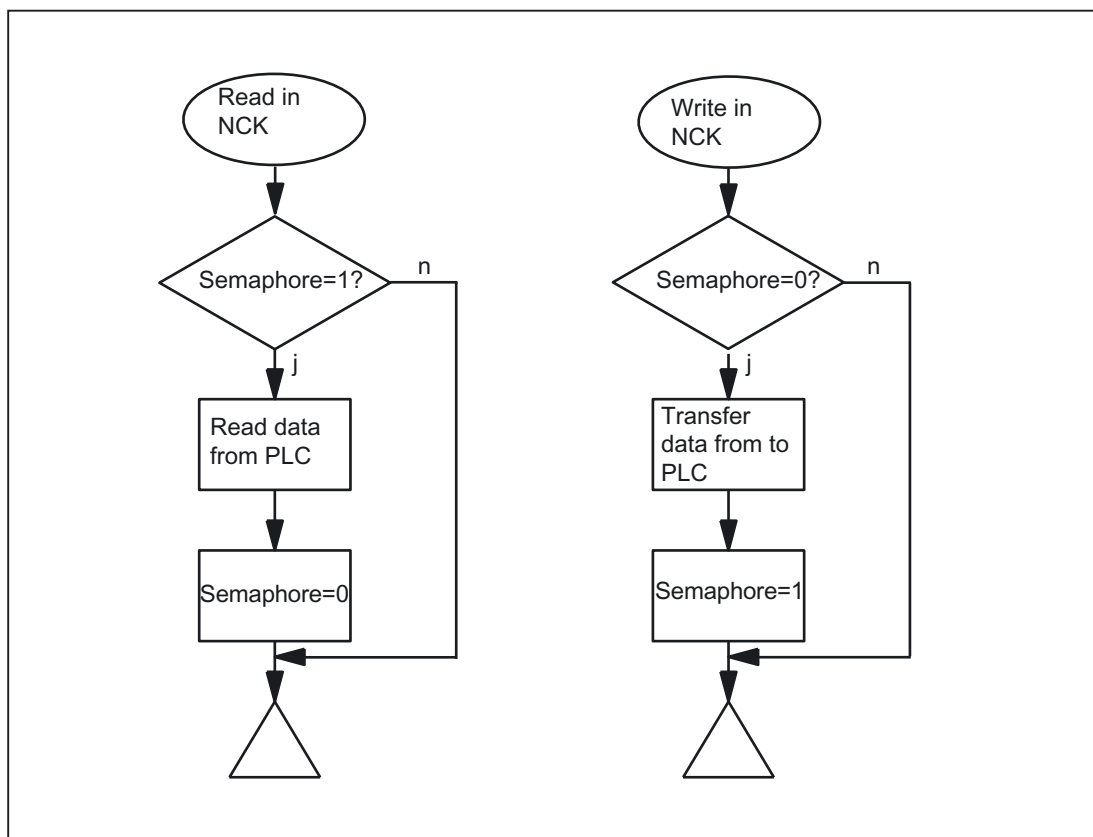
Basic structure in NCK:

## Variable value ranges

The following signals are relevant:

Signal	Type	Type	Value range	Remark
Enable	I	BOOL		= FC 21 active
Funct	I	BYTE	3 ,4	3: Read data 4: Write data
S7Var	I	ANY	S7 data area, except local data	Source/destination data storage area
IVAR1	I	INT	0..1023	Position offset
IVAR2	I	INT	-1 .. 1023	Semaphore byte Transfer without semaphore: -1
Error	A	BOOL		
ErrCode	A	INT		20: Alignment error 21: Illegal position offset 22: Illegal semaphore byte 23: No new data to be read 24: Cannot write data 25: Local data parameterized for S7Var

Call example

1. Read double word of position offset 4 with semaphore in byte 0 and store in MD 100:

Data type Dword (4 bytes)

Position offset 4
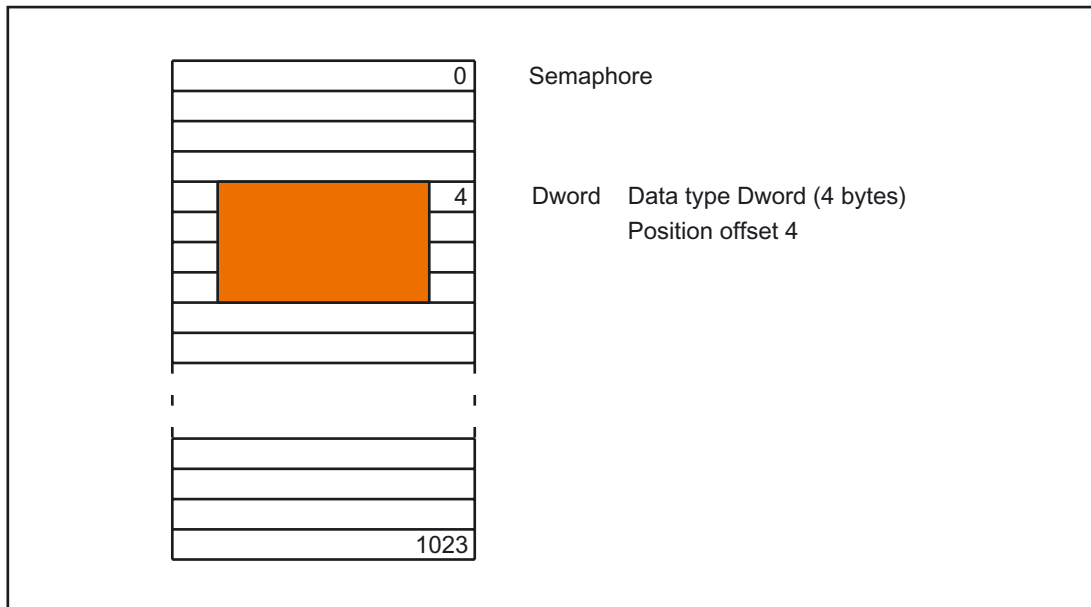

```
CALL FC 21 (
            Enable :=        M 10.0,                    // if TRUE, FC 21 active
            Funct :=         B#16#3,                    //Read data
            S7Var :=         P#M 100.0 DWORD 1,
            IVAR1 :=         4,
            IVAR2 :=         0,
            Error :=         M 10.1,
            ErrCode :=       MW12);
UN M10.1;                            //Enable while 1, until value is read
R F10.0;
```

Examples of NCK programming from synchronized actions:

Data transfer from NC to PLC, with data written via synchronized actions;
Byte 0 serves as the semaphore

ID=1 WHENEVER $A_DBB[0] == 0 DO $A_DBR[4] = $AA_IM[X] $A_DBB[0] = 1

Data transfer from PLC to NC, with data read via synchronized actions;
Byte 1 serves as the semaphore

ID=2 WHENEVER $A_DBB[1] == 1 DO $R1 = $A_DBR[12] $A_DBB[1] = 0

2. Read word of position offset 8 without semaphore and store in MW 104:

```
CALL FC 21 (
            Enable :=        M 10.0,                 // if TRUE, FC 21 active
            Funct :=         B#16#3,                 //Read data
            S7Var :=         P#M 104.0 WORD 1,
            IVAR1 :=         8,
            IVAR2 :=         -1,
            Error :=         M 10.1,
            ErrCode :=       MW12);
```

Function

5: Update control signals to channel:

The purpose of function 5 is to transmit important control signals at high speed in between cyclic data transfers. Data bytes 6 and 7 of user interfaces DB21 to DB30 are transferred to the NC. The channel is specified in parameter "IVAR1". This function can be used, for example, to transfer a feed stop, read-in disable outside the PLC cycle.

The following signals are relevant:

Signal	Type	Type	Value range	Remark
Enable	I	BOOL		1= FC 21 active
Funct	I	BYTE	5	5: Control signals to channel
S7Var	I	ANY	S7 data storage area	Not used
IVAR1	I	INT	1. Max. channel	Channel number
Error	A	BOOL		
ErrCode	A	INT		1: "Funct" invalid 10: Channel no. invalid

6: Update control signals to axes:

The purpose of function 6 is to transmit important control signals at high speed in between cyclic data transfers. The **data byte 2** of application interface DB31 to DB61 is transferred to the NC. The transfer is performed for all activated axes. This allows the servo enable to be transferred outside the PLC cycle, for example.

The following signals are relevant:

Signal	Type	Type	Value range	Remark
Enable	I	BOOL		1= FC 21 active
Funct	I	BYTE	6	6: Control signals to axes
S7Var	I	ANY	S7 data storage area	Not used
IVAR1	I	INT	0	
Error	A	BOOL		
ErrCode	A	INT		1: "Funct" invalid

7: Update control signals to axes:

The purpose of function 7 is to transmit important control signals at high speed in between cyclic data transfers. The **data byte 4** of application interface DB31 to DB61 is transferred to the NC. The transfer is performed for all activated axes. This function can be used, for example, to transfer a feed stop outside the PLC cycle.

The following signals are relevant:

Signal	Type	Type	Value range	Remark
Enable	I	BOOL		1= FC 21 active
Funct	I	BYTE	7	7: Control signals to axes
S7Var	I	ANY	S7 data storage area	Not used
IVAR1	I	INT	0	
Error	A	BOOL		
ErrCode	A	INT		1: "Funct" invalid

2.12.25   FC 22: TM_DIR Direction selection for tool management

Description of Functions

The block TM_DIR provides the shortest path for positioning a magazine or a revolver based on the actual and setpoint position.

As long as a 1 signal is applied to the **Start** input, all output parameters are updated cyclically. Changes can be made to input parameters (e.g., position values) in subsequent PLC cycles.
The output signals are undefined when the start signal is at 0 level.

In the case of direction selection with special positioning input "Offset" > 0, a new setpoint position is calculated from the setpoint and special positions and the number of magazine locations according to the following formula:

New setpoint position = (setpoint pos. - (special pos. -1)) neg. modulo # locations

The new setpoint position corresponds to the location number at which the magazine must be positioned so that the setpoint position requested by the user corresponds to the location number of the special position. The directional optimization is active both with and without special positioning.

The block must be called once with the appropriate parameter settings for each magazine.

⚠️ **Warning**

The block may be called only in conjunction with tool management or after a DB 74 data block has been set up as described below in the example. In this example there are two magazines. The first magazine has 10 locations and the second has 12 locations. When adapting to the real machine, you only need to change the data AnzMag, MagNo[.].

```
DATA_BLOCK DB 74
STRUCT
    P: ARRAY [1..9] OF      DINT;
    w1:                     WORD ;
    AnzMag:                 BYTE ;
    Res.                    BYTE ;


        MagNo: array [1..16] of struct          //Byte 40
        AnzPlatz :      INT ;
        res1 :          BYTE ;
        res2 :          BYTE ;
    end_struct;
end_struct
BEGIN
    P[4]:=L#320;                            //Absolutely essential
    AnzMag:=b#16#2;                         //Total number of magazines = 2
    MagNo[1].AnzPlatz:=10;                  //Number of locations for magazine 1 = 10
```

MagNo[2].AnzPlatz:=12;   //Number of locations for magazine 2 = 12

```
END_DATA_BLOCK
```

Note

For further details on tool management (also with regard to PLC) refer to the Description of Functions Tool Management. Furthermore, PI services are provided for tool management via the FB 4, FC 7 and FC 8 (see also the corresponding Sections in this documentation).

Declaration of the function

STL representation

```
FUNCTION FC 22: VOID
// NAME:              TM_DIR
VAR_INPUT
    MagNo:            INT ;
    ReqPos:           INT ;
    ActPos:           INT ;
    Offset:           BYTE ;
    Start:            BOOL ;
END_VAR

VAR_OUTPUT
    Cw:               BOOL ;
    Ccw:              BOOL ;
    InPos:            BOOL ;
    Diff:             INT ;
    Error :           BOOL ;
END_VAR
BEGIN
END_FUNCTION
```

Description of the formal parameters

The table below shows all formal parameters of the "TM_DIR" function.

Signal	I/O	Type	Value range	Remark
MagNo	I	INT	1..	Magazine number
ReqPos	I	INT	1..	Setpoint location
ActPos	I	INT	1..	Actual location
Offset	I	BYTE	0..	Offset for special positioning
Start	I	BOOL		Start of calculation
Cw	Q	BOOL		1 = Move magazine clockwise
Ccw	Q	BOOL		1 = Move magazine counterclockwise
InPos	Q	BOOL		1 = In position
Diff	Q	INT	0..	Differential path (shortest path)
Error	Q	BOOL		1 = error

Call example

```
CALL FC 22 (              //Tool management direction selection
        MagNo :=    2,                  //Magazine number
        ReqPos :=   mw 20,              //Target position
        ActPos :=   mw 22,              //Current position
        Offset :=   b#16#0,            //Offset for special positioning
        Start :=    m 30.4,            //Start trigger
                                        //Return parameters
        Cw :=       m 30.0,           //Move magazine
                                        //in clockwise direction
        Ccw :=      m 30.1,           //Move magazine
                                        //in anticlockwise direction
        InPos :=    m 30.2,           //Magazine in position
        Diff :=     mw 32,             //Differential path
        Error :=    m30.3              //Error has occurred
        );
```

2.12.26   FC 24: MCP_IFM2 Transmission of MCP signals to interface

Description of Functions

With FC MCP_IFM2 (M variant slim operator panel), the following are transferred from the machine control panel (MCP) to the corresponding signals of the NCK/PLC interface:

- Mode groups

- Axis selections

- WCS/MCS switchover

- Traversing keys

- Overrides or override simulation signals

In the basic program (FC 2), FC 27 transmits handwheel selections, modes and other operation signals from the operator panel or MMC to the NCK/PLC interface in such a way as to allow the modes to be selected from the machine control panel or the operator panel.

Transfer of MMC signals to the interface can be deactivated by setting the value of the parameter "MMCToIF" to "FALSE" in FB 1 (DB 7). "MMCToIF" can also be activated/deactivated in the cyclical program by setting and resetting (e.g., R gp_par.MMCToIF).

The following specifications apply to the **feed override, axis travel keys** and **INC keys** depending on the active mode or on the coordinate system selected:

- **Feed override:**

  – The feed override is transferred to the interface of the selected channel and to the interface of the axes.

  – The feed override signals are transferred in addition to interface byte "Rapid traverse" override (DBB 5) to the NC channel if MMC signal "Feed override for rapid traverse effective" is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this MMC signal.

- **Machine functions for INC and axis travel keys:**

  – When the MCS is selected, the signals are transferred to the interface of the selected machine axis.

  – When the WCS is selected, the signals are transferred to the geoaxis interface of the parameterized channel.

  – When the system is switched between MCS and WCS, the active axes are generally deselected.

The **handwheel selection signals from the MMC** are decoded and activated in the machine-axis or the Geo-axis interface of the handwheel selected (only if parameter "HWheelMMC := TRUE" in FB 1).

The associated LEDs of the machine control panel are derived from the acknowledgments from the relevant selections.

Feedrate and spindle Start/Stop are not transferred to the interface, but output modally as a "FeedHold" or "SpindleHold" signal. The user can link these signals to other signals leading to a feed or spindle stop (this can be implemented, e.g., using the appropriate input signals in FC 10: AL_MSG). The associated LEDs are activated at the same time.

The **spindle direction** (+, -) is not switched directly either, but made available as output parameter "SpindleDir", permitting, for example, FC 18 to be parameterized. A spindle enable signal is also switched via parameter "SpindleHold". One possible method of moving a spindle directly is to preselect it as an axis so that it can be traversed via (axis) direction keys.

If the machine control panel fails, the signals it outputs are preset to zero; this also applies to "FeedHold" and "SpindleHold" output signals.

Multiple calls of FC 24 or FC 19, FC 25, FC 26 are permitted in a single PLC cycle. In this case, the first call in the cycle drives the LED displays. Furthermore, all actions of the parameterized block are carried out in the first call. In the following calls, only a reduced level of processing of the channel and mode group interface takes place. The geometry axes are supplied with directional data only in the first block call in the cycle.

Single block processing can be selected/deselected only in the first call in the cycle.

The second machine control panel can be processed if parameter "BAGNo" has been increased by B#16#10. When parameterizing, the HHU number is contained in the lower nibble (lower 4 bits).
"BAGNo" = 0 or B#16#10 means that the mode group signals are not processed.
ChanNo = 0 means that the channel signals are not processed.

The INC selections are transferred to the mode group interface. This results in runtime improvements. This specification is activated via the DB10 ... DBX 57.0 (INC-inputs active in the BAG-range) through this block once after power up.

Machine control panels can still be handled in parallel by this module. The module 2 call for the 2nd machine control panel in OB1 cycle must come after the call of the 1st A support for 2 MSTT is still provided within certain restrictions in the control panel blocks (support is not provided for standard axis numbers 10 to 31, mutual interlocking of axis selections with 2 MSTT).

Flexible axis configuration

With SW V6 and higher, it is possible to be flexible in the assignment of axis selections or direction keys of machine axis numbers.

Better support is now provided by the MSTT blocks for use of two MCPs, which are operated simultaneously, in particular for an application using two channels and two mode groups. Note that the axis-numbers are also specified in the parameterized mode group number of the MCP block in the axis tables of the relevant MCP.

For this flexibility there are tables for axis numbers in DB 10.
For **1.** Machine control panel (MCP) the table starts from the byte 8 (symbolic name: MCP1AxisTbl[1..22]) and
for **2.** Machine control panel (MCP) the table starts from the byte 32 (symbolic name: MCP2AxisTbl[1..22]) for the second MCP. The machine axis numbers must be entered byte by byte here.

It is permissible to enter a value of 0 in the axis table. Checks are not made to find impermissible axis numbers, meaning that false entries can lead to a PLC Stop.

For **FC 24**, the **maximum possible number of axis selections** can also be restricted. This upper limit is set for the

1. Machine control panel in DB10.DBW30 (symbolic name: MCP1MaxAxis) or for the
2. Machine Control Panel in DB10.DBW54 (symbolic name: MCP2MaxAxis) for the respective MSTT.

The default setting is 0, corresponding to the maximum number of configured axes. The axis numbers and the limit can also be adapted dynamically. Afterwards, a new axis must be selected on FC 19. Axis numbers may not be switched over while the axes are traversing the relevant direction keys.
The compatibility mode is preset with axis numbers **1 to 6** for both MCPs and the restriction for the configured number of axes.

Declaration of the function

```
FUNCTION FC 24: VOID
// NAME:                            MCP_IFM2


VAR_INPUT
     BAGNo :                        BYTE ;
     ChanNo:                        BYTE ;
     SpindleIFNo:                   BYTE ;
END_VAR


VAR_OUTPUT
     FeedHold :                     BOOL ;
     SpindleHold :                  BOOL ;
     SpindleDir:                    BOOL ;
END_VAR


BEGIN
END_FUNCTION
```

Description of the formal parameters

The table below shows all formal parameters of the "MCP_IFM2" function:

Signal	I/O	Type	Value range	Remark
BAGNo	I	BYTE	0 - b#16#0A and b#16#10 - b#16#1A	No. of mode group to which the mode signals are transferred. ModeGroupNo >= b#16#10 means access to the second machine control panel.
ChanNo	I	BYTE	0 - B#16#0A	Channel no. for the channel signals
SpindleIFNo	I	BYTE	0 - 31 (B#16#1F)	Number of the axis interface declared as a spindle
FeedHold	Q	BOOL		Feed stop from machine control panel, modal
SpindleHold	A	BOOL		Spindle stop from machine control panel, modal
SpindleDir	Q	BOOL		Spindle direction 0 equals + (counterclockwise) 1 equals - (clockwise)

Call example

`CALL FC 24 (`	//Slim machine control panel M variants	
	//signals to interface	
BAGNo :=	B#16#1,	//Mode group no. 1
ChanNo :=	B#16#1,	//Channel no. 1
SpindleIFNo :=	B#16#4,	//Spindle interface number = 4
FeedHold :=	m22.0,	//Feed stop signal modal
SpindleHold :=	db2.dbx151.0);	//Spindle stop modal in message data block
SpindleDir:=	m22.1);	//Spindle direction return

With these parameter settings, the signals are sent to the 1st mode group, the 1st channel and all axes. In addition, the spindle override is transferred to the 4th axis/spindle interface. The feed hold signal is passed to bit memory 22.0 and the spindle stop signal to data block DB2, data bit 151.0. The spindle direction feedback signal supplied via parameter SpindleDir can be used as a direction input for an additional FC 18 call.

2.12.27 FC 25: MCP_IFT transfer of MCP/OP signals to interface

Description of Functions

With FC MCP_IFT (T variants), the following are transferred from the machine control panel (MCP) to the corresponding signals of the NCK/PLC interface:

- Mode groups
- Direction keys of four axes
- WCS/MCS switchover commands
- Overrides
- Key switch

In the basic program (FC 2), FC 27 transmits handwheel selections, modes and other operation signals from the operator panel or MMC to the NCK/PLC interface in such a way as to allow the modes to be selected from the machine control panel or the operator panel.

Transfer of MMC signals to the interface can be deactivated by setting the value of the parameter "MMCToIF" to "FALSE" in FB 1 (DB 7).

The following specifications apply to the **feed override, axis travel keys** and **INC keys** depending on the active mode or on the coordinate system selected:

- **Feed override:**
  - The feed override is transferred to the interface of the selected channel and to the interface of the axes.
  - The feed override signals are transferred in addition to interface byte "Rapid traverse" override (DBB 5) to the NC channel if MMC signal "Feed override for rapid traverse effective" is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this MMC signal.

- **Machine functions for INC and axis travel keys:**
  - When the MCS is selected, the signals are transferred to the interface of the selected machine axis.
  - When the WCS is selected, the signals are transferred to the geoaxis interface of the parameterized channel.

The **handwheel selection signals from the MMC** are decoded and activated in the machine-axis or the Geo-axis interface of the handwheel selected (only if parameter "HWheelMMC := TRUE" in FB 1).

The associated LEDs of the machine control panel are derived from the acknowledgments from the relevant selections.

Feedrate and spindle Start/Stop are not transferred to the interface, but output modally as a "FeedHold" or "SpindleHold" signal. The user can link these signals to other signals leading to a feed or spindle stop (this can be implemented, e.g., using the appropriate input signals in FC 10: AL_MSG). The associated LEDs are activated at the same time.

If the machine control panel fails, the signals it outputs are preset to zero; this also applies to "FeedHold" and "SpindleHold" output signals.

Multiple calls of FC 25 or FC 19, FC 24, FC 26 are permitted in a single PLC cycle. In this case, the first call in the cycle drives the LED displays. Furthermore, all actions of the parameterized block are carried out in the first call. In the following calls, only a reduced level of processing of the channel and mode group interface takes place. The geometry axes are supplied with directional data only in the first block call in the cycle.

Single block processing can be selected/deselected only in the first cycle.

The second machine control panel can be processed, when the parameter

"BAGNo" is incremented by B#16#10. When parameterizing, the HHU number is contained in the lower nibble (lower 4 bits).
"BAGNo" = 0 or B#16#10 means that the mode group signals are not processed.
ChanNo = 0 means that the channel signals are not processed.

Flexible axis configuration

With SW V6 and higher, it is possible to be flexible in the assignment of axis selections or direction keys of machine axis numbers.

Better support is now provided by the MSTT blocks for use of two MCPs, which are operated simultaneously, in particular for an application using two channels and two mode groups. The module call for the 2nd machine control panel in OB1 cycle must come after the call of the 1st Note that the axis-numbers are also specified in the parameterized mode group number of the MCP block in the axis tables of the relevant MCP.

For this flexibility there are tables for axis numbers in DB 10.
For **1.** Machine control panel (MCP) the table starts from the byte 8
(symbolic name: MCP1AxisTbl[1..22]) and
for **2.** Machine control panel (MCP) starts from the byte 32
(symbolic name: MCP2AxisTbl[1..22]) for the second MCP. The machine axis numbers must be entered byte by byte here.
It is permissible to enter a value of 0 in the axis table. Checks are not made to find impermissible axis numbers, meaning that false entries can lead to a PLC Stop.

The restriction of the **possible axis selections in FC 25** is done via the 0-values in the axis table. The axis numbers can also be adapted dynamically. Axis numbers may not be switched over while the axes are being traversed via the relevant direction keys.
The compatibility mode is preset with axis numbers **1 to 4** for both MCPs and the restriction for the configured number of axes.

Note

For supplementary information see functions description of FC 19.

Declaration of the function

```
FUNCTION FC 25: VOID
// NAME:                            MCP_IFT


VAR_INPUT
    BAGNo :                         BYTE ;
    ChanNo:                         BYTE ;
    SpindleIFNo:                    BYTE ;
END_VAR


VAR_OUTPUT
    FeedHold :                      BOOL ;
    SpindleHold :                   BOOL ;
END_VAR


BEGIN
END_FUNCTION
```

Description of the formal parameters

The table below shows all formal parameters of the "MCP_IFT" function:

Signal	Type	Type	Value range	Remark
BAGNo	I	BYTE	0 - b#16#0A and b#16#10 - b#16#1A	No. of mode group to which the mode signals are transferred. BAGNo >= b#16#10 means access to the second machine control panel.
ChanNo	I	BYTE	0 - B#16#0A	Channel no. for the channel signals
SpindleIFNo	I	BYTE	0 - 31 (B#16#1F)	Number of the axis interface declared as a spindle
FeedHold	A	BOOL		Feed stop from machine control panel, modal
SpindleHold	A	BOOL		Spindle stop from machine control panel, modal

Call example

`CALL FC 25 (`	//Machine control panel T variants	
	//signals to interface	
BAGNo :=	B#16#1,	//Mode group no. 1
ChanNo :=	B#16#1,	//Channel no. 1
SpindleIFNo :=	B#16#4,	//Spindle interface number = 4
FeedHold :=	m22.0,	//Feed stop signal modal
SpindleHold :=	db2.dbx151.0);	//Spindle stop modal in message data block

With these parameter settings, the signals are sent to the 1st mode group, the 1st channel and all axes. In addition, the spindle override is transferred to the 4th axis/spindle interface. The feed hold signal is passed to bit memory 22.0 and the spindle stop signal to data block DB2, data bit 151.0.

2.12.28   FC 26: HPU_MCP Transfer of HPU/HT6 signals to the interface

2.12.28.1   FC 26: HPU_MCP Transfer of HPU/HT6 signals to the interface

Description of Functions

With FC HPU_MCP (machine control panel signals of the handheld terminal), the following are transferred from the machine control panel (MCP) to the corresponding signals of the NCK/PLC interface:

- Mode groups
- WCS/MCS switchover
- Traversing keys
- Override

In the basic program (FC 2), FC 27 transmits handwheel selections, modes and other operation signals from the operator panel or MMC to the NCK/PLC interface in such a way as to allow the modes to be selected from the machine control panel or the operator panel.

Transfer of MMC signals to the interface can be deactivated by setting the value of the parameter "MMCToIF" to "FALSE" in FB 1 (DB 7).

The following specifications apply to the **feed override, axis travel keys** and **INC keys** depending on the active mode or on the coordinate system selected:

- **Feed override:**

  – The feed override is transferred to the interface of the selected channel and to the interface of the axes.

  – The feed override signals are transferred in addition to interface byte "Rapid traverse" override (DBB 5) to the NC channel if MMC signal "Feed override for rapid traverse effective" is set (exception: Switch setting "Zero"). "Rapid traverse override effective" is also set with this MMC signal.

  Machine functions for INC and axis travel keys:

  – When the MCS is selected, the signals are transferred to the interface of the selected machine axis (for 6 axes).

  – When the WCS is selected, the signals of the first three axes are transferred to the Geo axis interface of the parameterized channel. The remaining three axes are transferred to the interface of the selected machine axis.

The **handwheel selection signals from the MMC** are decoded and activated in the machine-axis or the Geo-axis interface of the handwheel selected (only if parameter "HWheelMMC := TRUE" in FB 1).

The LEDs on the machine control panel derived from the selections in the acknowledgment.

Feedrate and spindle Start/Stop are not transferred to the interface, but output modally as a "FeedHold" or "SpindleHold" signal. The user can link these signals to other signals leading to a feed or spindle stop (this can be implemented, e.g., using the appropriate input signals in FC 10: AL_MSG). The associated LEDs are activated at the same time.

In the case of machine control panel failure, the signals it supplies are set to zero.

Multiple calls of FC 26 or FC 19, FC 24, FC 25 are permitted in a single PLC cycle. In this case, the first call in the cycle drives the LED displays. Moreover, all actions of the parameterized block are performed in the first call. In the following calls, only a reduced level of processing of the channel and mode group interface takes place. The geometry axes are supplied with directional data only in the first block call in the cycle.

Single block processing can be selected/deselected only in the first cycle.

The second machine control panel can be processed if parameter "BAGNo" has been increased by B#16#10. When parameterizing, the HHU number is contained in the lower nibble (lower 4 bits).
"BAGNo" = 0 or B#16#10 means that the mode group signals are not processed.
ChanNo = 0 means that the channel signals are not processed.

The INC selections are transferred to the mode group interface. This results in runtime improvements. This specification is activated via the DB10 ... DBX 57.0 takes place through this block once after power up. Furthermore, two machine control panels can be handled in parallel by this block. The module 2 call for the 2nd machine control panel in OB1 cycle must come after the call of the 1st A support for 2 MSTT is still provided within certain restrictions in the control panel blocks (support is not provided for standard axis numbers 10 to 31, mutual interlocking of axis selections with 2 MSTT).

Flexible axis configuration

It is possible to be flexible in the assignment of axis selections or direction keys for machine axis numbers.

Better support is now provided by the MSTT blocks for use of two MCPs, which are operated simultaneously, in particular for an application using two channels and two mode groups. Note that the axis-numbers are also specified in the parameterized mode group number of the MCP block in the axis tables of the relevant MCP.

For this flexibility there are tables for axis numbers in DB 10.
For **1.** Machine control panel (MCP) the table starts from the byte 8
(symbolic name: MCP1AxisTbl[1..22]) and
for **2.** Machine control panel (MCP) the table starts from the byte 32
(symbolic name: MCP2AxisTbl[1..22]) for the second MCP. The machine axis numbers must be entered byte by byte here.

It is permissible to enter a value of 0 in the axis table. No check is made for illegal axis numbers and an incorrect entry can thus lead to a PLC stop.
With **FC 26**, it is possible to restrict the number of axes using 0 values in the axis table. The axis numbers can also be adapted dynamically. Axis numbers may not be switched over while the axes are being traversed via the relevant direction keys.

The compatibility mode is preset with axis numbers **1 to 6** for both MCPs and restricted to the configured number of axes.

Note

With FC26 (PHG and HT6) there is a deviation from other MCP blocks
FC 19, FC 24, FC 25 in the selected workpiece coordination system (WCS).

Here the first 3 traversing keys have an effect on the channel axes (WCS axes). The following traversing keys affect the machine axes as per the axis numbers allocated from the axis tables. These are the machine axes 4 to 6 by default. Wiring of output byte 3, bit 7 to 1 on the MCP outputs (parameter "MCP1Out" or "MCP2OUT") permits only the channel axes to be traversed when WCS is activated.

Declaration of the function

```
FUNCTION FC 26: void
// NAME:                          HPU_MCP
VAR_INPUT
     BAGNo :                      BYTE ;
     ChanNo:                      BYTE ;
END_VAR
BEGIN
END_FUNCTION
```

Description of the formal parameters

The table below shows all formal parameters of the "HPU_MCP" function.

Signal	Type	Type	Value range	Remark
BAGNo	I	BYTE	0 - b#16#0A and b#16#10 - b#16#1A	No. of mode group to which the mode signals are transferred. BAGNo >= b#16#10 means access to the second machine control panel.
ChanNo	I	BYTE	0 - B#16#0A	Channel no. for the channel signals

2.12.28.2 MCP selection signals to the user interface

Operating modes and machine functions

Source: MCP key	Destination: Interface DB (parameter ModeGroupNo) representation for mode group 1
AUTOMATIC	DB11, ... DBX0.0
MDA	DB11, ... DBX0.1
JOG	DB11, ... DBX0.2
REPOS	DB11, ... DBX1.1
REF	DB11, ... DBX1.2
TEACH IN	DB11, ... DBX1.0
INC 1 ... 10 000, INC Var. (starting from SW 5)	DB11, ... DBB2, Bit 0 to 5

Source: MCP key	Destination: Interface DB (parameter ChanNo)
INC 1 ... 10 000, INC Var. (till SW 4)	DB21, ... DBB13, Bit 0 to 5
INC 1 ... 10 000, INC Var. (till SW 4)	DB21, ... DBB17, Bit 0 to 5
INC 1 ... 10 000, INC Var. (till SW 4)	DB21, ... DBB21, Bit 0 to 5

Source: MCP key	Destination: Interface DB (6-axis DB)
INC 1 ... 10 000, INC Var. (till SW 4)	DB31, ... DBB5, Bit 0 to 5

Direction keys rapid traverse override

The transfer is dependent upon the selected axis. The corresponding interface bits are deleted for axes that are not selected.

Source: MCP button	Destination: Interface DB (Parameter ChanNo)
Direction key +	DB21, ... DBX12.7
Direction key -	DB21, ... DBX12.6
Rapid traverse override	DB21, ... DBX12.5
Direction key +	DB21, ... DBX16.7
Direction key -	DB21, ... DBX16.6
Rapid traverse override	DB21, ... DBX16.5
Direction key +	DB21, ... DBX20.7
Direction key -	DB21, ... DBX20.6
Rapid traverse override	DB21, ... DBX20.5

Source: MCP key	Destination: Interface DB (6 allocated axis DBs)
Direction key +	DB31, ... DBX4.7
Direction key -	DB31, ... DBX4.6
Rapid traverse override	DB31, ... DBX4.5

Override

Source: MCP - Setting	Destination: Interface DB (parameter ChanNo)
Feedrate override	DB21, ... DBB4

Source: MCP - Setting	Destination: Interface DB (6 allocated axis DBs)
Feedrate override	DB31, ... DBB0 (selected axis number) The feed override of the 1st MCP applies to all axes.

Channel signals

Source: MCP - keys	Destination: Interface DB (parameter ChanNo)
NC Start	DB21, ... DBX7.1
NC stop	DB21, ... DBX7.3
RESET	DB21, ... DBX7.7
Single block	DB21, ... DBX0.4

2.12.28.3 Checkback signals from user interface for controlling displays

Operating modes and machine functions

Destination: MCP - Output	Source: Interface DB (parameter ModeGroupNo) representation for mode group 1
AUTOMATIC	DB11, ... DBX7.0
MDA	DB11, ... DBX6.1
JOG	DB11, ... DBX6.2
REPOS	DB11, ... DBX7.1
REF	DB11, ... DBX7.2
TEACH IN	DB11, ... DBX7.0

WCS/MCS - output is operated through key actuation.

Call example

```
CALL FC 26 (                          //Machine control panel of the HPU/HT6
                                      //signals to interface
            BAGNo :=      B#16#1,     //Mode group no. 1
            ChanNo :=     B#16#1);    //Channel no. 1
```

With these parameter settings, the signals from the first parameterized machine control panel are sent to the 1st mode group, the 1st channel and all axes.

2.12.29   FC 19, FC 24, FC 25, FC 26 source code description

Task

Machine control panel on user interface
(FC19 M variant, FC24 slim-line variant, FC25 T variant, FC26 PHG/HT6 variant).

Associated blocks

DB 7, no. of BAGs, channels, axes

DB 7, pointer of machine control panel

DB 8, storage for the next cycle

FC 20, output of error messages

Resources used

None.

General

Blocks FC 19 (M version), FC 24 (slim-line version), FC 25 (T version) and FC 26 (PHG/HT6 version) transfer the signals of the machine control panel to and from the application interface. In the input parameters, "ModeGroupNo" selects the mode group to be processed by the block. The "ModeGroupNo" parameter also selects the number of the machine control panel (bit 4). "ChanNo" selects the channel to be processed. The "SpindleIFNo" parameter defines the axis interface of the spindle. The spindle override is transferred to this spindle interface. The input parameters are checked for incorrect parameterization.

Output parameters "FeedHold" and "SpindleHold" are generated from the 4 feed/spindle disable and feed/spindle enable keys and are returned with "logical 1" for disable. Information for the next cycle is stored in DB8, bytes 0 to 3 or bytes 62 to 65, depending on the machine control panel number. This information is edge trigger flag, feedrate value and the selected axis number. The blocks are provided with user data via the pointer parameters in DB 7 "MCP1In" and "MCP1Out" ("MCP2In" and "MCP2Out"). The pointers are addressed indirectly via a further pointer from the VAR section of DB7 in order to avoid absolute addressing. This additional pointer is determined symbolically in FB1.

Block Description

All 4 blocks are similar in structure and are structured according to subtasks:

In the input network
various parameters are copied into local variables. The machine control signals (user data for input/output area) are also copied between locations using the various pointers in DB 7 (gp_par). These local variables are handled in the block for reasons of efficiency. Some values are initialized for the startup.

The
MCS/WCS switchover with edge evaluation, axis selections, direction keys and rapid traverse overlay are determined in Network Global_in for further processing in the block. The user-specific changes must take place in this part of the program, will usually involve axis selection.

Only the keyswitch information is copied in Network NC.

The mode group network
transfers the modes of the keys as dynamic signals to the NCK.
If the mode group number is 0, this network is not processed. A too large BAG number produces a message 401901 or 402501 and is switched after stop.

In the Channel network
the NC Start, Stop, Reset and Single Block functions are activated by corresponding checkback signals. The direction keys of the geometry axes are supplied if a corresponding preselection is made, otherwise they are cleared. If the channel number is 0, this network is not processed. A too large BAG number produces a message 401902 or 402502 and is switched after stop.

The spindle network
transfers the spindle override into the interface configured via SpindleIFNo.

The axis network
transfers the feedrate override to the selected axis interface. The direction keys are assigned to the selected axis/spindle. If an axis has been selected previously, the direction information is set to 0. The INC checkback of the selected axis is displayed.

The output parameters are prepared and the LED signals of the INC machine function are generated in the
global_out network.

The output network
transfers the output signals of the machine control panel from the VAR_TEMP image to the logical address. The data for the next cycle are also saved.

Axis selection extension

Network Global_in
must be modified for more than 9 axes. If other keys and LEDs are to be used on the machine control panel here, proceed as follows:

1. The UD DW#16#Wert command deletes all LEDs defined for axis selections.
   The bit mask is currently processing the 9 axis selection LED.

2. The UW W#16# command, with the comment "Mask all axis selection keys" checks for a new selection of direction. The bit string must be adjusted here.

3. The branch destination list (SPL) must be expanded with new jump labels.
   The new jump labels should be inserted in descending order before label m009. The selection information should be extended for the new jump labels, as described for labels m009 and m008.

2.13 Signal/data descriptions

2.13.1 Interface signals NCK/PLC, MMC/PLC, MCP/PLC

General

The NCK/PLC, HMI/PLC, and MSTT/PLC interface signals are listed in the Parameter Manual for SINUMERIK 840D with references to the respective function description where the signals are described.

References:
/LIS2/ Lists (Volume 2)

The NCK signals that are evaluated by the basic program and transferred in conditioned form to the user interface are presented in the following sections.

2.13.2 Decoded M signals

General

The M functions programmed in the part program, ASUB or synchronized actions are channel specifically transferred from the NC to the PLC:

- M functions from channel 1: DB21
- M functions from channel 2: DB22
- etc.

The signal length is one PLC cycle.

Note

The spindle-specific M functions below are not decoded: M3, M4, M5, and M70.

Addresses in DB21, ...	Variable	Type	Comment
DBX 194.0 ... 7	M_Fkt_M0 ... M7	Bool	M signals M0 ... M7
DBX 195.0 ... 7	M_Fkt_M8 ... M15	Bool	M signals M8 ... M15
DBX 196.0 ... 7	M_Fkt_M16 ... M23	Bool	M signals M16 ... M23
DBX 197.0 ... 7	M_Fkt_M24 ... M31	Bool	M signals M24 ... M31
DBX 198.0 ... 7	M_Fkt_M32 ... M39	Bool	M signals M32 ... M39
DBX 199.0 ... 7	M_Fkt_M40 ... M47	Bool	M signals M40 ... M47
DBX 200.0 ... 7	M_Fkt_M48 ... M55	Bool	M signals M48 ... M55
DBX 201.0 ... 7	M_Fkt_M56 ... M63	Bool	M signals M56 ... M63
DBX 202.0 ... 7	M_Fkt_M64 ... M71	Bool	M signals M64 ... M71
DBX 203.0 ... 7	M_Fkt_M72 ... M79	Bool	M signals M72 ... M79
DBX 204.0 ... 7	M_Fkt_M80 ... M87	Bool	M signals M80 ... M87
DBX 205.0 ... 7	M_Fkt_M88 ... M95	Bool	M signals M88 ... M95
DBX 206.0 ... 3	M_Fkt_M96 ... M99	Bool	M signals M96 ... M99

Note

The M02/M30 auxiliary function output to the PLC does not state that the part program has been terminated. In order to securely identify the end of a part program in the channel, DB21, ... DBX33.5 (M02/M30 active)
must be evaluated. The channel status must be RESET. The auxiliary function output could arise from an asynchronous subroutine (ASUB) or a synchronized action and has nothing to do with the real end of the parts program in this case.

2.13.3   G Functions

General

The G functions programmed in the part program, ASUB or synchronized actions are channel specifically transferred from the NC to the PLC:

- G functions from channel 1: DB21

- G functions from channel 2: DB22

- etc.

The signal length is one PLC cycle.

POWER ON
After POWER ON, the value zero, i.e., active G groups undefined, is specified in the NC/PLC interface for all G groups.

Part program-End or Abort
After part program end or abort, the last state of the G group is retained.

NC-START
After NC-START the values specified in the machine data:
MD22510 $NC_ GCODE_GROUPS_TO_PLC
specified 8 G-groups corresponding to the basic setting done via the machine data, as well as the values programmed in the part program, are overwritten.

Addresses in DB21, ...	Variables	Type	Basic position	Comment
DBB 208	G_FKT_GR_1	BYTE	0	Active G function of group 1
DBB 209	G_FKT_GR_2	BYTE	0	Active G function of group 2
DBB 210	G_FKT_GR_3	BYTE	0	Active G function of group 3
DBB 211	G_FKT_GR_4	BYTE	0	Active G function of group 4
DBB 212	G_FKT_GR_5	BYTE	0	Active G function of group 5
DBB 213	G_FKT_GR_6	BYTE	0	Active G function of group 6
DBB 214	G_FKT_GR_7	BYTE	0	Active G function of group 7
DBB 215	G_FKT_GR_8	BYTE	0	Active G function of group 8
DBB 216	G_FKT_GR_9	BYTE	0	Active G function of group 9
DBB 217	G_FKT_GR_10	BYTE	0	Active G function of group 10
DBB 218	G_FKT_GR_11	BYTE	0	Active G function of group 11
DBB 219	G_FKT_GR_12	BYTE	0	Active G function of group 12
DBB 220	G_FKT_GR_13	BYTE	0	Active G function of group 13
DBB 221	G_FKT_GR_14	BYTE	0	Active G function of group 14
DBB 222	G_FKT_GR_15	BYTE	0	Active G function of group 15
DBB 223	G_FKT_GR_16	BYTE	0	Active G function of group 16
DBB 224	G_FKT_GR_17	BYTE	0	Active G function of group 17
DBB 225	G_FKT_GR_18	BYTE	0	Active G function of group 18
DBB 226	G_FKT_GR_19	BYTE	0	Active G function of group 19
DBB 227	G_FKT_GR_20	BYTE	0	Active G function of group 20
DBB 228	G_FKT_GR_21	BYTE	0	Active G function of group 21

Addresses in DB21, ...	Variables	Type	Basic position	Comment
DBB 229	G_FKT_GR_22	BYTE	0	Active G function of group 22
DBB 230	G_FKT_GR_23	BYTE	0	Active G function of group 23
DBB 231	G_FKT_GR_24	BYTE	0	Active G function of group 24
DBB 232	G_FKT_GR_25	BYTE	0	Active G function of group 25
DBB 233	G_FKT_GR_26	BYTE	0	Active G function of group 26
DBB 234	G_FKT_GR_27	BYTE	0	Active G function of group 27
DBB 235	G_FKT_GR_28	BYTE	0	Active G function of group 28
DBB 236	G_FKT_GR_29	BYTE	0	Active G function of group 29
DBB 237	G_FKT_GR_30	BYTE	0	Active G function of group 30
DBB 238	G_FKT_GR_31	BYTE	0	Active G function of group 31
DBB 239	G_FKT_GR_32	BYTE	0	Active G function of group 32
DBB 240	G_FKT_GR_33	BYTE	0	Active G function of group 33
DBB 241	G_FKT_GR_34	BYTE	0	Active G function of group 34
DBB 242	G_FKT_GR_35	BYTE	0	Active G function of group 35
DBB 243	G_FKT_GR_36	BYTE	0	Active G function of group 36
DBB 244	G_FKT_GR_37	BYTE	0	Active G function of group 37
DBB 245	G_FKT_GR_38	BYTE	0	Active G function of group 38
DBB 246	G_FKT_GR_39	BYTE	0	Active G function of group 39
...	. . .	...	...	...
DBB 271	G_FKT_GR_64	BYTE	0	Active G function of group 64

G functions (values)

A complete listing of all the G functions is given in:
References:
/PG/ Programming Manual, Fundamentals; "List of G Functions/Preparatory functions"

2.13.4   Message signals in DB 2

General

DB2 allows the user to display the messages for individual signals on the operator panel front. As the lists of interface signals show, signals are divided into predefined groups. In the event of incoming and outgoing messages and message acknowledgements, the number entered in the message number column is transferred to the PCU. Text can be stored for each message number.
References:
/IAD/Installation and Start-Up Guide; Message Numbers

Note

The number of user areas can be parameterized via FB1.

After the configuration has been modified (FB1: MsgUser) DB2/DB3 must be deleted.

Channel areas in DB2

Area	Address	Message number
Channel 1	DBX0.0 - DBX11.7	510.000 -- 510.231
Channel 1, geo axes	DBX12.0 - DBX17.7	511.100 - 511.315
Channel 2	DBX18.0 - DBX29.7	520.000 - 520.231
Channel 2, geo axes	DBX30.0 - DBX35.7	521.100 - 521.315
Channel 3	DBX36.0 - DBX47.7	530.000 - 530.231
Channel 3, geo axes	DBX48.0 - DBX53.7	531.000 - 531.315
Channel 4	DBX54.0 - DBX65.7	540.000 - 540.231
Channel 4, geo axes	DBX66.0 - DBX71.7	541.100 - 541.315
Channel 5	DBX72.0 - DBX83.7	550.000 - 550.231
Channel 5, geo axes	DBX84.0 - DBX89.7	551.100 - 551.315
Channel 6	DBX90.0 - DBX101.7	560.000 - 560.231
Channel 6, geo axes	DBX102.0 - DBX107.7	561.100 - 561.315
Channel 7	DBX108.0 - DBX119.7	570.000 - 570.231
Channel 7, geo axes	DBX120.0 - DBX125.7	570.100 - 571.315
Channel 8	DBX126.0 - DBX137.7	580.000 - 580.231
Channel 8, geo axes	DBX138.0 - DBX143.7	581.100 - 581.315
Channels 9 and 10 are not currently implemented		

User areas in DB2

Area	Address	Message number
Axis/spindle 1	DBX144.0 - DBX145.7	600.100 - 600.115
Axis/spindle 2	DBX146.0 - DBX147.7	600.200 - 600.215
Axis/spindle 3	DBX148.0 - DBX149.7	600.300 - 600.315
Axis/spindle 4	DBX150.0 - DBX151.7	600.400 - 600.415
Axis/spindle 5	DBX152.0 - DBX153.7	600.500 - 600.515
Axis/spindle 6	DBX154.0 - DBX155.7	600.600 - 600.615
Axis/spindle 7	DBX156.0 - DBX157.7	600.700 - 600.715
Axis/spindle 8	DBX158.0 - DBX159.7	600.800 - 600.815
Axis/spindle 9	DBX160.0 - DBX161.7	600.900 - 600.915
Axis/spindle 10	DBX162.0 - DBX163.7	601.000 - 601.015
Axis/spindle 11	DBX164.0 - DBX165.7	601.100 - 601.115
Axis/spindle 12	DBX166.0 - DBX167.7	601.200 - 601.215
Axis/spindle 13	DBX168.0 - DBX169.7	601.300 - 601.315
Axis/spindle 14	DBX170.0 - DBX171.7	601.400 - 601.415
Axis/spindle 15	DBX172.0 - DBX173.7	601.500 - 601.515
Axis/spindle 16	DBX174.0-DBX175.7	601.600 - 601.615
Axis/spindle 17	DBX176.0 - DBX177.7	601.700 - 601.715
Axis/spindle 18	DBX178.0 - DBX179.7	601.800 - 601.815
Axes 19 to 31 are not currently implemented		

User areas in DB2

Area	Address	Message number
User area 0	DBX180.0 - DBX187.7	700.000 - 700.063
User area 1	DBX188.0 - DBX195.7	700.100 - 700.163
User area 2	DBX196.0 - DBX203.7	700.200 - 700.263
User area 3	DBX204.0 - DBX211.7	700.300 - 700.363
User area 4	DBX212.0 - DBX219.7	700.400 - 700.463
User area 5	DBX220.0 - DBX227.7	700.500 - 700.563
User area 6	DBX228.0 - DBX235.7	700.600 - 700.663
User area 7	DBX236.0 - DBX243.7	700.700 - 700.763
User area 8	DBX244.0 - DBX251.7	700.800 - 700.863
User area 9	DBX252.0 - DBX259.7	700.900 - 700.963
User area 10	DBX260.0 - DBX267.7	710.000 - 701.063
User area 11	DBX268.0 - DBX275.7	710.100 - 701.163
User area 12	DBX276.0 - DBX283.7	710.200 - 701.263
User area 13	DBX284.0 - DBX291.7	710.300 - 701.363
User area 14	DBX292.0 - DBX299.7	710.400 - 701.463
User area 15	DBX300.0 - DBX307.7	710.500 - 701.563
User area 16	DBX308.0 - DBX315.7	710.600 - 701.663
User area 17	DBX316.0 - DBX323.7	710.700 - 701.763
User area 18	DBX324.0 - DBX331.7	710.800 - 701.863
User area 19	DBX332.0 - DBX339.7	710.900 - 701.963
User area 20	DBX340.0 - DBX347.7	702.000 - 702.063
User area 21	DBX348.0 - DBX355.7	702.100 - 702.163
User area 22	DBX356.0 - DBX363.7	702.200 - 702.263
User area 23	DBX364.0 - DBX371.7	702.300 - 702.363
User area 24	DBX372.0 - DBX379.7	702.400 - 702.463

2.14 Programming tips with STEP 7

2.14.1 General

General

Some useful tips on programming complex machining sequences in STEP7 are given below. This information concentrates mainly on the handling of data type POINTER and ANY. Detailed information about the structure of data types POINTER and ANY can be found in Chapter "CPU register and storage of data" in STEP7 manual "Designing user programs".

2.14.2 Copying data

For the high-speed copying of data from one DB into another it is recommended

- for larger data quantities to use the system function SFC BLKMOV or SFC FILL, because here a high-speed copying takes place.

- the routine given below is for smaller data quantities, because the supply of ANY parameter to the SFCs consumes additional time.

The following is an example of how to copy data at high speed from one DB into another.

Code			Comment
			// DB xx.[AR1] is the source
			// DI yy.[AR2] is the destination
OPEN	DB 100;		//Source DB
LAR1	P#20.0;		//Source start address on data byte 20
OPEN	DI 101;		//Destination DB
LAR2	P#50.0;		//Destination start address on data byte 50
			//AR1, AR2, DB, DI loaded beforehand
L	4;		//Transfer 8 bytes
M001:			
L	DBW [AR1,P#0.0];		//Copy word-oriented
T	DIW [AR2,P#0.0];		
+AR1	P#2.0;		
+AR2	P#2.0;		
TAK;			
LOOP	M001;		

2.14.3 ANY and POINTER

2.14.3.1 POINTER or ANY variable for transfer to FC or FB

With version 1 or later of STEP7 it is possible to define a POINTER or an ANY in VAR_TEMP. The following two examples show how an ANY can be supplied.

Example 1 Transfer ANY parameter via a selection list to another FB (FC)

Several ANY parameters are defined in an FB (FC). A specific ANY parameter must now be chosen from a selection list for transfer to another FB (FC). This can only be done by means of an ANY in VAR_TEMP. 1 to 4 can be set in parameter "WhichAny" in order to select Addr1 to Addr4.

Note

Address register AR2 is used in the block. However, this address register AR2 is also used for multiinstance DBs. For this reason, this FB should **not** be declared as multi-instance DB.

FUNCTIONBLOCK FB 100	Comment
`CODE_VERSION1`	//starting from STEP 7 Version 2 for deactivating the multi-instance DB
`VAR_INPUT`	
`WhichAny : INT ;`	
`    Addr1 : ANY ;`	//Observe predetermined order
`    Addr2 : ANY ;`	
`    Addr3 : ANY ;`	
`    Addr4 : ANY ;`	
`END_VAR`	
`VAR_TEMP`	
`    dbchr : WORD ;`	
`    Number: WORD ;`	
`    type : BYTE ;`	
`    Temp_addr : ANY ;`	
`END_VAR`	
`BEGIN`	
`NETWORK`	
`TITLE =`	
`L          WhichAny;`	
`DEC 1;`	
`L          P#10.0;`	//10 bytes per ANY
`*I;`	
`LAR2;`	

FUNCTIONBLOCK FB 100	Comment
L P##Addr1;	
+AR2;	//Add ANY start addresses
L P##Temp_addr;	
LAR1 ;	//Retrieve pointer from VAR_TEMP
L DID [AR2,P#0.0];	//Transfer pointer value to VAR_TEM
T LD [AR1,P#0.0];	
L DID [AR2,P#4.0];	
T LD [AR1,P#4.0];	
L DIW [AR2,P#8.0];	
T LW [AR1,P#8.0];	
CALL FB 101, DB 100	
(ANYPAR := #Temp_addr);	//ANYPAR is data type ANY

Example 2 transfer an ANY parameter constructed earlier to another FB (FC)

An ANY parameter that has already been compiled must be transferred to another FB (FC). This can be done only by means of an ANY stored in VAR_TEMP

FUNCTIONBLOCK FB 100	Comment
VAR_INPUT	
DBNumber: INT ;	
DBOffset : INT ;	
Data type: INT ;	
Number: INT ;	
END_VAR	
VAR_TEMP	
dbchr : WORD ;	
Temp_addr : ANY ;	
END_VAR	
BEGIN	
NETWORK	
TITLE =	
L P##Temp_addr;	
LAR1 ;	//Retrieve pointer from VAR_TEMP
L B#16#10;	//ANY identifier
T LB [AR1,P#0.0];	
L Data type;	
T LB [AR1,P#1.0];	
L Amount;	
T LW [AR1,P#2.0];	
L DBNumber;	
T LW [AR1,P#4.0];	
L DBOffset;	

FUNCTIONBLOCK FB 100	Comment
SLD 3;	//Offset is a bit offset
T LD [AR1,P#6.0];	
CALL FB 101, DB 100	
(ANYPAR := #Temp_addr);	//ANYPAR is data type ANY

2.14.3.2 General

General

The following programming examples illustrate different programming mechanisms. They demonstrate how input/output and transit variables (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT) are accessed by data types "POINTER" or "ANY" within an FC or FB. The access operations are described in such a way that a part symbolic method of programming can be used.

2.14.3.3 Use of POINTER and ANY in FC if POINTER or ANY is available as parameter

Description of Functions

FC 99 has inputs parameters that are defined as POINTER or ANY.

The example shows a body program via which the subcomponents of the POINTER or ANY can be accessed. In this case, the DB parameterized with POINTER or ANY is opened and the address offset stored as a crossarea pointer in address register AR1, thus allowing access to data elements of variables (generally structures and arrays) that are addressed via the POINTER, ANY.

This access operation is described at the end of the relevant program sequence in the example. With data type ANY, it is also possible to execute a check or branch when the variable is accessed based on the data type and the number of elements involved.

Example in FC, if POINTER or ANY are present as parameters

FUNCTION FC 99: VOID		Comment
VAR_INPUT		
Row : BYTE ;		
Convert : BOOL ;		//Activate numerical conversion
Addr: POINTER;		//Points to variable
Addr1 : ANY ;		
END_VAR		
VAR_TEMP		
dbchr : WORD ;		
Number: WORD ;		
type : BYTE ;		
END_VAR		
BEGIN		
NETWORK		
TITLE =		
		//POINTER
L	P##Addr;	
LAR1 ;		//Retrieve pointer
L	W [AR1,P#0.0];	//Retrieve DB number
T	#dbchr;	
L	D [AR1,P#2.0];	//Offset part of pointer
LAR1 ;		
AUF DB [#dbchr];		//Open DB of variables
L B [AR1,P#40.0];		//Retrieve byte value using pointer with
		//address offset 40
		//ANY
L	P##Addr1;	
LAR1 ;		//Retrieve ANY
L	B [AR1,P#1.0];	//Retrieve type
T	#typ;	
L	W [AR1,P#2.0];	//Retrieve amount
T	#Amount;	
L	W [AR1,P#4.0];	//Retrieve DB number
T	#dbchr;	
L	D [AR1,P#6.0];	//Offset part of pointer
LAR1 ;		
OPEN	DB [#dbchr];	//Open DB of variables
L	B [AR1,P#0.0];	//Retrieve byte value using ANY

2.14.3.4   Use of POINTER and ANY in FB if POINTER or ANY is available as parameter

Description of Functions

FB 99 has inputs parameters that are defined as POINTER or ANY.

The example shows a body program via which the subcomponents of the POINTER or ANY can be accessed. In this case, the DB parameterized with POINTER or ANY is opened and the address offset stored as a crossarea pointer in address register AR1, thus allowing access to data elements of variables (generally structures and arrays) that are addressed via the POINTER, ANY.

This access operation is described at the end of the relevant program sequence in the example. With data type ANY, it is also possible to execute a check or branch when the variable is accessed based on the data type and the number of elements involved.

Example in FB, if POINTER or ANY are present as parameters

FUNCTIONBLOCK FB 99		Comment
VAR_INPUT		
Row : BYTE ;		
Convert : BOOL ;	//Activate numerical conversion	
Addr: POINTER;	//Points to variable	
Addr1 : ANY ;		
END_VAR		
VAR_TEMP		
dbchr : WORD ;		
Number: WORD ;		
type : BYTE ;		
END_VAR		
BEGIN		
NETWORK		
TITLE =		
		//POINTER
L	P##Addr;	
LAR1 ;		//Retrieve pointer from instance DB
L	DIW [AR1,P#0.0];	//Retrieve DB number
T	#dbchr;	
L	DID [AR1,P#2.0];	//Offset part of pointer
LAR1 ;		
OPEN	DB [#dbchr];	//Open DB of variables
L	B [AR1,P#40.0];	//Retrieve byte value using pointer with
//address offset 40		
		//ANY
L	P##Addr1;	
LAR1 ;	//Retrieve ANY from instance DB	//Retrieve ANY from instance DB
L	DIB [AR1,P#1.0];	//Retrieve type

FUNCTIONBLOCK FB 99		Comment
T	#typ;	
L	DIW [AR1,P#2.0];	//Retrieve amount
T	#Amount;	
L	DIW [AR1,P#4.0];	//Retrieve DB number
T	#dbchr;	
L	DID [AR1,P#6.0];	//Offset part of pointer
LAR1 ;		
OPEN	DB [#dbchr];	//Open DB of variables
L	B [AR1,P#0.0];	//Retrieve byte value using ANY

2.14.4 Multiinstance DB

With version 7 and higher of STEP 7, FBs might have a multiinstance capability, i.e., they might incorporate multiinstance DBs. The primary characteristic of multiinstance DBs is that a block can be used for various instances of FBs (see STEP 7 documentation). As such, the quantity structure of the DBs can be optimized.

Multi-instance DBs should be activated only when they are actually going to be used since they increase the runtime and code size of the FBs.

Note

When complex programs are implemented in multiinstance FBs that use a pointer and address register, it is important for the programmer to observe certain rules.
With multiinstance DBs, the start address of the variable (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR) is transferred with the DI data block register and address register AR2. When variables are accessed within the multiinstance FB, the compiler independently controls the access operation via address register AR2. However, when complex program sections also have to work with address registers in the same FB (e.g., to copy data), then the old contents of AR2 must be saved before the register is changed. The contents of AR2 must be restored to their original state before an instance variable (VAR_INPUT, VAR_OUTPUT, VAR_IN_OUT, VAR) is accessed. The AR2 register of the instance is to be saved most usefully in a local variable (VAR_TEMP).
The command "Load pointer to an instance variable" returns a pointer value from the start of the instance data. To be able to access this variable via a pointer, the offset stored in AR2 must be added.

Example

FUNCTION_BLOCK FB 99		Comment
VAR_INPUT		
varin: INT ;		
END_VAR		
VAR		
variable1: ARRAY[0 to 9] of INT;		
variable2: INT ;		
END_VAR		
BEGIN		
L	P##variable1;	//Pointer at start of ARRAY
		//The value 8500 0010 is now in the accumulator
		//and a cross-area pointer is in the AR2. If cross-area processing is to take place, then an area should be skipped when these two pointers are added.
AD	DW#16#00FF_FFFF,	//Skipping of an area
LAR1		//Load into AR1
TAR2;		
+AR1 AR2;		//AR2 instance offset to be added
		//The ARRAY of variable1 can now be accessed indirectly via AR1.
L	DIW [AR1, P#0.0];	//E.g., access to first element
END_FUNCTION_BLOCK		

2.14.5   Strings

General

The STRING data type is required by certain services of the basic program. For this reason, some additional facts about the string structure and general handling procedures for parameter assignments are given below.

Structure of STRING

A data of type STRING is generally stored (defined) in a data block. There are two methods of defining a string:

1. Only the data type STRING is assigned to a variable. The STEP7 compiler automatically generates a length of 254 characters.

2. Data type STRING is assigned to a variable together with a string length in square parenthesis (e.g., [32]). With this method, the STEP7 compiler generates a string length corresponding to the input.

Two bytes more than prescribed by the definition are always stored for variables of the STRING data type. The STEP 7 compiler stores the maximum possible number of characters in the 1st byte. The 2nd byte contains the number of characters actually used. Normally, the useful length of the assigned STRINGs is stored by the STEP 7 compiler. The characters (1 byte per character) are then stored from the 3rd byte onwards.

String parameters are generally assigned to blocks of the basic program by means of a POINTER or ANY. Such assignments must generally by made using symbolic programming methods. The data block, which contains the parameterizing string, must be stored in the symbol list. The assignment to the basic program block is then made by means of the symbolic data block name followed by a full stop and the symbolic name of the string variable.

2.14.6   Determining offset addresses for data block structures

General

Another task, which occurs frequently, is symbolic determination of an offset address within a structured DB, e.g., an ARRAY or STRUCTURE is stored somewhere within the DB. After loading the address register symbolically with the start address, you might like to access the individual elements of the ARRAY or STRUCTURE via an address register. One way of loading the address register symbolically is to use an FC whose input parameter is a pointer. The address of the ARRAY or STRUCTURE is then assigned symbolically to the input parameter of this FC in the program. The program code in the FC now determines the offset address from the input parameter, and passes the offset address in the address register (AR1) to the calling function. Symbolic addressing is thus possible even with indirect access.

FUNCTION FC 99: VOID	**Comment**
VAR_INPUT	
Addr: POINTER;	//Points to variable
END_VAR	
BEGIN	
NETWORK	
TITLE =	
L P##Addr;	
LAR1 ;	//Retrieve pointer from Addr
L D [AR1,P#2.0];	//Offset part of pointer of variable
LAR1 ;	
END_FUNCTION	

Supplementary conditions

<div style="text-align: right; font-size: 2em; font-weight: bold;">3</div>

There are no supplementary conditions to note.

Examples

<div style="text-align: right">

4

</div>

No examples are available.

Data lists

5

5.1 Machine data

5.1.1 NC-specific machine data

Number	Identifier: $MN_	Description
10100	PLC_CYCLIC_TIMEOUT	Cyclic PLC monitoring time
14504	MAXNUM_USER_DATA_INT	Number of user data (INT)
14506	MAXNUM_USER_DATA_HEX	Number of user data (HEX)
14508	MAXNUM_USER_DATA_FLOAT	Number of user data (FLOAT)
14510	USER_DATA_INT	User data (INT)
14512	USER_DATA_HEX	User data (HEX)
14514	USER_DATA_FLOAT[n]	User data (FLOAT)

Machine data in integer/hex format is operated in the NC as DWORD.
Machine data in floating comma format are operated in the NC as FLOAT (IEEE 8 byte).
They are stored in the NC/PLC interface and can be read by the PLC user program during PLC power-up from the DB 20.

5.1.2 Channelspecific machine data

Number	Identifier: $MC_	Description
28150	MM_NUM_VDIVAR_ELEMENTS	Number of elements for writing PLC variables

Index