NCT®

Machine Tool Controls

PLC Programmer's Manual

From SW Version x.066 **(M) (L)**

Produced and developed by **NCT Automation kft**.
H1148 Budapest Fogarasi út 7
⊠ Letters: 1631 Bp. P.O. Box 26
☎ Phone: (+36 1) 467 63 00
☎ Fax:(+36 1) 467 63 09
E-mail: nct@nct.hu
Home Page: www.nct.hu

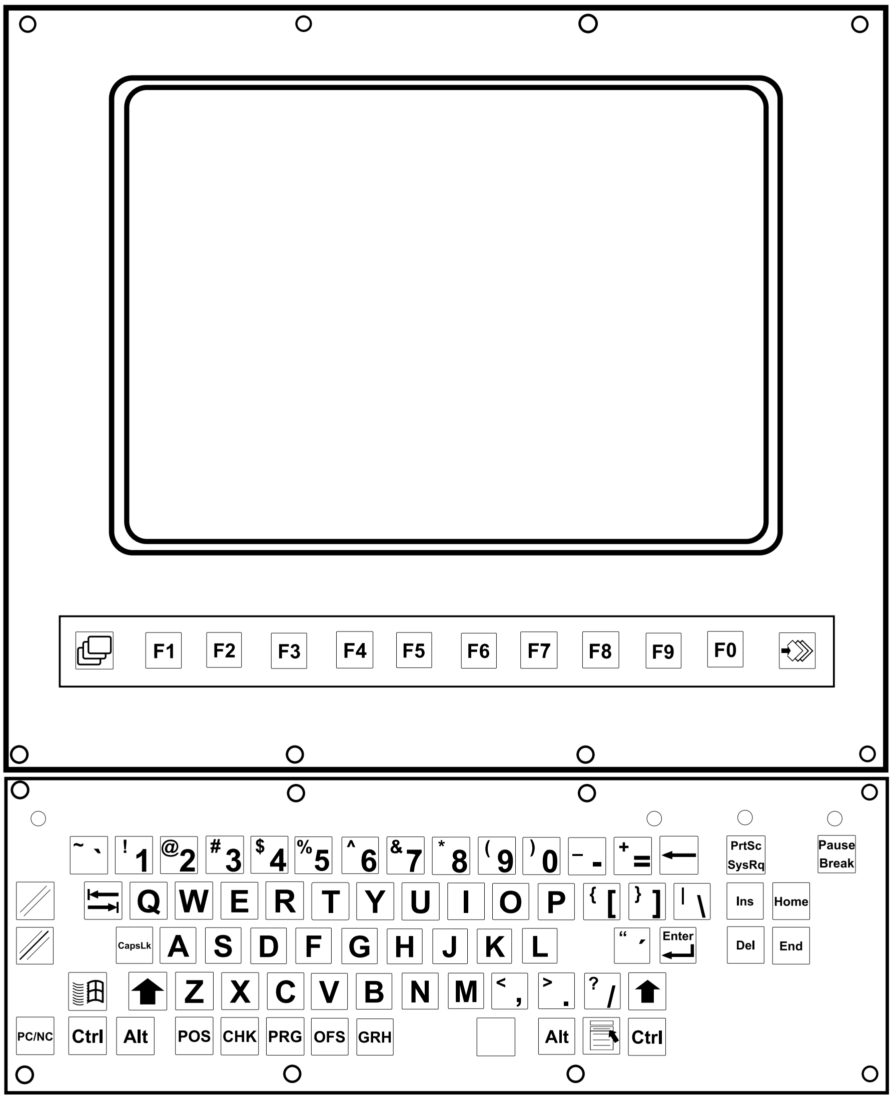
Contents

February 5, 2010

1 General Description

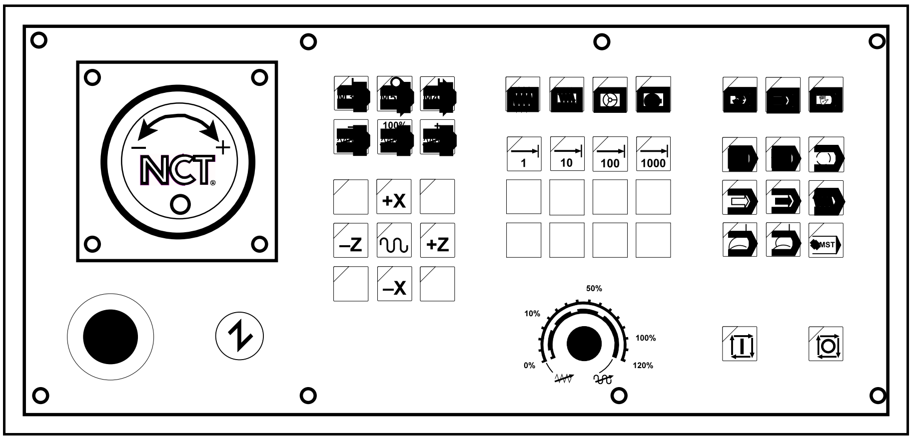
1.1 Fundamental Terms

To clearly understand this handbook some fundamental terms have to be elicited.
Control: The entire device controlling the machine tool, storing the part programs and interpreting them in the course of program execution.
NC: A part of the control, which stores and preprocesses part programs, and transfers their commands to the servos and PLC.
PLC: It interprets commands coming from the NC not referring to servos and sends them to the machine tool.
Operator's panel: It consists of the monitor unit and the keyboard. The keyboard is made up of two parts, of the NC or data input keyboard, which contains editing keys, data input keys and softkeys,

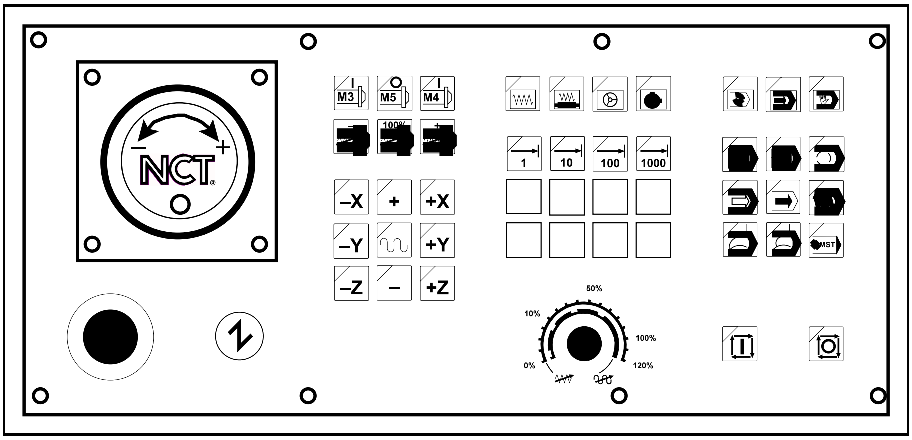


The monitor unit, the data input keyboard and the softkeys

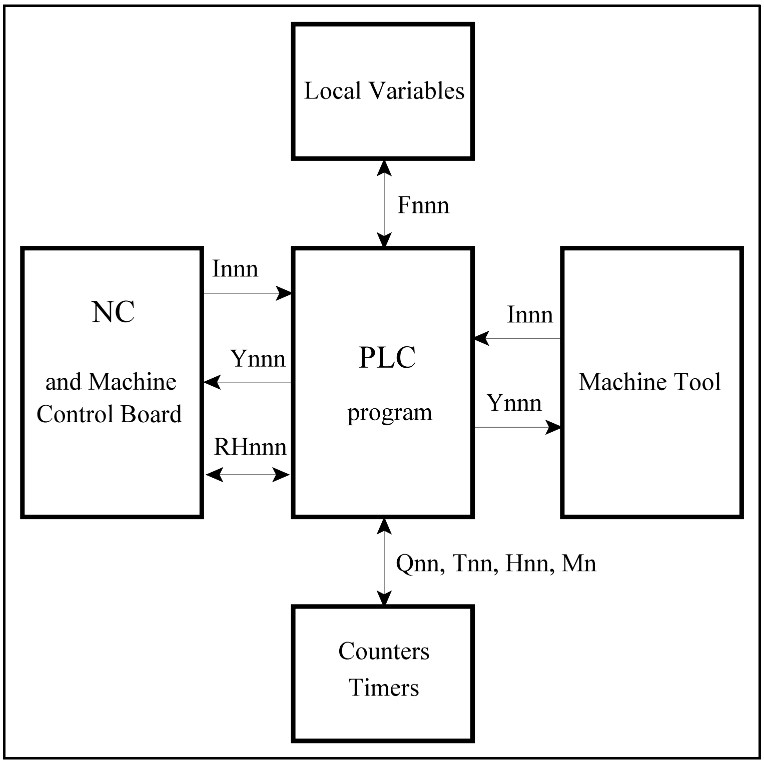
as well as of machine control board, which frames the operation mode push-buttons, the manual movement buttons and other switches, buttons and lamps. The machine control board may be integrated in the control.



Machine Control Board for Turning Machines



Machine Control Board for Milling Machines



Coordination between control and machine tool is done by the PLC. The PLC is one of the programs running in the control, which is connected with:

– the machine tool through the interface board(s) built in the control,

– the machine control board through flags, perhaps interface input lines,

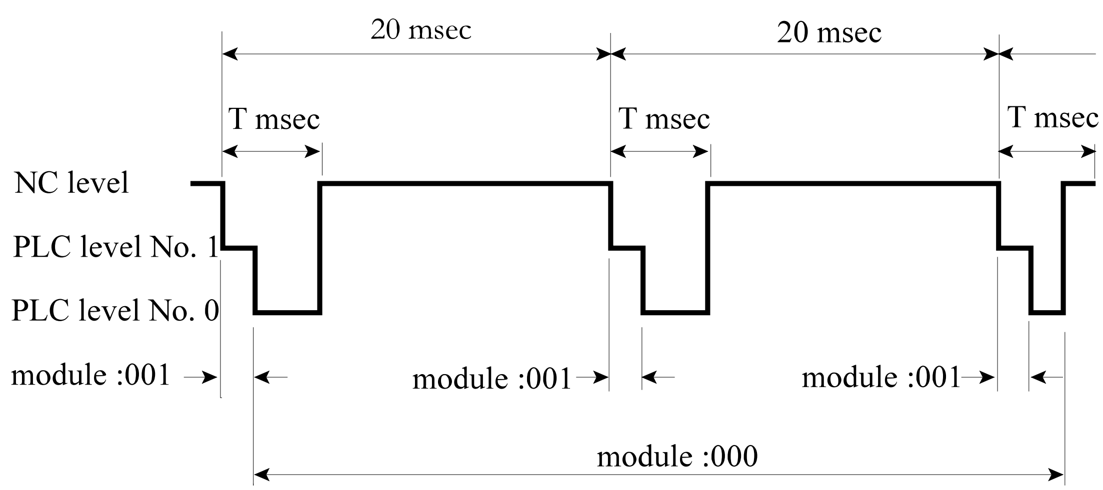
– the NC through input and output flags, as well as registers.

The above mentioned interface input and output lines, as well as input and output flags and registers are variables in the PLC program, the detailed description of whose is discussed in chapter 2.

Besides the memory area, which stores local variables and within this memory area two special tables accessible from control panel, which support tool handling are at disposal in the PLC. Among the local variables of PLC program there are also counters and timers.

1.2 Structure and function of PLC program

The PLC program of NCT controls is written in a special, high-level language developed especially for this task. In this language bit variables (flags) can be switched on and off, as well as condition tests can be done on the variables. The register communication and operations are supported by word (16 bit) value assigning, data transfer, arithmetic, logic and condition test statements. The values of parameters and macro variables in the NC memory can be accessed by special commands. Finally it enables execution of 8, 16, 32 bit, signed, fix-point, binary arithmetical basic operations.
The structure of PLC program is obligatory, so that by executing it cyclically, it should fit the control function to the machine tool. Therefore the PLC program receives from the control a T msec long time slice in every 20 msec, when the PLC activities can be executed.
The activities to be executed by the PLC program can run in two levels (modules) within the T-msec-long PLC time slice. The length of the T msec time slice is different in different types of controls.



Level No. 1, module :001
Level No. 1 is executed from the beginning in every PLC time slice, thus in every 20 msec. The complete execution of this level is mandatory in all PLC time slices. If it does not happen, error message PLC TIMEOUT1 is displayed by the control. The beginning of level No. 1 is indicated by label :001, while its end by statement J1 in the source language text of PLC program.
Level No. 0, module :000
The execution of level No. 0 is done after the execution of PLC program level No. 1 in the part time left from the 5 msec. PLC module level No. 0 is not obligatorily executed within a time slice, it can last for more time slices. In case level No. 0 has been executed, the rest time of the PLC is returned to the NC. The beginning of level No. 0 is indicated by label :000, while its end slice by statement J0 in the source language text of PLC program.
As seen above it is advisable to use module :001 (level No. 1) for supervisory actions. Such actions may be the watching of and reacting on the flag state of alarms, limits, signals coming from reference position switches or operator's interventions, as well as receiving commands sent by the NC in the course of command execution.

Module :000 (Level No. 0) can be used by tasks, the execution of which takes a longer time, as e.g. spindle handling.
Certain commands are disabled in the PLC program level No. 1, yet other ones, the executing time of which is long, are not advisable to use.
In emergency cases there may be need to answer input signals instantly. This can be done with the help of module :002.

Level No. 2, module :002

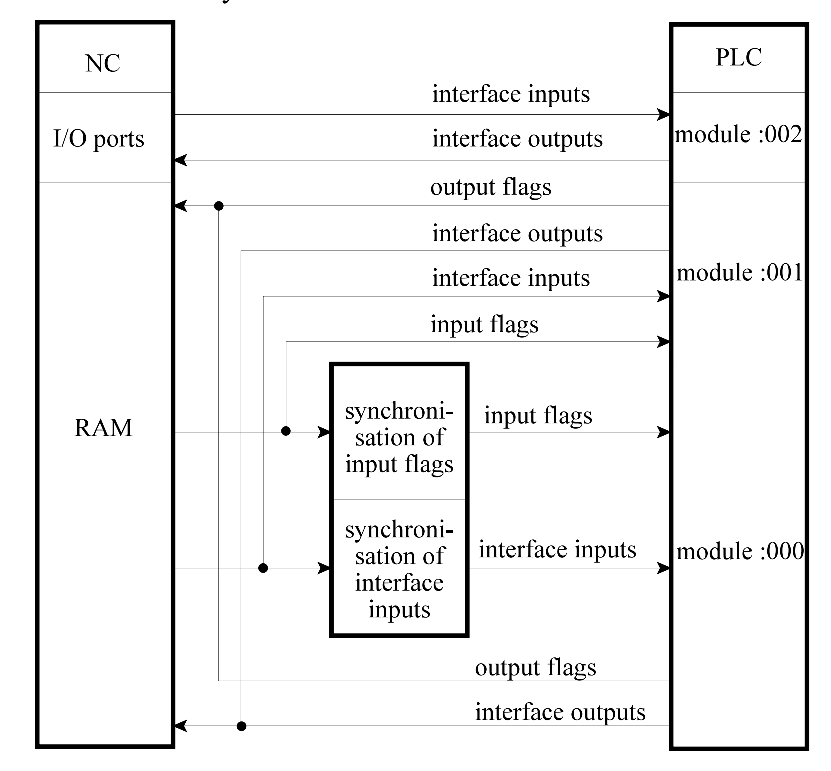
Module :002 is called by the NC in each

t=5 msec (in control types NCT98, NCT99, NCT2000)
t=2 msec (in control types NCT990, NCT100, NCT115)
t=1 msec (in control types NCT101, NCT104)

provided module call is enabled. Module :002 must be short in source code and must be executed as fast as it is possible, otherwise error message PLC TIMEOUT2 is displayed by the NC. The beginning of level No. 2 is indicated by label :002, while its end by statement J2 in the source language text of PLC program. Call of module :002 is enabled or disabled by flag Y546.

1.3 Processing of PLC Input and Output Signals

Generally PLC program handles state of interface I/O lines and I/O flags indirectly, according to their code stored in RAM. State of input lines is updated at the beginning of PLC time slice by directly reading input signals and by storing their state code into RAM. The state of output signals is updated at the end of PLC slice by writing the code of output flags stored in RAM to the output lines. The output lines are connected effectively at this moment.



Th difference between level No. 1 (module :001) and level No. 0 (module :000) is that level No. 0 observes input lines updated in every 20 msec, while module :000 does not. The interface input lines and input flags seem synchronized by the level No. 0. This means, that at the beginning module :000 observes the input RAM code received at the beginning of the time slice till module :000 goes to command J0, even if it takes more time slices. This means, that within one PLC slice the program executed in level No. 1 observes different input states from the ones observed by that executed in level No. 0. The above mentioned synchronizing does not occur in the handling of interface output lines and output flags, therefore output lines switched on or off in a given PLC slice by module :000 are updated at the end of the PLC time slice the same as the ones switched

on or off in level No. 1.
Handling of output lines and input lines by their RAM codes is needed partly to execute PLC programs as fast as possible, partly for synchronizing reasons. The difference between the input RAM codes of levels No. 0 and 1 is only due to synchronizing reasons.
For level No. 2. or module :002 neither output and input updating, nor input synchronizing is done. For handling the most essential output lines and input lines two special commands are found in module :002, with the help of which the input signal(s) of the interface board can be tested directly (command Ppqr), and with which the output signal(s) can be set right away (commands UOpqr, DOpqr). Thus these output lines and input lines are not processed through RAM. This time no synchronizing is implemented. On the other hand the executing time of these commands is five times slower than the commands processed through RAM. Therefore the use of these commands is only advisable in case rapid intervention is needed.

1.4 Synchronizing Functions with Interpolation

A part program may contain:
- only interpolation commands (interpolation block)
- only function commands (function block), and
- miscellaneous commands containing both interpolation and function.

Most of the function blocks, or blocks containing also functions demand PLC actions. Exceptions are the program controlling functions, as e.g. command M99 Pnnnn, which executes subprogram call.
During program processing commands of miscellaneous blocks are sent to interpolator and to the PLC simultaneously. That is the control executes interpolation and function at the same time. The task of PLC programmer is to synchronize the two actions if needed as the function of the structure of the machine and the applied technology.
Let us see an example on the above discussed matters by examining the positioning command G0 and the spindle start and stop as a function beside it.

G0 Xx Yy M3
G0 Xx Yy M4
G0 Xx Yy M5
G0 Xx Yy M19

In the above case spindle rotation switch on or off or spindle orientation can be done parallel to the positioning, i.e. when executing these blocks there is no need for synchronizing.
The situation is different if spindle is switched on parallel to a milling command.

G1 Xx Yy Ff M3
G1 Xx Yy Ff M4

The interpolation cannot be started till the spindle reaches the desired revolution speed, i.e. the interpolation must be synchronized.
If spindle rotation stop or spindle orientation is programmed in a milling block the situation is reversed.

G1 Xx Yy Ff M5
G1 Xx Yy Ff M19

The function, i.e. the spindle stop or spindle orientation must be executed only after the execution of interpolation.
The synchronizing of interpolation and function is supported by output and input flags.

2 PLC Program Variables

Reference can be made to PLC program variables with 1 or 2 characters followed by 2, 3 or 4 digits.

2.1 Variables of Connection between PLC and Machine Tool

The physical connection between the machine tool and the PLC is implemented by the INT (interface) board or boards built in the control. INT boards are capable of receiving or emitting two-state (TRUE=24V/FALSE=0V) and level 24V=.signals.

2.1.1 Signal from Machine to PLC (Interface Input Lines)

Reference can be made to synchronized interface input lines stored in RAM with character I and three digits.

Ipqr

The value range of the first digit:

p=0,1,2,3

The second digit is decimal and its value range is

q=0,1,2,3,4,5,6,7,8,9

The third digit defines the serial number of a bit within the selected byte and is therefore octal. Its value range is

r=0,1,2,3,4,5,6,7

Reference to input lines of INT interface boards

The first digit **(p)** defines the **board**, one the input lines of which is to be referred to. At most 4 INT interface boards can be built in the NCT controls. Therefore reference has to be made to the first board with string I0qr, to the second one with string I1qr, to the third one with string I2qr, while to the fourth one with string I3qr.

p=0,1,2,3

The second digit **(q)** defines the **byte** within the selected board, in which the desired input line can be found. For on a board 48 (56) input lines are available the second digit can alter from 0 to 5 (6).

q=0,1,2,3,4,5,(6)

The third digit **(r)** defines the **bit** within the selected byte. Therefore the values of r may be as follows:

r=0,1,2,3,4,5,6,7

The NCT controls have a 16-bit bus, that is why the interface input flags are updated word by word in the memory from INT boards. This way in the view of signal processing 16 input lines can be regarded as totally simultaneous.

It follows that the second indexes of input lines are regarded as simultaneous:

q=1,0

q=3,2

q=5,4

Reference can be made to certain groups of interface input lines as to word operands. In case of word operands reference is made to input line groups in the PLC program by dropping the last digit:

Ipq

If reference is not to be made to input lines synchronized and stored in RAM, but directly to the state of input lines on interface board, it can be done with the help of statement

Ppqr

in case of a bit operand and with the help of statement

Pqr

in case of a word operand, where interpretation of indexes p, q, r corresponds to that of Ipqr.

In module :001, i.e. on level No. 1 also the change test of input lines is enabled. The change test can be executed with the help of statement

Vpqr

on bit operand, while with the help of statement

Vpq

on word operand, where interpretation of indexes p, q, r corresponds to that of Ipqr.
Result of statement Vpqr is 1 if the value of input line Ipqr of the previous PLC time slice differs from that valid in the current time slice.

1st interface board can be optionally equipped with 4 12-bit AD (analog to digital) converters capable of receiving analog inputs. Their values can be displayed through registers RH035, ..., RH038.

The below table summarizes the correspondence between the input connection points of interface boards and the input lines in the PLC program.

Reference to Input Lines of Connector I1 of INT Interface Boards:

Connection Point	1st INT board	2nd INT board	3rd INT board	4th INT board
35	I000	I100	I200	I300
32	I001	I101	I201	I301
14	I002	I102	I202	I302
13	I003	I103	I203	I303
37	I004	I104	I204	I304
36	I005	I105	I205	I305
18	I006	I106	I206	I306
17	I007	I107	I207	I307
29	I010	I110	I210	I310
28	I011	I111	I211	I311
10	I012	I112	I212	I312
9	I013	I113	I213	I313

Connection Point	1st INT board	2nd INT board	3rd INT board	4th INT board
31	I014	I114	I214	I314
30	I015	I115	I215	I315
12	I016	I116	I216	I316
11	I017	I117	I217	I317
25	I020	I120	I220	I320
24	I021	I121	I221	I321
6	I022	I122	I222	I322
5	I023	I123	I223	I323
27	I024	I124	I224	I324
26	I025	I125	I225	I325
8	I026	I126	I226	I326
7	I027	I127	I227	I327
21	I030	I130	I230	I330
20	I031	I131	I231	I331
2	I032	I132	I232	I332
1	I033	I133	I233	I333
23	I034	I134	I234	I334
22	I035	I135	I235	I335
4	I036	I136	I236	I336
3	I037	I137	I237	I337

Reference to Input Lines of Connector I2 of INT Interface Boards:

Connection Point	1st INT board	2nd INT board	3rd INT board	4th INT board
35	I040	I140	I240	I340
32	I041	I141	I241	I341
14	I042	I142	I242	I342
13	I043	I143	I243	I343
37	I044	I144	I244	I344
36	I045	I145	I245	I345
18	I046	I146	I246	I346
17	I047	I147	I247	I347
29	I050	I150	I250	I350
28	I051	I151	I251	I351
10	I052	I152	I252	I352
9	I053	I153	I253	I353
31	I054	I154	I254	I354
30	I055	I155	I255	I355
12	I056	I156	I256	I356
11	I057	I167	I257	I357
25[1]	I060	I160	I260	I360
24[1]	I061	I161	I261	I361
6[1]	I062	I162	I262	I362
5[1]	I063	I163	I263	I363
27[1]	I064	I164	I264	I364
26[1]	I065	I165	I265	I365
8[1]	I066	I166	I266	I366
7[1]	I067	I167	I267	I367

[1] Available in types NCT2000, 100, 104, NCT115

Connection Point	1st INT board	2nd INT board	3rd INT board	4th INT board
1[2]	A1: RH035			
2[2]	GND1			
3[2]	A2: RH036			
4[2]	GND2			
20[2]	A3: RH037			
21[2]	GND3			
22[2]	A4: RH038			
23[2]	GND4			

[2] Optional in types NCT100, 104, NCT115

2.1.2 Signals from PLC to Machine (Interface Output Lines)

Reference to interface output lines stored in RAM can be made with character Y and three digits:

Ypqr

The value range of the first digit:

p=0,1,2,3

The second digit is decimal and its value range is

q=0,1,2,3,4,5,6,7,8,9

The third digit defines the serial number of a bit within the selected byte and is therefore octal. Its value range is

r=0,1,2,3,4,5,6,7

Reference to output lines of INT interface boards

The first digit **(p)** defines the **board**, one the output lines of which is to be referred to. At most 4 INT interface boards can be built in the NCT controls. Therefore reference has to be made to the first board with string I0qr, to the second one with string Y1qr, to the third one with string Y2qr, while to the fourth one with string Y3qr, so

p=0,1,2,3

The second digit **(q)** defines the **byte** within the selected board, in which the desired output line can be found. For on a board 32 output lines are available the second digit can alter from 0 to 3.

q=0,1,2,3

The third digit **(r)** defines the **bit** of the selected byte. Therefore the values of r may be as follows:

r=0,1,2,3,4,5,6,7

The NCT controls have a 16-bit bus, that is why the interface output lines are updated word by word from the RAM. This way in the view of signal transfer 16 output lines can be regarded as totally simultaneous.

It follows that the second indexes of output flags are regarded as simultaneous:

q=1,0

q=3,2

Reference can be made to certain groups of interface output flags, as to word operands. In case of word operands reference is made to output line groups in the PLC program by dropping the last digit:

Ypq

If reference is not made to output lines via RAM, but the state of output lines is to be changed directly, it can be done with the help of statement

Opqr

in case of a bit operand and with the help of statement

Opq

in case of a word operand. Interpretation of indexes p, q, r corresponds to that of Ypqr.

Reference to Output Lines of Connector O1 of INT Interface Boards:

Connection Point	1st INT board	2nd INT board	3rd INT board	4th INT board
14	Y000	Y100	Y200	Y300
12	Y001	Y101	Y201	Y301
31	Y002	Y102	Y202	Y302
29	Y003	Y103	Y203	Y303
30	Y004	Y104	Y204	Y304
13	Y005	Y105	Y205	Y305
16	Y006	Y106	Y206	Y306
15	Y007	Y107	Y207	Y307
6	Y010	Y110	Y210	Y310
4	Y011	Y111	Y211	Y311
21	Y012	Y112	Y212	Y312
23	Y013	Y113	Y213	Y313
7	Y014	Y114	Y214	Y314
5	Y015	Y115	Y215	Y315
24	Y016	Y116	Y216	Y316
22	Y017	Y117	Y217	Y317
10	Y020	Y120	Y220	Y320
8	Y021	Y121	Y221	Y321
25	Y022	Y122	Y222	Y322
27	Y023	Y123	Y223	Y323
26	Y024	Y124	Y224	Y324
9	Y025	Y125	Y225	Y325
28	Y026	Y126	Y226	Y326
11	Y027	Y127	Y227	Y327
20	Y037	Y130	Y230	Y330
34	Y031	Y131	Y231	Y331
32	Y032	Y132	Y232	Y332
1	Y033	Y133	Y233	Y333

Connection Point	1st INT board	2nd INT board	3rd INT board	4th INT board
2	Y034	Y134	Y234	Y334
35	Y035	Y135	Y235	Y335
3	Y036	Y136	Y236	Y336
33	Y037	Y137	Y237	Y337

2.2 Variables of Connection between PLC and NC

The PLC and the NC communicate through RAM with the help of flags (1-bit variables) and registers (16-bit variables). In the view of PLC there are input and output flags and registers. Input flags and registers are set by the NC, while those of the output by the PLC.

2.2.1 Flags from NC to PLC (Input Flags)

Reference to input flags can be done with character I and three digits similarly to interface input flags stored in RAM:

Ipqr

The first digit must be equal to or greater than 4. The value range of the first digit:

p=4,5,6,7,8,9

The value range of the second digit (q):

q=0,1,2,3,4,5,6,7,8,9

The third one (r) defines the serial number of a bit within the selected byte and is therefore octal. Its value range is:

r=0,1,2,3,4,5,6,7

In case of word operand reference to an input flag group can be made in the PLC program by dropping the last digit:

Ipq

In module :001, i.e. on level No. 1 also the change test of input flags is enabled. The change test can be executed with the help of statement

Vpqr

in case of a bit operand, while with the help of statement

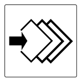
Vpq

in case of a word operand. Interpretation of indexes p, q, r corresponds to that of Ipqr.
The result of statement Vpqr is 1 if the value of input flag Ipqr of the previous PLC time slice differs from that valid in the current time slice.

In the followings a full list of input flags is shown:

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I400	Reference point return mode push-button
I401	Manual handle mode push-button
I402	Incremental jog mode push-button
I403	Jog mode push-button
I404	
I405	Manual data input mode push-button
I406	Automatic mode push-button
I407	Edit mode push-button

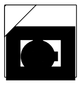
If Y520=1 (operation mode selected by softkey from NC keyboard, action menu MODES), or Y532=1 (selected from machine control board 2) the current state of mode push-buttons is sent by the NC through flags I400, ..., I407.
If Y520=1 (mode buttons operate from SW control panel) mode switch is executed by means of selecting one of screens OPEATOR'S PANEL, POSITION or CHECK.

Afterwards action menu MODES F[1] must be selected after pressing action menu button  .

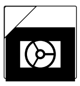
In this case the captions of the different modes appear on softkeys. The desired mode can be selected as the effect of the appropriate softkey.
If Y532=1 mode buttons operate from machine control board 2 and all modes can be displayed directly by means of push-buttons.

☞ ***Warning!***
Always only one of Y520 or Y532 can be 1, i.e. modes can be selected exclusively from either softkeys or machine control board 2!

I400: Reference point return mode push-button

The flag is set to 1, if operator activates softkey REFERENCE or mode push-button  .

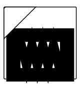
I401: Manual handle mode push-button

The flag is set to 1, if operator activates softkey HNDL or mode push-button  .

I402: Incremental jog mode push-button

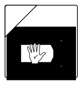
The flag is set to 1, if operator activates softkey INCR or mode push-button  .

I403: Jog mode push-button

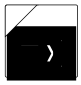
The flag is set to 1, if operator activates softkey JOG or mode push-button  .

I404: -

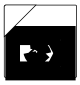
I405: Manual data input mode push-button

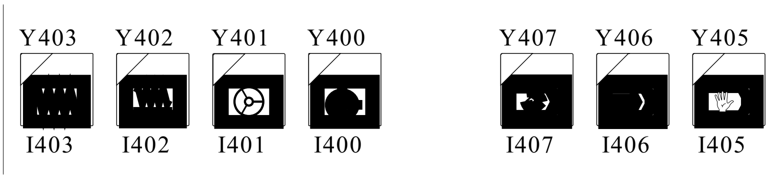
The flag is set to 1, if operator activates softkey MDI or mode push-button  .

I406: Automatic mode push-button

The flag is set to 1, if operator activates softkey AUTO or mode push-button  .

I407: Edit mode push-button

The flag is set to 1, if operator activates softkey EDIT or mode push-button key  .



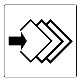
Arrangement of mode buttons on machine control board 2

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I410	1st axis selector push-button
I411	2nd axis selector push-button
I412	3rd axis selector push-button
I413	4th axis selector push-button
I414	5th axis selector push-button
I415	6th axis selector push-button
I416	7th axis selector push-button
I417	8th axis selector push-button

If Y521=1 (axis selected by softkey from NC keyboard, action menu AXES) the current state of axis push-buttons is sent by the NC through flags I410, ..., I417.
The axes are indexed according to the axis arrangement seen in display: X, Y, Z, U, V, W, A, B, C. If a letter is not selected for an axis, the next one takes its place.

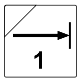
I410, ..., I417: 1st, ..., 8th axis selector push-button
The flag is set to 1, if the operator activates the 1st, ..., 8th axis softkey push-button.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I420	1 increment push-button
I421	10 increment push-button
I422	100 increment push-button
I423	1000 increment push-button
I424	
I425	
I426	Automatic tool length measurement softkey
I427	JOG rapid traverse push-button

If Y522=1 (increment selected by softkey from NC keyboard, action menu INCR), or Y532=1 (selected from machine control board 2) the current state of increment push-button is sent by the NC through flags I420, ..., I423.
If Y522=1 (increment size selection operates from SW control panel) increment size is chosen by means of opening one of screens OPEATOR'S PANEL, POSITION or CHECK.
Afterwards action menu INCR F3 must be selected after pressing action menu button  . In this case the captions of the different incremenet sizes (1, 10, 100, 1000) appear on softkeys. The desired increment can be selected as the effect of the appropriate softkey.
If Y532=1 increment size selection operates from machine control board 2 and all increment sizes can be activated directly by means of push-buttons.

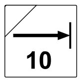
☞ ***Warning!***
Always only one of Y520 or Y532 can be 1, i.e. increment sizes can be selected exclusively from either softkeys or machine control board 2!

I420: 1 increment push-button

The flag is set to 1, if the operator activates the <1> increment softkey or the 

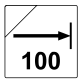
push-button.

I421: 10 increment push-button

The flag is set to 1, if the operator activates the <10> increment softkey or the 

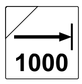
push-button.

I422: 100 increment push-button

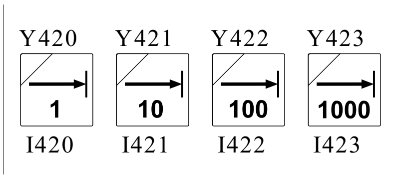
The flag is set to 1, if the operator activates the <100> increment softkey or the 

push-button.

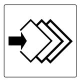
I423: 1000 increment push-button

The flag is set to 1, if the operator activates the <1000> increment softkey or the 

push-button.

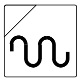


Arrangement of increment buttons on machine control board 2

I426: Automatic tool length measurement softkey
In case of lathe controls select action menu T. LENG MEASUR ^{F4} (length offset measurement) within screen OFFSETS ^{F5}. Press action menu button  . Softkey AUTO MEAS F^{3} appears among the actions. In case this softkey is pressed value of I426 is set to 1.

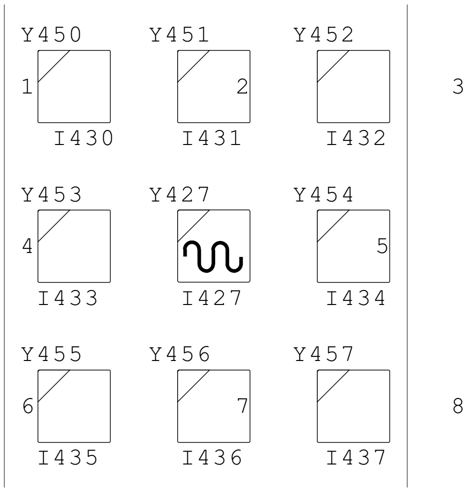
If Y530=1 (JOG selected by softkey from NC keyboard), or Y531=1 (selected from machine control board 1), or Y532=1 (machine control board 2) the current state of JOG rapid traverse push-button is sent by the NC through flag I427.

I427: JOG rapid traverse push-button

The flag is set to 1 if operator activates the rapid traverse  push-button.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I430	JOG 1 push-button
I431	JOG 2 push-button
I432	JOG 3 push-button
I433	JOG 4 push-button
I434	JOG 5 push-button
I435	JOG 6 push-button
I436	JOG 7 push-button
I437	JOG 8 push-button

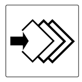
I430, ..., I437: JOG 1, ..., 8 push-buttons
It can only be used if Y531=1 (selected from machine control board 1), or Y532=1 (selected from machine control board 2) is in effect. In this case if flag is set to 1 the appropriate axis direction push-button has been activated on either machine control board.
The diagram shows the arrangement and numeration of JOG buttons on machine control board 1 and machine control board 2. If for example button (1) is pressed, then flag I430 is set to 1. If caption X+ is indicated on top of the button (1), the axis direction flag X+ needs to be switched on. (The caption-specific arrangement of JOG buttons may alter.)
In case of machine control board 2 each push-button is equipped with a lamp switched through flags Y427, Y450, ...Y457.



Arrangement of JOG buttons on machine control board 2

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I440	Test push-button
I441	Machine lock push-button
I442	Dry run push-button
I443	Block restart push-button
I444	Block return push-button
I445	Conditional stop push-button
I446	Conditional block skip push-button
I447	Single block mode push-button

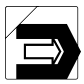
If Y523=1 (state selection from NC) or Y532=1 (from machine control board 2) the signals of state buttons are sent by the NC through flags I440, ..., I447.
If Y523=1 (state selection operates from SW control panel) state is chosen by means of opening one of screens OPEATOR'S PANEL, POSITION or CHECK.

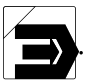
Afterwards action menu STATES F[5] must be selected after pressing action menu button  .

In this case the captions of the available states appear on softkeys. The desired state can be selected as the effect of the appropriate softkey.
If Y532=1 state selection operates from machine control board 2 and all states can be displayed directly by means of push-buttons.

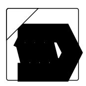
☞ ***Warning!***
Always only one of Y520 or Y532 can be 1, i.e. states can be selected exclusively from either softkeys or machine control board 2!

I440: Test push-button

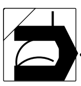
The state of the flag goes high if operator presses softkey TEST or push button  .

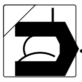
I441: Machine lock push-button
The state of the flag goes high if operator presses softkey MACHINE LOCK or push button .

I442: Dry run push-button

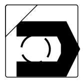
The state of the flag goes high if operator presses softkey DRY RUN or push button  .

I443: Block restart push-button

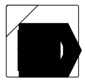
The state of the flag goes high if operator presses softkey BLOCK RESTART or push button .

I444: Block return push-button
The state of the flag goes high if operator presses softkey BLOCK RETURN or push button .

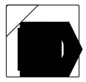
I445: Conditional STOP push-button

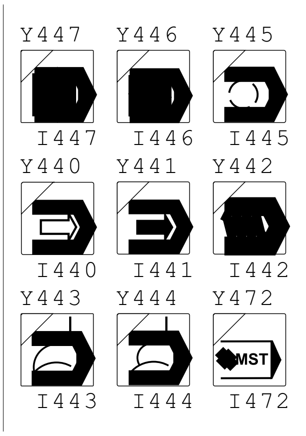
The state of the flag goes high if operator presses softkey COND STOP or push button .

I446: Conditional block push-button

The state of the flag goes high if operator presses softkey COND. BLOCK or push button .

I447: Single block mode push-button

The state of the flag goes high if operator presses softkey SINGLE BLOCK or push button .



Arrangement of state buttons
on machine control board 2

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I450	1st user's push-button
I451	2nd user's push-button
I452	3rd user's push-button
I453	4th user's push-button
I454	5th user's push-button
I455	6th user's push-button
I456	7th user's push-button
I457	8th user's push-button

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I460	9th user's push-button
I461	10th user's push-button
I462	11thuser's push-button
I463	12th user's push-button
I464	13th user's push-button
I465	14th user's push-button
I466	15th user's push-button
I467	16th user's push-button

I450, ..., I467: 1st, ..., 16th user's push-button
The user can - as written in the Insallation Manual of the NC control - connect buttons or rotary switches to definite places of the operator's panel matrix. This way the application of at most 16 flags is possible. If flag Y537=1 the state of user's buttons or rotary switches is sent by the NC to the PLC through input flags I450, ..., I457, I460, ..., I467. It can be used for example for testing state of axis and increment selector switches placed on top of the external handwheel boxes.

Assignment of input flags in case of applying NCT external handwheel

I450 - X axis selected
I451 - Y axis selected
I452 - Z axis selected
I453 - 4th axis selected
I454 - 5th axis selected
I455 - 6th axis selected
I456 -
I457 -

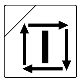
I460 - 1 increment
I461 - 10 increment
I462 - 100 increment

I463	-	
I464	-	=1: enable mode switch/axis selection from machine keyboard, =0: external handwheel mode
I465	-	external handwheel plugged
I466	-	
I467	-	

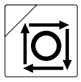
Flag Identity	Meaning of Flag if Value=1 (TRUE)
I470	Start push-button
I471	Stop push-button
I472	Function lock push-button
I473	
I474	M3 push-button
I475	M4 push-button
I476	M5 push-button
I477	RESET push-button

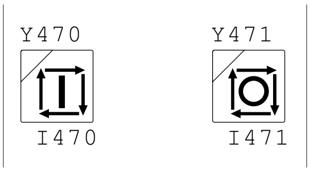
If Y531=1, or Y532=1 (selection of machine control board 1 or 2) the state of push-buttons M3, M4, M5 and RESET are sent by the NC through flags I474, ..., I477. If Y532=1 (selection of machine control board 2) also the state of START, STOP and function lock push-buttons are sent by the NC.

I470: Start push-button

The flag is set to 1 if operator activates Start  push-button. It is used only when applying machine control board 2.

I471: Stop push-button

The flag is set to 1 if operator activates Stop  push-button. It is used only when applying machine control board 2.

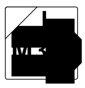


Arrangement of start and stop buttons on machine control board 2

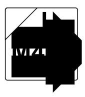
I472: Function lock push-button

The flag is set to 1 if operator activates function lock  push-button. It is used only when applying machine control board 2.

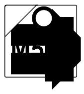
I474: M3 push-button

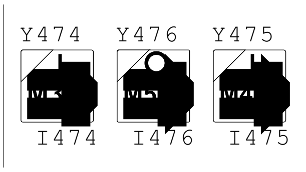
The flag is set to 1 if operator activates push-button M3  .

I475: M4 push-button

The flag is set to 1 if operator activates push-button M4  .

I476: M5 push-button

The flag is set to 1 if operator activates push-button M5  .



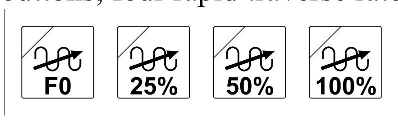
Arrangement of spindle rotation buttons on machine control board 2

I477: RESET push-button
The flag is set to 1 if operator activates RESET push-button.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I480	1st user's push-button of machine control board 2
I481	2nd user's push-button of machine control board 2
I482	3rd user's push-button of machine control board 2
I483	4th user's push-button of machine control board 2
I484	5th user's push-button of machine control board 2
I485	6th user's push-button of machine control board 2
I486	7th user's push-button of machine control board 2
I487	8th user's push-button of machine control board 2

8 lighted push-buttons are mounted on machine control board 2 the function of which is defined by the machine builder. Hereby the machine builder must also take care of push-button labels or captions. The following functions in the order of importance are expedient to be defined for these buttons:

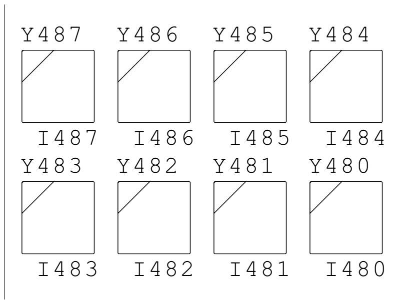
– If more than four axes are built in the machine the axis selector buttons of the 4th, 5th, etc. axes are to be put here. In this case condition Y521=0 must be true, i.e. the axes are not selected from SW control panel (softkeys).
– Coolant-operating buttons.
– Rapid traverse override buttons; four rapid traverse rates can be selected here:



Advised arrangement of rapid traverse override buttons

– Tool clamp/unclamp etc.

I480, ..., I487: 1st, ..., 8th user's push-button of machine control board 2
If one of the 8 user's push-buttons is activated on the machine control board 2, the appropriate flag is set to 1.

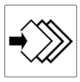


Arrangement of user buttons on machine control board No.2

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I490	
I491	
I492	
I493	
I494	
I495	
I496	
I497	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I500	PLC defined softkey 1
I501	PLC defined softkey 2
I502	PLC defined softkey 3
I503	PLC defined softkey 4
I504	PLC defined softkey 5
I505	PLC defined softkey 6
I506	PLC defined softkey 7
I507	PLC defined softkey 8

If Y524=1 (selected by PLC softkeys from NC keyboard) signs of the 8 optionally used softkeys offered by the NC is sent through flags I500, ..., I507. (If Y524=0 these softkeys are not offered by the NC.) The caption of the softkeys can be defined by the PLC programmer in module :197. The softkeys can be reached by means of selecting one of screens OPERATOR'S PANEL, POSITION or CHECK.

Afterwards action menu MACHINE F^6 must be selected after pressing action menu button  .

In this case the captions defined by the PLC programmer in module :197 appear on softkeys.

I500, ..., I507: PLC defined softkey 1, ..., 8
The flag is set to 1 if operator presses softkey 1, ..., 8.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I510	First call of module :001
I511	Automatic operation interrupted
I512	
I513	
I514	
I515	
I516	
I517	Parts required = Parts count

I510: First call of module :001
The flag is 1 during the full period of the first running of module :001 after power-on. It is used in PLC program for gating of initialization procedure after power-on.

I511: Automatic operation interrupted
This flag is set to 1 if automatic operation is interrupted due to emergency state, change of operation mode or RESET. In this case caption INTD is displayed in the 3rd field of status bar. The PLC programmers should take care of storing functions not executed into the suspended block, and after canceling INTD state, of executing them, provided automatic operation is restarted unconditionally or with condition BLOCK RESTART. To enable the modification of functions by means of manual data input in suspended state is also a task of the programmer, e.g. to overwrite spindle revolution so that by returning to automatic operation the new S is valid.

I517: Parts required = Parts count
If in the TIME/COUNTER table the value of PARTS COUNT has reached the value of PARTS REQUIRED the flag is set to 1.
The value of parts count is increased by one
- by means of commands M02 and M30, if parameter 9024 **PRTCNTM** =0,
- by means of command Mnn, if parameter 9024 **PRTCNTM**=nn.

(The value of PARTS COUNT equals to the value of parameter 9022 **PRTCOUNT**, so does the value of PARTS REQUIRED to the value of parameter 9023 **PRTREQRD**.)

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I520	1st M function strobe, code in register RH000
I521	2nd M function strobe, code in register RH001
I522	3rd M function strobe, code in register RH002
I523	4th M function strobe, code in register RH003
I524	5th M function strobe, code in register RH004
I525	S function strobe, code in register RH005
I526	T function strobe, code in register RH006
I527	“A” function strobe, code in register RH007

I520, ..., I524: 1st, ..., 5th M function strobe
At most 5 functions M, which are sent to PLC can be written within a program block. According to the order written in the block NC writes the first loaded M data into register RH000 and sets flag I520 to 1, it writes the 2nd M data into register RH001 and sets flag I521 to 1, and so on. The PLC programmer determines the order of the execution of the different functions M within the given block.

I525: S function strobe
If function S is written within a program block data S is stored into input register RH005 and the NC sets flag I525 to 1, namely it validates the value of register RH005.

I526: T function strobe
If function T is written within a program block data T is stored into input register RH006 and the NC sets flag I526 to 1, namely it validates the value of register RH006..

I527: “A” function strobe
If address A is enabled for function (parameter 0183 **A.MISCEL**=1), and function A is written within a program block data A is stored into input register RH007 and the NC sets flag I527 to 1 namely it validates the value of register RH007.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I530	“B” function strobe, code in register RH008
I531	“C” function strobe, code in register RH009
I532	Chopping Function Strobe, Code on Flag I675
I533	
I534	
I535	
I536	Valid push-button code in register RH049
I537	Message on screen

I530: “B” function strobe
If address B is enabled for function (parameter 0186 **B.MISCEL**=1), and function B is written within a program block data B is stored into input register RH008 and the NC sets flag I530 to 1 namely it validates the value of register RH008..

I531: “C” function strobe
If address C is enabled for function (parameter 0189 **C.MISCEL**=1), and function C is written within a program block data C is stored into input register RH009 and the NC sets flag I531 to 1 namely it validates the value of register RH009.

I532: Chopping Function Strobe, Code on Flag I675
If chopping on command G81.1 or chopping off command G80 is executed NC strobes flag I532 and indicates command on or off by setting or resetting of flag I675.

I536: Valid push-button code in register RH049
If a button is pushed on data input keyboard flag I536 is set to 1 and the button code appears in register RH049. Push-button codes are specified in chapter 6.5 Listing of Push-button Codes on page 237.

I537: Message on screen
If a message is displayed in the message field, i.e. in the 2nd line of screen, no matter which one, NC or PLC had sent it, this flag is set to 1. The message code can be found in register RH020. The code table contains the codes and their description in chapter 6.4 Listing of Global Messages 234 on page.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I540	Status of Machine on output
I541	Status of NC Ready signal
I542	Machine on output disabled
I543	Module :000 started from beginning
I544	
I545	Programmed reference point return (G28)
I546	Executable block in buffer
I547	Stop request from NC

I540: Status of Machine on output
MACHINE ON output is a 24V output found on interface board. In case MACHINE ON output is on

- other outputs of interface board receive power supply,
- the measuring system closes position control loop (otherwise it only measures),
- the NC enables any movement start,
- or PLC action.

In case MACHINE ON output is off the NC registers EMG (emergency stop) status and disables all above actions.

Flag I540 serves for testing state of MACHINE ON output. MACHINE ON output is the logic multiplication of the following signals:

MACHINE ON=(machine on request) and (NC ready) and (no crash), i.e.
I540=(Y540) and (I541) and (I542),

that is MACHINE ON signal is on only if the PLC requests power-on, the NC is ready and there is no crash, e.g. servo error.

I541: Status of NC Ready signal
The status of NC Ready signal can be tested separately through flag I541.

I542: Machine on output disabled
If the NC observes fatal error (servo, feedback, encoder) and the machine magnetic must be turned off this flag is set to 1.

I543: module :000 start from beginning
This flag is set to 1 in the PLC cycle, in which module :000 is started from the beginning. If in the same cycle module :000 does not reach statement J0 it is set to 0 in the next cycle. If module :000 is always terminated in the starting cycle the flag always remains 1.

I545: Programmed reference point return (G28)
If the control executes programmed reference point return (G28) this flag is set to 1.

I546: Executable block in buffer
If a block is ready to be executed by pressing START this flag is set to 1.

I547: STOP request from NC
If the NC arrives at STOP state during execution, e.g. due to an error, or in single block mode this flag is set to 1. In this case it is the PLC programmer's task to turn on the STOP lamp.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I550	Interpolator stopped
I551	Interpolator empty (terminated)
I552	Override disabled
I553	Spindle rotation request
I554	Thread cutting (G33)
I555	Thread cutting cycle (G76, G78)
I556	
I557	

☞ *The flags below are effective only in case of axes selected for start from NC at flags Y630, ..., Y637.*

I550: Interpolator stopped
If the flag
=0 the interpolator is in START state
=1 the interpolator is in STOP state.
The interpolator STOP state does not correspond to STOP state of the control (STOP lamp is on). This flag is set to 1 due to RESET (neither START, nor STOP lamp is on), or during plain function block (START lamp is on), or perhaps in FEED HOLD state (Y542=1). If the flag is set to 1 (STOP state) it does not mean, that the given axis has been already stopped, in order to do this the appropriate flag I560, ..., I567 (1^{st}, ..., 8^{th} axis in position) must also be set to 1.

I551: Interpolator empty (terminated)
If the flag
=0 interpolator is active: it is in motion, or stopped but there is still path left
=1 interpolator has been terminated: empty.
This flag is set to 1 due to RESET. If I550=0 and I551=0 the control is in START state, but not only in this case. If I550=1 and I551=0 the control is in STOP state, but not only in this case.

I552: Override disabled
This flag is set to 1 if override and feed STOP is disabled on the control due to technological reasons when executing commands
– G33, G34, G63, G76, G78, G84, G84.1 in case of turning control,
– G33, G63, G74, G84 in case of milling control.

I553: Spindle rotation request
The interpolator sets this flag to 0 before starting one of commands G0, G4, G28, G29, G30, G31, G53 and single axis movements (JOG, manual handle, reference point return) In this case the interpolator starts the movement unconditionally, independent of the state of output flag Y650 (spindle rotates).
The interpolator sets this flag to 1 before executing commands G1, G2, G3, G33, G34 if spindle does not take part in the interpolation (I651=0 or I661=0 spindle loop not closed).
In this case the interpolator does not start the movement till the PLC permits it by setting output flag Y650 (spindle rotates) to 1.

In case of miscellaneous blocks (containing both interpolation and function) this flag can be used for synchronizing interpolator and PLC activities. For during block execution the interpolator and the PLC to receive their part of the given block at the same time the PLC must be aware of the following cases:

```
G0 Xx Yy M3
G0 Xx Yy M4
G0 Xx Yy M5
G0 Xx Yy M19
```

Spindle rotation request (I553=0) is not transferred by the interpolator, the spindle can be started or stopped parallel to the movement.

```
G1 Xx Yy Ff M3
G1 Xx Yy Ff M4
```

The interpolator sets flag I553 and waits with movement start till the PLC executes command M3 or M4 (switches on spindle) and permits movement with flag Y650 (spindle rotates).

```
G1 Xx Yy Ff M5
G1 Xx Yy Ff M19
```

During block execution flag I553 is set. The PLC must wait until the interpolator becomes empty (I551=1) and the spindle can be stopped (M5) only than.

I554: Thread cutting (G33)
If this flag is set to 1 the interpolator executes a thread cutting interpolation G33 or G34. In this case switching STOP state (Y471) on is disabled, only the spindle may be stopped.

I555: Thread cutting cycle (G76, G78)
If the turning machine control is doing thread cutting in one of the cycles G76 or G78 this flag is set to 1. (Flags override disabled I552=1 and thread cutting I554=1 are also set.) In this case both pressing the STOP button and setting flag Y471 (STOP state) are to be enabled too in order to be effective the thread cutting cycle stop function, detailed in programming manual. This function generates interrupted (INTD) state, therefore it must be handled.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I560	1st axis in position
I561	2nd axis in position
I562	3rd axis in position
I563	4th axis in position
I564	5th axis in position
I565	6th axis in position
I566	7th axis in position
I567	8th axis in position

I560, ..., I567: 1st, ..., 8th axis in position
If the appropriate axis is within the tolerance interval set at parameters 4261 **INPOS1**, ..., 4268 **INPOS8** compared to the difference between the current position and the desired position the state of the appropriate input flag I560, ..., I567 is 1 (TRUE).

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I570	1st axis lubrication request
I571	2nd axis lubrication request
I572	3rd axis lubrication request
I573	4th axis lubrication request
I574	5th axis lubrication request
I575	6th axis lubrication request
I576	7th axis lubrication request
I577	8th axis lubrication request

I570,...,I577: 1st,...,8th axis lubrication request
Flags for lubrication according to the path already done. If the axis has already finished path set at parameter 0161 **LUBCONST1**, ..., 0168 **LUBCONST8** on the appropriate axis the NC sets the appropriate flag I57n to 1. The flag is on for 20 msec period.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I580	
I581	
I582	
I583	
I584	
I585	
I586	
I587	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I590	
I591	
I592	
I593	
I594	
I595	
I596	
I597	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I600	
I601	
I602	Program execution in DNC
I603	Program execution in NCT DNC
I604	Message acknowledged
I605	Transmission error
I606	Data transmitted from memory
I607	Data received in memory

I602: Program execution in DNC
The flag is 1 in case DNC program execution is selected on control. This may occur if DNC menu of Run action menu of DIRECTORY screen is selected, or if flag Y602 is set to 1.

I603: Program execution in NCT DNC
The flag is 1 in case NCT DNC program execution is selected on control. This may occur if NCT DNC menu of Run action menu of DIRECTORY screen is selected from data input keyboard, or in case flag Y603 is set to 1.

I604: Message acknowledged
PLC strobes flag Y604 with command U604 and waits until flag I604 turns to 1. Afterwards flag Y604 must be switched off by means of command D604. This pair of flags is for synchronizing manual handle machining executed on PC. (Both manual data input mode and manual handle mode are on: Y405AY401).

I605: Transmission error
If the PLC program initiates data transfer by setting either flag Y605 or Y606 to 1 and transmission error occurs this flag is set to 1 by the NC. After it the PLC program should reset transmission command flags by the instructions D605 or D606. The NC gives the error message the following cases:
- Overrun error during reception (data are coming more quickly than the PLC evaluates them).
- If the I/O channel is busy. E.g.: The PLC program initiates data transfer during part program input/output trough serial port.
- Hardware error (eg.: parity, overrun) happens during data input.

I606: Data transmitted from memory
If the PLC desires to send data from memory (F010, ..., F499) through a periphery it sets flag Y606 to 1. After the data output had occured the NC sets I606 to 1. Then the PLC should set flag Y606 to 0, hereat the data transfer is finished. Befor the PLC program would send new data it must wait until flag I606 is set to 0.
The start address of valid data is contained by register RH051, while the number of bytes to be sent (record length) by register RH052. The number of periphery, through which the data is sent is defined at register RH053.

I607: Data received in memory
The PLC program opens input channel by setting flag Y605 to 1. If all bytes specified in register RH055 has arrived to the memory location specified by registers RH054 the NC sets flag I607 to 1. If he PLC has evaluated the data sent by the NC it sets flag Y607 to 1. As a handshake NC will then reset I607. This means that the selected memory area can be overwritten.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I610	1st axis motion request
I611	2nd axis motion request
I612	3rd axis motion request
I613	4th axis motion request
I614	5th axis motion request
I615	6th axis motion request
I616	7th axis motion request
I617	8th axis motion request

I610,...,I617: 1st, ..., 8th axis motion request
Before the interpolator sends motion command to an axis in the given path calculation cycle, it asks for motion request on the appropriate axis. It waits until the PLC permits the motion command in level 0 with the appropriate flag Y610, ..., Y617 set to 0.
These flags can be used for example for mechanical fixing of axes, or if a motor drives more axes to set the movable axes. If these are unnecessary, when initializing flags Y610, ..., Y617 are set to 0 (motion request) and so the interpolator works continuously. After the motion request flag has been ceased, before fixing an axis or switching over the axis switch the given axis must reach its desired position. (See flags I560, ..., I567).

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I620	1st axis rapid traverse request
I621	2nd axis rapid traverse request
I622	3rd axis rapid traverse request
I623	4th axis rapid traverse request
I624	5th axis rapid traverse request
I625	6th axis rapid traverse request
I626	7th axis rapid traverse request
I627	8th axis rapid traverse request

I620,...,I627: 1st, ..., 8th axis rapid traverse request
Before the interpolator sends rapid traverse motion command (G0, G28, G29, G30, G53, activating JOG rapid traverse push-button) to an axis, in the given path calculation cycle it sends a rapid traverse request on the appropriate axis. Flags I620, ..., I627 are always transferred together with the motion request flags. It waits until the PLC permits the motion command with the appropriate flag Y610, ..., Y617 set to 0.
These flags can be used for example if different mechanical transmissions need to be connected to feed motions and to rapid traverse movements on an axis.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I630	
I631	
I632	
I633	
I634	
I635	
I636	
I637	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I640	G51.2: polygonal turning
I641	polygonal turning, reverse direction (Q<0)
I642	
I643	
I644	
I645	
I646	
I647	

☞*Available only in turning machine controls*

I640: G51.2: polygonal turning
The flag turns to high if program block G51.2 P_ Q _ is to be executed. The ratio of P/Q defines the ratio of revolution of the main spindle (workpiece) and the slave spindle (tool). Programmed absolute value of P is available in register RH040 while value Q in register in RH041. The revolution of the tool spindle is calculated according the formula below:

$$S_{toolspindle} = \frac{\mathrm{Q}}{\mathrm{P}} S = \frac{RH041}{RH040} S$$

The PLC program should turn the tool spindle to the revolution calculated before, then it should request synchronization via flags Y655 or Y665.
Command G50.2 turns polygonal turning off and flag I640goes to low. The PLC program should cancel the synchronization of the two spindles, then turn the tool spindle off.

I641: Polygonal turning, reverse direction (Q<0)
The direction of revolution of the tool spindle is determined by the sign of address Q in blocks G51.2 P_ Q _. If the value of address Q is negative flag I641 turns to 1. The PLC program should turn the tool spindle in the same direction as the main spindle if flag I641=0 and the reverse one if it is 1, then should request synchronization the same or the counter direction by flags Y656 or Y666.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I650	1^{st} spindle command signal ramping ready
I651	1^{st} spindle orientation ready
I652	1^{st} spindle in position
I653	State G96 on active spindle
I654	State G25 on active spindle
I655	Revolution fluctuated on active spindle
I656	1^{st} spindle $n=n_S$
I657	1^{st} spindle n=0

I650: 1^{st} spindle command signal ramping ready
The control sends the 1^{st} spindle command signal to the drive by linearly ramping as set at parameters (5041 **S1 ACCT**, 5061 **S1 DECT**). If after a while the command signal does not change the NC sets this flag to 1. Waiting for the switch-on of flag I656 can be started if this signal has arrived. For the control executes rise and fall of the command signal in every 20 msec, the flag is set to 0 in the PLC cycle following the command signal transfer command.

I651: 1^{st} spindle orientation ready
If the spindle drive can be positioned, spindle orientation can be requested from the NC by switching on flag Y651 (U651). If the orientation is finished (spindle is set on the zero pulse of the encoder) the NC acknowledges it by switching on input flag I651.

I652: 1^{st} spindle in position
If the spindle functions as axis, i.e. the position loop is closed (I651=1), flag I652 shows, if the lag of spindle is within the tolerance interval set at parameter 4269 **INPOSS1**. The orientation is finished if condition (I651AI652) is true. It is advised to test the flag, if parameter 7169 **REFSHIFTS1** is other than 0, i.e. the spindle is not stopped on the zero pulse, but is comparatively offset. Flag I651 is set to 1 if the interpolator has stepped the offset, while flag I652 is 1 in case the lag of the measuring system has ceased.

I653: State G96 on active spindle
If request of the constant surface speed calculation is switched on on the active spindle by means of command G96 this flag is set to 1. In state G97 (constant surface speed calculation is off) the flag is set to 0. In state G96 the contents of register RH012 (calculated spindle revolution for the current position) must be copied to spindle revolution register RH060 by the PLC programmer, for the revolution of the appropriate constant surface speed to be in effect also in case of command signal transfer.

I654: State G25 on active spindle
If the spindle revolution fluctuation check has been switched off by means of command G25 this flag is set to 1. In this case flag I655 is always 0 (no fluctuation), independent of the spindle revolution fluctuation. When turning the power on this flag is always 0. The fluctuation is monitored by testing the 1^{st} spindle's encoder if flag Y660=0, while in state Y660=1 by testing the 2^{nd} spindle's encoder.

I655: Revolution fluctuated on active spindle
If flag I654 is 0, provided the spindle is mounted with encoder, the revolution fluctuation of the spindle is measured by the NC in respect of the values set at parameters 5001 **TIME**, 5002 **SCERR**, 5003 **FLUCT%** and 5004 **FLUCTW** if the 1st spindle is active (Y660=0). If the 2nd one is active (Y660=1) then 5441 **TIME2**, 5442 **SCERR2**, 5443 **FLUCT%2** and 5444 **FLUCTW2** parameters are used. If the revolution fluctuates flag I655 is set to 1.

I656: 1st spindle $n=n_S$
Provided the spindle is mounted with encoder the NC sets flag I656 to 1 if the spindle has already registered the revolution. Flag I656 is switched according to the values set at parameters 5005 **N%** and 5006 **NW**.

I657: 1st spindle n=0
Provided the spindle is mounted with encoder the NC sets flag I657 to 1 if the spindle revolution is less than the value set at parameter 5007 **N0**.

☞*Warning!*
Flags I656 $n=n_S$ and I657 n=0 function independent of the state of flag Y654, i.e. that the command signal output occurs from register RH060 or RH061.
In case of spindle stop:
I656=1 and I657=1

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I660	2nd spindle command signal ramping ready
I661	2nd spindle orientation ready
I662	2nd spindle in position
I663	1st spindle synchronized to the 2nd one
I664	2nd spindle synchronized to the 1st one
I665	
I666	2nd spindle $n=n_S$
I667	2nd spindle n=0

I660: 2nd spindle command signal ramping ready
The control sends the 2nd spindle command signal to the drive by linearly ramping as set at parameters (**5081 S2 ACCT**, **5101 S2 DECT**). If after a while the command signal does not change the NC sets this flag to 1. Waiting for the switch-on of flag I666 can be started if this signal has arrived. For the control executes rise and fall of the command signal in every 20 msec, the flag is set to 0 in the PLC cycle following the command signal transfer command.

I661: 2nd spindle orientation ready
If the spindle drive can be positioned, spindle orientation can be requested from the NC by switching on flag Y661 (U661). If the orientation is finished (spindle is set on the zero pulse of the encoder) the NC acknowledges it by switching on input flag I661.

I662: 2nd spindle in position
If the spindle functions as axis, i.e. the position loop is closed (I661=1), flag I662 shows, if the lag of spindle is within the tolerance interval set at parameter 4270 **INPOSS2**. The orientation is finished if condition (I661AI662) is true. It is advised to test the flag, if parameter 7170 **REFSHIFTS2** is other than 0, i.e. the spindle is not stopped on the zero pulse, but is comparatively offset. Flag I661 is set to 1 if the interpolator has stepped the offset, while flag I662 is 1 in case the lag of the measuring system has ceased.

I663: 1st spindle synchronized to the 2nd one
The PLC indicates to the NC by turning flag Y655 to 1 to synchronize the 1st spindle to the 2nd one. If the distance of the zero pulses of the two spindles is in the range defined on parameters 5402 SPSHIFT1± 4269 INPOSS1, the NC turns the flag I663 to 1. It indicates to the PLC that synchronization is over.

I664: 2nd spindle synchronized to the 1st one
The PLC indicates to the NC by turning flag Y665 to 1 to synchronize the 2nd spindle to the 1st one. If the distance of the zero pulses of the two spindles is in the range defined on parameters 5422 SPSHIFT2± 4270 INPOSS2, the NC turns the flag I664 to 1. It indicates to the PLC that synchronization is over.

I666: 2nd spindle $n=n_S$
Provided the spindle is mounted with encoder the NC sets flag I666 to 1 if the spindle has already registered the revolution. Flag I666 is switched according to the values set at parameters 5445 **N%2** and 5446 **NW2**.

I667: 2nd spindle n=0

Provided the spindle is mounted with encoder the NC sets flag I667 to 1 if the spindle revolution is less than the value set at parameter 5447 **N02**.

☞*Warning!*

Flags I666 $n=n_S$ and I667 n=0 function independent of the state of flag Y664, i.e. that the command signal output occurs from register RH065 or RH066.

In case of spindle stop:

I666=1 and I667=1

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I670	1st analog command signal ramping ready
I671	
I672	2nd analog command signal ramping ready
I673	
I674	
I675	Chopping Function Code (G81.1, G80)
I676	Axis Is Chopping
I677	Chopping Axis on Point R

I670, I672: 1st, 2nd analog command signal ramping ready
The control sends the 1st and 2nd analog output command signal to the drive by linearly ramping as set at parameters 0124 **A1 ACC**, 0144 **A2 ACC**, 0125 **A1 DCC**, 0145 **A2 DCC**. If after a while the command signal does not change the NC sets this flag to 1. For the control executes command signal ramping in every 20 msec, the flag is set to 0 in the PLC cycle following the command signal transfer command.

I675: Chopping Function Code (G81.1, G80)
When executing G81.1 command NC sets flag I675 and strobes flag I532. Chopping begins if PLC sets flag Y675. Upon execution of command G80 NC resets flag I675 and strobes flag I532. Chopping cancelled when PLC resets flag Y675.

I676: Axis Is Chopping
If PLC sets flag chopping Y675 PLC should wait until NC sets flag I676. This flag indicates PLC that FIN signal can be set and execution of program can go on.

I677: Chopping Axis on Point R
NC sets the flag when chopping axis is on point R. If PLC resets flag Y675, NC moves chopping axis from lower dead point to point R, stops it and sets flag I677. This flag indicates PLC that the process is over and signal FIN can be set.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I680	
I681	
I682	
I683	
I684	
I685	
I686	
I687	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I690	
I691	
I692	
I693	
I694	
I695	
I696	
I697	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I700	1st indexed message on the screen
I701	2nd indexed message on the screen
I702	3rd indexed message on the screen
I703	4th indexed message on the screen
I704	5th indexed message on the screen
I705	6th indexed message on the screen
I706	7th indexed message on the screen
I707	8th indexed message on the screen

I700, ..., I707: 1st, ..., 8th indexed message on the screen
8 different user messages, indexed according to the contents of registers RH090, ..., RH097 can be displayed on the screen containing user messages with the help of flags Y700, ..., Y707. Of the maximum 8 messages only one, displayed in the 2nd line of screen, is active. (For reading the active message there is no need to switch over to the screen containing the user messages.)
Due to this only one of flag of I700, ..., I707 has TRUE state. It is the task of the PLC programmer to define the method of canceling the user messages.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I710	1st message on the screen
I711	2nd message on the screen
I712	3rd message on the screen
I713	4th message on the screen
I714	5th message on the screen
I715	6th message on the screen
I716	7th message on the screen
I717	8th message on the screen

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I790	65th message on the screen
I791	66th message on the screen
I792	67th message on the screen
I793	68th message on the screen
I794	69th message on the screen
I795	70th message on the screen
I796	71st message on the screen
I797	72nd message on the screen

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I800	73rd message on the screen
I801	74th message on the screen
I802	75th message on the screen
I803	76th message on the screen
I804	77th message on the screen
I805	78th message on the screen
I806	79th message on the screen
I807	80th message on the screen

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I890	145th message on the screen
I891	146th message on the screen
I892	147th message on the screen
I893	148th message on the screen
I894	149th message on the screen
I895	150th message on the screen
I896	151st message on the screen
I897	152nd message on the screen

I710, ..., I897: 1st, ..., 152nd message on the screen
152 different user messages can be displayed on the screen containing user messages with the help of flags Y710, ..., Y897. Of the maximum 152 messages only one, displayed in the 2nd line of screen, is active. (For reading the active message there is no need to switch over to the screen containing the user messages.)
Due to this only one of flags I710, ..., I897 has TRUE state. It is the task of the PLC programmer to define the method of canceling the user messages. To cancel an error message also the RESET push-button, the signal of which is sent through input flag I477 can be used.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I900	1st axis interpolator stopped
I901	1st axis interpolator empty (terminated)
I902	
I903	1st axis reference point ready
I904	
I905	
I906	
I907	1st axis drive ready

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I910	2nd axis interpolator stop
I911	2nd axis interpolator empty (terminated)
I912	
I913	2nd axis reference point ready
I914	
I915	
I916	
I917	2nd axis drive ready

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I920	3rd axis interpolator stopped
I921	3rd axis interpolator empty (terminated)
I922	
I923	3rd axis reference point ready
I924	
I925	
I926	
I927	3rd axis drive ready

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I930	4th axis interpolator stopped
I931	4th axis interpolator empty (terminated)
I932	
I933	4th axis reference point ready
I934	
I935	
I936	
I937	4th axis drive ready

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I940	5th axis interpolator stopped
I941	5th axis interpolator empty (terminated)
I942	
I943	5th axis reference point ready
I944	
I945	
I946	
I947	5th axis drive ready

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I950	6th axis interpolator stopped
I951	6th axis interpolator empty (terminated)
I952	
I953	6th axis reference point ready
I954	
I955	
I956	
I957	6th axis drive ready

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I960	7th axis interpolator stopped
I961	7th axis interpolator empty (terminated)
I962	
I963	7th axis reference point ready
I964	
I965	
I966	
I967	7th axis drive ready

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I970	8th axis interpolator stopped
I971	8th axis interpolator empty (terminated)
I972	
I973	8th axis reference point ready
I974	
I975	
I976	
I977	8th axis drive ready

☞ *The below flags are effective only in case of axes selected for start from PLC at flags Y630, ..., Y637.*

I900, I910, ..., I970: 1st, 2nd, ..., 8th axis interpolator stopped
If flag
=0 the interpolator is in START state on the appropriate axis
=1 the interpolator is in STOP state on the appropriate axis.
Due to RESET the flag is set to 1.
If the flag is 1 (STOP state) it does not mean, that the given axis has already stopped, this can only be achieved if the appropriate flag I560, ..., I567(1st, ..., 8th axis in position) is also set to 1.

I901, I911, ..., I971: 1st, 2nd, ..., 8th axis interpolator empty (terminated)
If the flag
=0 the interpolator is active in the appropriate axis: it moves or has already stopped, but there is still path left
=1 the interpolator is empty on the appropriate axis.
Due to RESET the flag is set to 1.

☞ *The below flags are effective on all axes, even the ones not selected for being controlled by PLC at flags Y630, ..., Y637.*

I903, I913, ..., I973: 1^{st}, 2^{nd}, ..., 8^{th} axis reference point ready
If the flag
=1 reference point return has already occurred on the appropriate axis.

I907, I917, ..., I977: 1^{st}, 2^{nd}, ..., 8^{th} axis drive ready.
If the flag
=1 digital drive is ready on the appropriate axis
☞ *Warning!*
This flag can only be used with NCT digital servo drives and XMU CAN digital measuring system board!

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I980	
I981	
I982	
I983	
I984	
I985	
I986	
I987	1st main drive ready

I987: 1st main drive ready
If the flag
=1 1st digital main drive is ready.
☞ *Warning!*
This flag can only be used with NCT digital main drives and XMU CAN digital measuring system board!

Flag Identity	Meaning of Flag if Value=1 (TRUE)
I990	
I991	
I992	
I993	
I994	
I995	
I996	
I997	2nd main drive ready

I997: 2nd main drive ready
If the flag
=1 2nd digital main drive is ready.

☞ *Warning!*
This flag can only be used with NCT digital main drives and XMU CAN digital measuring system board!

2.2.2 Flags from PLC to NC (Output Flags)

Reference to an output flag can be done with character Y and three digits similarly to the interface output line:

Ypqr

The first digit must be equal to or greater than 4. The value range of the first digit:

p=4,5,6,7,8,9

The value range of the second digit (q):

q=0,1,2,3,4,5,6,7,8,9

The third one (r) defines the serial number of a bit within the selected byte and is therefore octal. Its value range is

r=0,1,2,3,4,5,6,7

In case of a word operand reference to an output flag group can be made in the PLC program by dropping the last digit:

Ypq

In the followings a full list of output flags is shown:

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y400	Reference point return mode lamp
Y401	Manual handle mode lamp
Y402	Incremental jog mode lamp
Y403	Jog mode lamp
Y404	
Y405	Manual data input mode lamp
Y406	Automatic mode lamp
Y407	Edit mode lamp

The statuses of operation modes must be transferred to the NC through the following flags:

Y400: Reference point return mode lamp
The flag is set to 1 if mode REF has been selected by the operator and enabled by the PLC..

Y401: Manual handle mode lamp
The flag is set to 1 if mode HNDL has been selected by the operator and enabled by the PLC.

Y402: Incremental jog mode lamp
The flag is set to 1 if mode INCR has been selected by the operator and enabled by the PLC.

Y403: Jog mode lamp
The flag is set to 1 if mode JOG has been selected by the operator and enabled by the PLC.

Y404: -

Y405: Manual data input mode lamp
The flag is set to 1 if mode MDI has been selected by the operator and enabled by the PLC.

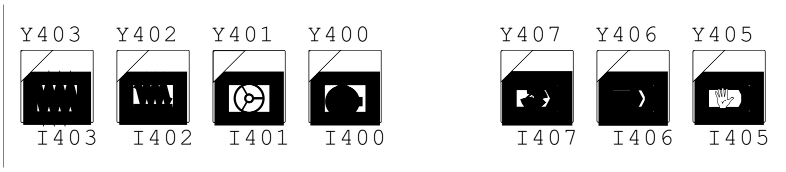
Y406: Automatic mode lamp
The flag is set to 1 if mode AUTO has been selected by the operator and enabled by the PLC.

Y407: Edit mode lamp
The flag is set to 1 if mode EDIT has been selected by the operator and enabled by the PLC.

The operation mode states must be kept in 1 till the given mode is active. The operator’s manual of the given control describes the operation modes that can be activated simultaneously. According to this the PLC programmer has to recognize the conflicting modes.

The states of the modes are displayed by the control on the softkeys on screens OPERATOR’S PANEL, POSITION and CHECK after selecting action menu MODES according to flags Y400, ..., Y407.

If machine control board 2 is applied on control the lamps of mode buttons are switched on or off also on the basis of flags Y400, ..., Y407.



Arrangement of mode buttons on machine control board 2

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y410	1st axis selected lamp
Y411	2nd axis selected lamp
Y412	3rd axis selected lamp
Y413	4th axis selected lamp
Y414	5th axis selected lamp
Y415	6th axis selected lamp
Y416	7th axis selected lamp
Y417	8th axis selected lamp

The following flags must be switched on to select an axis for either jog or incremental jog mode or manual handle movement, as well as for reference point return.

Y410, ..., Y417: 1st, ..., 8th axis selected lamp
The flag is set to one if the 1st, ..., 8th axis has been selected by the operator and enabled by the PLC

The operator's manual of the given control describes if more than one axis can be selected at the same time. If needed, simultaneous selection of more than one axis has to be forbidden by the PLC programmer.

The selected axis is displayed by the control on screens OPERATOR'S PANEL, POSITION and CHECK after selecting action menu AXIS according to flags Y410, ..., Y417.
If machine control board 2 is applied and maximum 4 axes are built in the machine, there is no need for axis selection in jog and increment modes, because the built-in jog buttons are adequate for selecting at most 4 axes. If there are more than 4 axes in the machine, one of the 8 free-purpose buttons must be used in order to select the 4th, 5th, etc. axis. In this case the lamp (Y480, ..., Y487) of the selected button on control panel and the appropriate flag Y410, ..., Y417 towards the NC must be switched on or off parallel.
In handwheel mode if maximum 4 axes are built in the machine, axis direction buttons can also be used for selecting the 1st, ..., 4th axis. In this case the lamp (Y450, ..., Y457) of the selected axis e.g. X belonging to both directions (+ and –) is expedient to be switched on by means of axis direction button together with the appropriate flag Y410, ..., Y417 towards the NC. If there are more than 4 axes in the machine, jog buttons of 3 axes can be used as hereinabove, while selection of further axes can be done as discussed for jog and increment modes.
If separate handwheels are being built on each axis (on axes X and Z in case of turning machines or on axes X, Y and Z in case of milling machines) X, Y, or Z handwheel is effective only in case if all axis select flags (lamps) are low (0). If one of them is on (1) only the common handwheel that can be used for all axes is effective.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y420	1 increment lamp
Y421	10 increment lamp
Y422	100 increment lamp
Y423	1000 increment lamp
Y424	
Y425	
Y426	Automatic tool length measure softkey lamp
Y427	JOG rapid traverse lamp

The increment flags are used in modes INCR and HNDL.

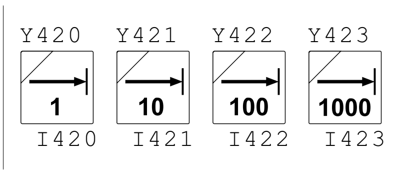
Y420: 1 increment lamp
It signals 1 increment step length in incremental jog.

Y421: 10 increment lamp
It signals 10 increment step length in incremental jog.

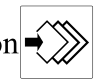
Y422: 100 increment lamp
It signals 100 increment step length in incremental jog.

Y423: 1000 increment lamp
It signals 1000 increment step length in incremental jog.

Only one increment flag can be active at a time, of which the PLC programmer must take care. The selected increment size is displayed by the control on screens OPERATOR'S PANEL, POSITION and CHECK after selecting action menu INCREMENT according to flags Y420, ..., Y427.
If machine control board 2 is applied on control the lamps of the selected increment size are switched on or off also on the basis of flags Y420, ..., Y427.

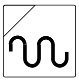


Arrangement of increment selector buttons on machine control board 2

Y426: Automatic tool length measure softkey lamp
In case of lathe controls select F4 T. LENG MEASUR (length offset measurement) within screen OFFSETS F5. Press action menu button  . Softkey AUTO MEAS F3 appears among the actions. Flag Y426 shows the on or off state of this function. **It can only be set to 1 in jog mode**.
If the flag is set to 1 and screen LENGTH MEAS is active as the effect of jog buttons (even if feed rate switch state is 0%) the selected axis moves at the rate defined at parameter 8022 **G37FD** until the button belonging to the selected direction of the tool sensor is pressed (flags Y580, ..., Y583).

Y427: JOG rapid traverse lamp

The flag is set to 1 if the operator has activated JOG rapid traverse push-button and 0 if it has been inactivated.

If machine control board 2 is applied on control, flag Y427 is at the same time the lamp of rapid traverse button  .

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y430	JOG X axis + direction selected
Y431	JOG Y axis + direction selected
Y432	JOG Z axis + direction selected
Y433	JOG + direction selected
Y434	JOG X axis − direction selected
Y435	JOG Y axis − direction selected
Y436	JOG Z axis − direction selected
Y437	JOG − direction selected

Y433, Y437: JOG +/− direction selected
In both cases the axis in compliance with the state of axis switch (defined at flag Y410, ..., Y417) moves in positive or negative direction until the appropriate flag is set to 1.

Y430, Y431, Y432, Y434, Y435, Y436: JOG X, Y, Z axis +/− direction selected
The flag is set to 1 when the appropriate axis is in motion.

In case of JOG push-buttons four axes can be selected at the same time.

On machine control board 2 all jog buttons have a lamp switched through flags Y450, ...Y457. When a jog button is pressed (I430, ..., I437) the appropriate flag Y430, ..., Y437 in accordance with the button caption must obligatorily be switched on towards the NC as well as it is also expedient to switch the lamp belonging to the appropriate button, signaling the push (Y450, ...Y457 on).

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y440	Test lamp
Y441	Machine lock lamp
Y442	Dry run lamp
Y443	Block restart lamp
Y444	Block return lamp
Y445	Conditional stop lamp
Y446	Conditional block 1 lamp
Y447	Block by block mode lamp

The statuses of different states must be transferred to the NC through the following flags:

Y440: Test lamp
If the flag is set to 1 no movement command is sent to the measuring system. In this case function commands must not be received by the PLC from NC. Use the lamp in toggle mode for each Test push-button action.

Y441: Machine lock lamp
If the flag is set to 1 no movement command is sent to the measuring system. In this case function commands must not be received by the PLC from NC. Use the lamp in toggle mode for each Machine lock push-button action.

Y442: Dry run lamp
If the flag is set to 1 all feed motion is executed at the rate specified at parameter group **4741 FEEDMAX**. Use the lamp in toggle mode for each Dry run push-button action.

Y443: Block restart lamp
If the flag is set to 1 by pressing START the block is reloaded and re-executed from beginning. Use the lamp in toggle mode for each Block restart push-button action.

Y444: Block return lamp
If the flag is set to 1 by pressing START the machining is continued from the interruption point of the block. Use the lamp in toggle mode for each Block return push-button action.

Behind flags Y443 and Y444 there are conflicting functions, so the PLC programmer should make sure that only one of the two flags is set to 1.

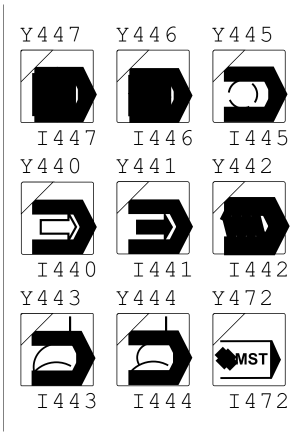
Y445: Conditional stop lamp
If the flag is set to 1 function M01 is executed. Use the lamp in toggle mode for each Conditional stop push-button action.

Y446: Conditional block 1 lamp
If the flag is set to 1 all blocks starting with /1 are skipped. Use the lamp in toggle mode for each Conditional block push-button action.

Y447: Single block mode lamp
If the flag is set to 1 the control stops after every block execution and registers STOP state.Use the lamp in toggle mode for each Single block push-button action.

The states are displayed by the control on screens OPERATOR’S PANEL, POSITION and CHECK after selecting action menu STATE according to flags Y440, ..., Y447.

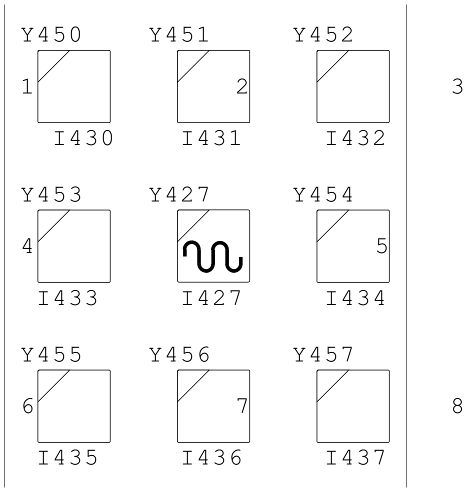
If machine control board 2 is applied on control the lamps of condition buttons are switched on or off also on the basis of flags Y420, ..., Y427.



Arrangement of state switches on machine control board 2

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y450	JOG 1 push-button lamp
Y451	JOG 2 push-button lamp
Y452	JOG 3 push-button lamp
Y453	JOG 4 push-button lamp
Y454	JOG 5 push-button lamp
Y455	JOG 6 push-button lamp
Y456	JOG 7 push-button lamp
Y457	JOG 8 push-button lamp

Y450, ..., Y457: JOG 1, ..., JOG 8 push-button lamp
If the machine control board 2 is used (Y532=1) the lamps of buttons JOG 1, ..., JOG 8 can be switched on through flags Y450, ..., Y457.



Arrangement of jog buttons on machine control board 2

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y460	1st axis lock selected
Y461	2nd axis lock selected
Y462	3rd axis lock selected
Y463	4th axis lock selected
Y464	5th axis lock selected
Y465	6th axis lock selected
Y466	7th axis lock selected
Y467	8th axis lock selected

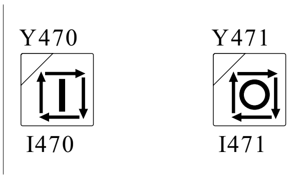
Y460, ..., Y467: 1st, ..., 8th axis lock selected
If the flag is set to 1 no movement command is sent to the measuring system of the appropriate axis. The axis arrangement corresponds to the physical axis arrangement set at parameter group **4281 AXIS** .

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y470	Start state lamp
Y471	Stop state lamp
Y472	Function lock lamp
Y473	Manual handle feed
Y474	M3 lamp of machine control board 2
Y475	M4 lamp of machine control board 2
Y476	M5 lamp of machine control board 2
Y477	RESET from PLC

Y470: Start state lamp
Y471: Stop state lamp
The enabled combinations, which must be ensured by the operator:

Y471	Y470	
0	**0**	neither
0	**1**	START state
1	**0**	STOP state
1	**1**	inhibited state

If machine control board 2 is applied on control the lamps of START and STOP buttons are switched on or off also on the basis of flags Y470, Y471.



Arrangement of start and stop buttons on machine control board 2

Y472: Function lock lamp
If the flag is set to 1 no function must be received by the PLC from the NC as well as sent to the machine.

If machine control board 2 is applied on control the lamp of function lock button  is switched on or off also on the basis of flag Y472.

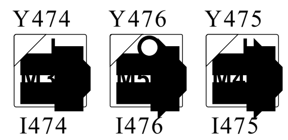
Y473: Manual handle feed
If the flag is set to 1 in automatic or manual data input mode feed is received from the mutual handwheel (available for all axes). Slides move faster or slower on the programmed path in function of the increment set on flags Y420, ..., Y422. It moves forward (positive direction) or backward (negative direction) on the path in function of the direction of turning.

Y474: M3 lamp of machine control board 2
In state M3 the flag must be set to 1 that lights up M3 lamp. It may be used only in case of machine control board 2 (Y532=1).

Y475: M4 lamp of machine control board 2
In state M4 the flag must be set to 1 that lights up M4 lamp. It may be used only in case of machine control board 2 (Y532=1).

Y476: M5 lamp of machine control board 2
In state M5 the flag must be set to 1 that lights up M5 lamp. It may be used only in case of machine control board 2 (Y532=1).

On machine control board 2 the rotation states (M3, M4) or stop state (M5) of spindle can be signaled with the help of the above lamps.

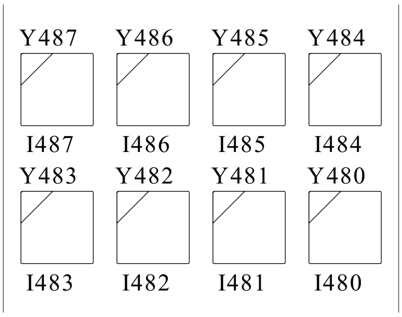


Arrangement of spindle rotation buttons on machine control board 2

Y477: RESET from PLC
In case the the data input keyboard is operated by the PLC (Y537=1), the PLC program can activate reset by setting flag Y477 to1. The effect of reset has to be awaited, since it is the result of a longer process. E.g.: if flag I537 is 1 (message on screen) flag Y477 must be kept set to 1 until the message disappears.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y480	1st user's push-button's lamp of machine control board 2
Y481	2nd user's push-button's lamp of machine control board 2
Y482	3rd user's push-button's lamp of machine control board 2
Y483	4th user's push-button's lamp of machine control board 2
Y484	5th user's push-button's lamp of machine control board 2
Y485	6th user's push-button's lamp of machine control board 2
Y486	7th user's push-button's lamp of machine control board 2
Y487	8th user's push-button's lamp of machine control board 2

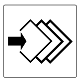
Y480, ..., Y487: 1st, ..., 8th user's push-button's lamp of machine control board 2
These flags are the lamps of free-purpose buttons mounted on machine control board 2, the function of which is defined by the PLC programmer.



Arrangement of free-purpose buttons on machine control board 2

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y490	
Y491	
Y492	
Y493	
Y494	
Y495	
Y496	
Y497	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y500	PLC defined softkey 1 lamp
Y501	PLC defined softkey 2 lamp
Y502	PLC defined softkey 3 lamp
Y503	PLC defined softkey 4 lamp
Y504	PLC defined softkey 5 lamp
Y505	PLC defined softkey 6 lamp
Y506	PLC defined softkey 7 lamp
Y507	PLC defined softkey 8 lamp

If Y524=1 (PLC switches from SW control panel) the signal of the 8 free-purpose softkey buttons offered by the NC is transferred by the NC through flags I500, ..., I507. (If Y524=0 these buttons are not offered by the NC.) The button captions can be determined by the PLC programmer in module :197.
The buttons are available if one of screns OPERATOR'S PANEL, POSITION or CHECK is selected.
Afterewards action menu F^6 MACHINE must be selected after pressing action menu button  . In this case the captions defined by the PLC programmer in module :197 appear on the softkeys.

These statuses are the lamps of push-buttons transferred through flags I500, ..., I507.

Y500, ..., Y507: PLC defined softkey 1, ..., 8 lamp

In order to switch on the status the appropriate flag must be set to1.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y510	Conditional block 2 skip
Y511	Conditional block 3 skip
Y512	Conditional block 4 skip
Y513	Conditional block 5 skip
Y514	Conditional block 6 skip
Y515	Conditional block 7 skip
Y516	Conditional block 8 skip
Y517	Conditional block 9 skip

Y510, ..., Y517: Conditional block 2, ..., 9 skip
If the flag is set to 1 it skips every block starting with /n (n=2, ..., 9).

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y520	Mode selection with softkeys
Y521	Axis selection with softkeys
Y522	Increment selection with softkeys
Y523	State selection with softkeys
Y524	PLC defined buttons with softkeys
Y525	R% (rapid traverse override) with softkeys
Y526	S% (spindle override) with softkeys
Y527	F% (feed override) with softkeys

With the help of the below output flags the PLC programmer decides, which machine action groups are activated by means of softkeys, and which are only used for displaying.

Y520: Mode selection with softkeys
If the flag is set to 1 the operation modes are activated by means of softkeys. PLC receives state of the softkeys through flags I400, ..., I407. The valid statuses of operation modes are sent to the NC through flags Y400, ..., Y407.

Y521: Axis selection with softkeys
If the flag is set to 1 the axes are activated by means of softkeys. PLC receives state of the axes through flags I410, ..., I417. The valid statuses of axes are sent to the NC through flags Y410, ..., Y417.

Y522: Increment selection with softkeys
If the flag is set to 1 the increments are activated by means of softkeys. PLC receives the states through flags I420, ..., I427. The valid statuses of increments are sent to the NC through flags Y420, ..., Y427.

Y523: State selection with softkeys
If the flag is set to 1 the states are activated by means of softkeys. PLC receives the states through flags I440, ..., I447. The valid statuses of conditions are sent to the NC through flags Y440, ..., Y447.

Y524: PLC defined buttons with softkeys
If the flag is set to 1 the PLC defined buttons are activated by means of softkeys. The caption of softkeys can be determined by the PLC programmer in module :197.
The length of a caption may be 6 character. The caption texts are separated by commas "," :

:197PLC1,PLC2,PLC3,PLC4,PLC5,PLC6,PLC7,PLC8$

The last string together with module :197 is closed by character $.
PLC receives state of the PLC defined buttons through flags I500, ..., I507. The valid statuses of PLC defined buttons are sent to the NC through flags Y500, ..., Y507.

Y525: R% (rapid traverse override) with softkeys
If the flag is set to 1 the rapid traverse override states are activated by means of softkeys. PLC receives values of R% through register RH039. The valid R% value is sent to the NC through register RH089.

Y526: S% (spindle override) with softkeys
If the flag is set to 1 the spindle override is activated by means of softkeys. PLC receives value of the S% through register RH029. The valid S% value is sent to the NC through register RH079.

Y527: F% (feed override) with softkeys
If the flag is set to 1 the feed override is activated by means of softkeys. PLC receives value of the F% through register RH028. The valid F% value is sent to the NC through register RH078.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y530	Jog buttons from NC keyboard
Y531	Selection of machine control board 1
Y532	Selection of machine control board 2
Y533	
Y534	
Y535	
Y536	Valid push-button code in register RH099
Y537	Data input from PLC

Y530: Jog buttons from NC keyboard
If the flag is set to 1 in continuous and incremental JOG modes the numeric keyboard is to be used. Interpretation of the keys is as follows:

<4>: movement in negative direction (-),
<5>: rapid traverse movement
<6>: movement in positive direction (+).

The appropriate axis must be set with the help of softkeys in AXES action menu, while in mode INCR the increment size in the INCREMENT action menu.
The selected axis direction is sent by the NC to the PLC through flags I433, ..., I437 In order to start the motion flags Y433, ..., Y437 must be set by the PLC. The selected rapid traverse is transferred through flag I427, which is to be sent by the PLC to the NC through flag Y427.

Y531: Selection of machine control board 1
On machine control board 1 the following buttons and rotary switches can be found:

spindle rotation and spindle stop buttons <M3>, <M4>, <M5>,
spindle override buttons <->, <100%>, <+>,
<feed override> rotary switch,
jog axis direction buttons <-X>, <+X>, <-Y>, <+Y>, <-Z>, <+Z>, <->, <+>,
<rapid traverse> button

As a result of the above list flags Y520, ..., Y530 must be set in case of using machine control board 1 in the following way:

Y520=1: mode selection with softkeys
Y521=1: axis selection with softkeys
Y522=1: increment selection with softkeys
Y523=1: state selection with softkeys
Y524=0, or 1: PLC defined buttons with softkeys
Y525=1: rapid traverse override with softkeys
Y526=0: spindle override from machine control board 1
Y527=0: feed override from machine control board 1
Y530=0: jog buttons from machine control board 1

- The spindle override value is now modified from machine control board 1, but in this case the PLC receives the current value also in register RH029, which is to be copied into register RH079.

- This also refers to feed override (registers RH028 - RH078).
- With jog axis direction buttons (1), ..., (8) in effect flags I430, ..., I437 are control ed on. These flags must be copied to the appropriate flags Y430, ..., Y437.

Y532: Selection of machine control board 2
If machine control board 2 is applied the below flags must be obligatorily filled out in the following way:

Y520=0: mode selection not from SW control panel
Y521=0 or 1: axis selection optionally from free-purpose buttons of machine control board 2 (Y521=0) or from SW control panel (Y521=1)
Y522=0: increment selection not from SW control panel
Y523=0: state selection not from SW control panel
Y524=0 or 1: PLC switches optionally from SW control panel
Y525=0 ory 1: rapid traverse override selection optionally from keyboard or SW control panel
Y526=0: spindle override selection from keyboard push-buttons
Y527=0: feedrate override selection from keyboard switch
Y530=0: jog buttons and rapid traverse button from keyboard
Y432=1: selecting machine control board 2

- Now the spindle override value is modified by the push-buttons on machine control board 2, but even in this case the PLC receives the current value in register RH029, which must be copied into register RH079.
- Likewise in case of feed rate override (registers RH028 - RH078).
- As the effect of jog buttons (1), ..., (8) flags I430, ..., I437 are switched on. These flags must be copied to the appropriate flags Y430, ..., Y437.

Y536: Valid push-button code in register RH099
If flag Y537 is 1, the NC does not acquires push-button codes of data input keyboard from the control panel but from PLC by reading register RH099. If flag Y536 is set to 1 the PLC has written one valid push-button code into register RH099. Push-button codes can be found in chapter 6.5 Listing of Push-button Codes on page 237.

Y537: Data input from PLC
If the flag is 0 the NC acquires the push-button codes from the NC or data input keyboard. If the flag is set to 1 push- button on data input keyboard is uneffective, the push-button codes are read by the NC from register RH099 when flag Y536 is set to 1. As the effect of the flag being set to 1 the screen takes the absolute position (RH027=0102h), while the softkeys take the screen selection (RH026=0000h) state.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y540	Machine on request
Y541	No input synchronization in module :000
Y542	Feed hold
Y543	General security gate enable
Y544	Interrupt macro call enable
Y545	Free purpose user's timer enable
Y546	Module :002 call enable
Y547	FIN: functions executed by PLC

Y540: Machine on request
MACHINE ON output is a 24V output found on interface board. In case MACHINE ON output is on
- other outputs of interface board receive power supply,
- the measuring system closes position control loop (otherwise it only measures),
- the NC enables any movement start,
- or PLC action.

In case MACHINE ON output is off the NC registers EMG (emergency stop) status and disables all above actions.
PLC may initiate the switch-on of MACHINE ON output by setting machine on request flag Y540 to 1. MACHINE ON output is the logic multiplication of the following signals:

MACHINE ON=(machine on request) and (NC ready) and (no crash), i.e.
I540=(Y540) and (I541) and (I542),

that is machine on request will only be effective if the NC is ready and there is no crash, e.g. servo error. (NC ready signal is switched by NC watchdog timer. If the watchdog timer misses MACHINE ON output is automatically switched off. The control can be restarted only upon power-off.)
If the power-on is successful flag I540 is 1.

Y541: No input synchronization in module :000
If flag Y541 is set to 1 when the PLC starts up (flag I510 is set to 1), synchronizing of interface input lines and input flags in module :000 is suspended, i.e. the PLC acknowledges their states updated in every 20 msec.

Y542: Feed hold
If this flag is set to 1 the feed is stopped on all axes unconditionally, independent of the state of START flag, and the status of G63 (override and stop inhibit) . In case the START flag is set to 1 the feed can only be started if this flag is set to 0. The movement starts with acceleration and stops with deceleration. If flag Y542 is switched on in state G63 (override and stop disabled) the spindle must be stopped in PLC program.

Y543: Enable of opening general security gate
As the effect of command U543 the control enables the opening of general security gate and of special security gates on SECURITY PANEL screen in SETTINGS function group. In order to open each security gate softkey **Open** must be pressed on he above screen.

Y544: Interrupt macro call enable
If the flag is set to 1 the interrupt macro is called as discussed in the programming manual.

Y545: Free purpose user's timer enable
If the flag is set to 1 the NC starts the free purpose user's timer, which measures time till the NC sets it to 0.

Y546: Module :002 call enabled
If the flag is set to 1 module :002 is called in every t msec (see: chapter 1.2 on page 8).

Y547: FIN: functions executed (FINished) by PLC
If the PLC has executed all function commands received from NC through flags I520, ..., I531 FIN flag is set to 1. Due to this the control sends commands of the next block to be executed to the interpolator or PLC instantly. In other words at the start of the first call of module :001 following the setting of flag to 1 flags I520, ..., I531 contain the commands of the next block to be executed.

☞ *Warning!*
If flag Y547 is not switched off when receiving a function and on after function execution, then in single block mode, provided the given function is by itself in the block there is no stop at the end of block, because it is also synchronized by READY signal.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y550	1st axis on reference switch
Y551	2nd axis on reference switch
Y552	3rd axis on reference switch
Y553	4th axis on reference switch
Y554	5th axis on reference switch
Y555	6th axis on reference switch
Y556	7th axis on reference switch
Y557	8th axis on reference switch

Y550, ..., Y557: 1st, ..., 8th axis on reference switch
Switching on the flag (U55n) tells the NC that the nth axis is on reference point switch. The PLC programmer must copy the state of reference position switches mounted on the machine to these flags. The axis numbers indicate the physical axis numbers defined at parameter grop **4281 AXIS**.
The NC uses these flags in mode Reference point return if MACHINE type setting is assigned among parameter groups **7261 REFTYPE1**, ..., **7401 REFTYPE8**.

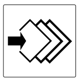
Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y560	1st axis on + limit switch
Y561	2nd axis on + limit switch
Y562	3rd axis on + limit switch
Y563	4th axis on + limit switch
Y564	5th axis on + limit switch
Y565	6th axis on + limit switch
Y566	7th axis on + limit switch
Y567	8th axis on + limit switch

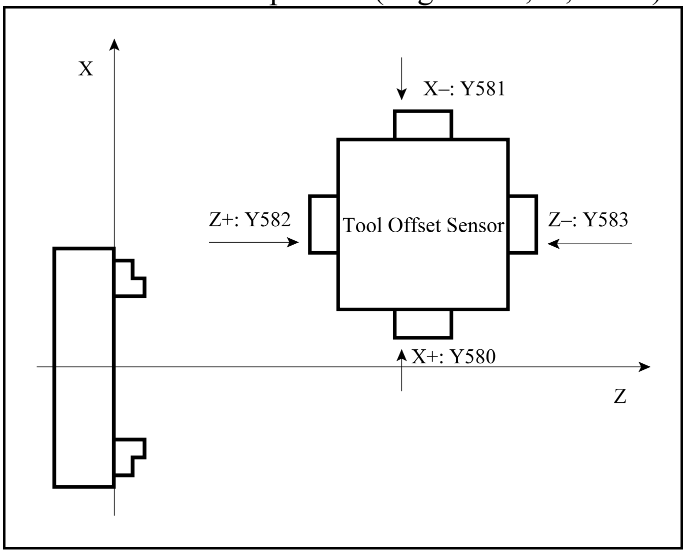
Y560, ..., Y567: 1st, ..., 8th axis on + limit switch
Switching on the flag (U56n) tells the NC that the nth axis is on + limit switch. In this case control displays error message LIMITn+ and forbids all movement in positive direction on the nth axis. Command D56n permits movement in positive direction on the nth axis again.
The axis numbers indicate the physical axis numbers defined at parameter group **4281 AXIS**. The PLC programmer must copy the state of limit switches mounted on the machine to these flags.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y570	1st axis on – limit switch
Y571	2nd axis on – limit switch
Y572	3rd axis on – limit switch
Y573	4th axis on – limit switch
Y574	5th axis on – limit switch
Y575	6th axis on – limit switch
Y576	7th axis on – limit switch
Y577	8th axis on – limit switch

Y570, ..., Y577: 1st, ..., 8th axis on – limit switch
Switching on the flag (U57n) tells the NC that the nth axis is on – limit. In this case control displays error message LIMITn– and forbids all movement in negative direction on the nth axis. Command D57n permits movement in negative direction on the nth axis again.
The axis numbers indicate the physical axis numbers defined at parameter group **4281 AXIS**. The PLC programmer must copy the state of limit switches mounted on the machine to these flags.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y580	Tool offset sensor pressed in X+ direction
Y581	Tool offset sensor pressed in X– direction
Y582	Tool offset sensor pressed in Z+ direction
Y583	Tool offset sensor pressed in Z– direction
Y584	
Y585	
Y586	
Y587	

In case of lathe controls select [F4] T. LENG MEASUR (length offset measurement) within screen OFFSETS [F5]. Press action menu button  . Softkey AUTO MEAS F[3] appears among the actions (flag I426). Flag Y426 shows the on or off state of this function. **It can only be set to 1 in jog mode**. If the key is pressed (Y426=1) as the effect of jog buttons (even if feed rate override switch state is 0%) the selected axis moves at the rate defined at parameter 8022 **G37FD** until the button belonging to the selected direction of the tool sensor is pressed (flags Y580, ..., Y583).



Y580: Tool offset sensor direction X+ pressed
Y581: Tool offset sensor direction X– pressed
Y582: Tool offset sensor direction Z+ pressed
Y583: Tool offset sensor direction Z– pressed

Signals of tool offset sensor are received by 24V interface inputs determined by the machine builder. The signals of these inputs must be copied to the appropriate flags Y580, ..., Y583. The inputs must be requested and copied over and over by means of module :002 for the interest of accurate measuring. The module enabling is expedient to be linked with the LED of automatic tool length measure Y426.

If the tool offset sensor has only one output for all four directions the common output must be copied to the appropriate flag Y580, ..., Y583 by the use of flags Y430, Y434, Y432, Y436 (JOG X+, JOGX–, JOGZ+, JOGZ–).

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y590	Axis 1 synchron slave on
Y591	Axis 2 synchron slave on
Y592	Axis 3 synchron slave on
Y593	Axis 4 synchron slave on
Y594	Axis 5 synchron slave on
Y595	Axis 6 synchron slave on
Y596	Axis 7 synchron slave on
Y597	Axis 8 synchron slave on

Y590, ..., Y597: Axis 1, ..., 8 synchron slave on
Two axes can be synchronized. In this case one of the axises will be the master and the other will be the slave. We can define the master axis of the slave axis with the 1391 SYNCHRON parameter group. The number of the master axis should always be specified at the parameter of the slave axis.
In the case of a milling machine with two spindles, the moving of the table (X axis) is the same for both spindles. The axes of the master spindle should be Y and Z. Then the Y and Z axes are the master axes. The axes of the other, slave spindle should be V and W. Then the V and W axes are the slave axes. If you would like to make two identical workpieces simultaneously, you do not have to make different programs for X, Y, Z and X, V, W, but in the corresponding program with M function the Y-V and the Z-W axes can be connected and synchron cutting can be carried out.
For example:

```
...
M78                  (Disconnection of the synchron axises)
T2
G30 YI0 ZI0 P2       (Y, Z moves to the change position)
M6                   (T2 tool to the master spindle)
G30 VI0 WI0 P2       (V, W moves to the change position)
T52
M6                   (T52  tool to the slave spindle)
G55 G0 X100 Y200     (positioning on the master side)
U100 W200            (positioning to the same position on the slave side)
G43 Z10 H2           (H2 compensation and positioning of Z on the master side)
G43 W10 H52          (H52 compensation and positioning of W on the slave side)
M77                  (Y-V, Z-W turning on synchron function)
...
X_ Y_                (The description of the program with X, Y, Z coordinates.
                     V-Y and W-Z move together)
Z_
...
```

If in the example above
X: is the 1. axis,
Y: is the 2. axis,
Z: is the 3. axis,
V: is the 4. axis,
W: is the 5. axis, then the SYNCHRON parameters are the following:
1394 SYNCHRON4=2 the master of V axis is 2., which is Y and
1395 SYNCHRON4=3 the master of W axis is 3., Z axis

The turning on of the flag (U59n) means to the NC, that the slave axis can start the synchronized functioning with its master axis. In the example above this would mean that the M77 function turns on the flag of the Y593 (V axis) and the flag of the Y594 (W axis), while M78 turns of these flags.
The synchron function works with manual movement as well. The synchron functions till the corresponding flag is on 1.
☞***Warning! The change of Y59n flags can only be made, when the block buffer is emptied! If changing happens by M functions the 022n MSUPRn parameters must be spcified to show the NC that the buffer is to be emptied, or if a subprogram does the changing, G53 should be used in the block before and after the change.***

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y600	Number of program selected for automatic mode in RH050
Y601	Number of program selected for manual data input mode in RH050
Y602	Program run in DNC
Y603	Program run in NCT DNC
Y604	Message strobe
Y605	Open input channel
Y606	Transmittable data in memory
Y607	PLC received data from memory

The same actions can be executed on flags Y600, ..., Y603 as when selecting action menu Run on screen DIRECTORY.

Y600: Number of program selected for automatic mode in RH050
If the flag is set to 1 the program, the number of which is specified in register RH050 is selected for run in automatic mode. The flag must be kept set to 1 until the number written in RH050 can be re-read from register RH031.

Y601: Number of program selected for manual data input mode in RH050
If the flag is set to 1 the program, the number of which is specified in register RH050 is selected for run in manual data input mode. The flag must be kept set to 1 until the number written in RH050 can be re-read from register RH032.

Y602: Program run in DNC
If the flag is set to 1 if program run in DNC without protocol in automatic mode is selected. The flag must be kept set to 1 until the program execution in DNC status flag I602 is set to 1.

Y603: Program run in NCT DNC
If the flag is set to 1 if program run in DNC on the basis of NCT protocol in automatic mode is selected. The flag must be kept set to 1 until the program execution in NCT DNC status flag I603 is set to 1.

Y604: Message strobe
PLC strobes flag Y604 with command U604 and waits until flag I604 turns to 1. Afterwards flag Y604 must be switched off by means of command D604. This pair of flags is for synchronizing manual handle machining executed on PC. (Both manual data input mode and manual handle mode are on: Y405AY401).

Y605: Open input channel
If the PLC program is to initiate data input via an input channel loads registers RH054, ..., RH056, then sets flag Y605 to 1.

Y606: Transmittable data in memory
If the flag is set to 1 the NC sends the contents of the selected memory area (F010, ..., F499) through the selected periphery. Register RH051 contains the start address of valid data, while register RH052 includes the number of bytes to be sent (record length). The number of periphery,

through which the data is to be sent is specified in register RH053. If the NC has sent the data it sets flag I606 to 1. Then PLC should reset flag Y606 and data transfer is terminated.

Y607: PLC received data from memory
If the PLC has worked the data sent by the NC it sets the flag to 1. This means that the selected memory area can be overwritten again. The NC fills the memory area (F010, ..., F499) from the start address given in register RH054 with the byte the number of which is specified in register RH055 through the periphery defined in register RH056. When ready it sets flag I607 to 1. The PLC answers with the help of flag Y607.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y610	1st axis motion disable
Y611	2nd axis motion disable
Y612	3rd axis motion disable
Y613	4th axis motion disable
Y614	5th axis motion disable
Y615	6th axis motion disable
Y616	7th axis motion disable
Y617	8th axis motion disable

Y610, ..., Y617: 1st, ..., 8th axis motion disable
Before the interpolator sends motion command to one of the axes, it asks for motion request on the appropriate axis through flags I610, ..., I617. It waits until the PLC permits the motion command through the appropriate flags Y610, ..., Y617 by means of statement

D61n.

If the motion request has been rejected the statement motion disable (axis clamping, drive enable off, command U61n) can only be executed after the appropriate one has already reached its end position, which can be observed on flags I560, ..., I567. These flags can be used for example for clamping of axes, if a motor drives more axes to set the movable axes, or for synchronizing, if rapid traverse movement implies axis gear setting. The axis numbers indicate the physical axis numbers defined at parameter group **4281 AXIS**.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y620	1st axis loop open
Y621	2nd axis loop open
Y622	3rd axis loop open
Y623	4th axis loop open
Y624	5th axis loop open
Y625	6th axis loop open
Y626	7th axis loop open
Y627	8th axis loop open

Y620, ..., Y627: 1st, ..., 8th axis loop open
With statement D62n in effect the position control loop is closed on the nth axis of the control, command signal goes out to the drives. The NC checks the state of position control loop continuously, and if needed, displays error message SERVOn, FEEDBACKn.
With statement U62n in effect the position control loop is opened on the nth axis of the control, command signal transfer does not occur, but the current position of the axis is measured and registered by the control. Servo and feedback error check is not done, but it keeps on checking the state of encoder, and if needed, displays error ENCODERn.
Before switching position control loop closed off the stopped state of the given axis must be checked, i.e. whether flag I56n is true.

Attention! If position control loop is opened then closed during program run after closing it the axis must always go to reference point otherwise position will be erroneous.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y630	1st axis motion by PLC
Y631	2nd axis motion by PLC
Y632	3rd axis motion by PLC
Y633	4th axis motion by PLC
Y634	5th axis motion by PLC
Y635	6th axis motion by PLC
Y636	7th axis motion by PLC
Y637	8th axis motion by PLC

Y630, ..., Y637: 1st, ..., 8th axis motion by PLC.
The interpolator may receive motion commands from both NC and PLC.
If motion commands are to be initiated by the NC on one of the axes the appropriate physical axis number must be entered beside the logic axis selections at parameter group **4281 AXIS**. For example if 4281 X = 1, then the commands written at address X are issued to the 1st physical axis by the interpolator. The appropriate flags Y630, .., Y637 of in such way selected axes must be set to 0.
If motion commands are to be initiated by the PLC on one of the axis the appropriate output flag Y630, .., Y637 must be set to 1. For no logic axis selection belongs to this kind of axis (no axis with this number was selected at parameter group 4281 AXIS) there is no room for this axis in the position display, and what is more these axes have no names. The parametering of axes controlled by the PLC correspond to those controlled by the NC.
The interpolator may receive simultaneous motion command from both sides, the NC and the PLC. It executes the two motion commands parallel and independently. E.g. milling is done with NC axes while a PLC axis rotates the magazine.
Feed and rapid traverse override as well as command FEED HOLD are all effective on PLC axes the same as on NC axes.
For axes selected for the NC (altogether) the interpolator status can be read at flags I550, ..., I557. There is interpolator status for each PLC axis, for these work independent of each other and cannot be connected for path generation. These statuses can be read at flags I900, ..., I977. Positions of PLC axes can be read at registers RH100, ..., RH139. PLC motion commands can be issued through strobe flags Y900, ..., Y977 and registers RH100, ..., RH139.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y640	1st axis encoder check off
Y641	2nd axis encoder check off
Y642	3rd axis encoder check off
Y643	4th axis encoder check off
Y644	5th axis encoder check off
Y645	6th axis encoder check off
Y646	7th axis encoder check off
Y647	8th axis encoder check off

Y640, ..., Y647: 1st ,..., 8th axis encoder check off
On the axes, on which broken encoder wire check is enabled by parameter 440n ENCDn (=0) encoder check can be switched off by setting the appropriate flag to 1.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y650	Active spindle rotates
Y651	1st spindle orientation request
Y652	1st spindle command signal enable
Y653	1st spindle command signal with + polarity
Y654	1st spindle binary command signal output (spindle JOG)
Y655	Synchronize 1st spindle to the 2nd
Y656	1st spindle synchronization in counter direction
Y657	1st spindle orientation in the shorter direction

Y650: Active spindle rotates
The interpolator sets flag I553 (spindle rotation request) to 1 before starting one of commands G1, G2, G3, G33 provided the spindle does not take part in the interpolation (the spindle loop is not closed, I651=0 and I661=0).
The interpolation is started when flag Y650 is set to1 (statement U650).
In case of miscellaneous blocks (containing both interpolation and functions) this flag can be used for synchronizing interpolator and PLC activities, for in the course of block execution the interpolator and the PLC receive their part of the block at the same time. (For activities see flag I553.)
The PLC programmer must be aware that the flag is to be sent to the NC without working the spindle even when in case of these blocks the spindle need not be on due to technological circumstances (e.g. there is a touch probe in the spindle).

Y651: 1st spindle orientation request
If the spindle drive can be positioned, i.e. if the position control loop can be closed through the spindle drive, closing and orientation of spindle control loop can be required from the NC by switching flag Y651 on by means of statement

U651.

The PLC programmer determines the speed of zero pulse search through 1st spindle jog command signal register RH061. If the orientation is finished (spindle is set on the zero pulse of encoder) the NC acknowledges the executed command by switching input flag I651 on.

Y652: 1st spindle command signal enable
By setting this flag to 1 the command signal ramping is started.

Y653: 1st spindle command signal with + polarity
The NC always takes the value entered into register RH060 as a positive number (+). The polarity of spindle command signal can be defined by switching flag Y653 to the appropriate state.:

With statement U653 in effect the spindle command signal has positive polarity,
With statement D653 in effect the spindle command signal has negative polarity.

Y654: 1st spindle binary command signal output (spindle JOG)
If the flag is set to 0 command signal transfer is done from register RH060 by taking polarity flag Y653 and range limits set at parameters into account.
If the flag is set to 1 command signal transfer is done in binary form from register RH061. In case of +10V the value to be entered into the register is 7FFFh, while in case of -10V it is 8000h.

Y655: Synchronize 1st spindle to the 2nd

If the 1st spindle is to be synchronized to the 2nd one a command signal must be output to the 1st spindle via register RH060 or RH061 equal to to the revolution of the 2nd one and in the same or in the counter direction.

After I656 n=n_S flag has been set to 1 set flag Y655 to 1 and wait for signal I651 (spindle loop closed) to be turned to 1.

– As a first step the zero pulse of the 1st spindle is closed to that of the 2nd one in the distance defined by parameter 5402 SPSHIFT1. The gain of the control loop is specified by parameter 5401 SYNCHR1. Then

– the NC closes the position control loop (I651=1) and from now on the pulses of the 2nd spindle encoder become the input of the position control loop of the 1st spindle and for it the SERVO parameters indexed by S1 are valid. If parameter 4509 FEEDFORWS1 is set to 128 the zero pulse of the 2nd spindle is followed up with minimal error specified by parameter 5402 SPSHIFT1.

Y656: 1st spindle synchronization in counter direction

If the value of this flag is 0 the NC rotates the 1st spindle in the same direction as that of the 2nd spindle otherwise in counter direction.

Y657: 1st spindle orientation in the shorter direction

PLC flag	Parameter	Spindle movement during orientation
Y657=0	7209 ZPULSS1=0	The spindle searches the zero pulse always in the shorter direction, independently of the value written in register RH061 (sign of the binary number)
	7209 ZPULSS1=1	The spindle always moves to the zero pulse in the direction specified by the value of register RH061
Y657=1		The spindle searches the zero pulse always in the shorter direction, independently of the value written in register RH061

As a rule of thumb execution of command M19 must be specified if the spindle loop is open previously, value of Y657 is 0 if the spindle loop is closed Y657=1.

Explanation: In fine boring cycle G76 spindle must be oriented in the direction of spindle rotation, otherwise rotation in the opposite direction scrapes the surface of the bore or the tool tip can be damaged. In rigid tapping cycles G84.2, G84.3 if a series of taps are to be carried out repeteated orientation is made at closed spindle loop and orientation in the shorter direction can save time.

Attention: Parameter 7209 ZPULSS1 must be set to 1 if the pulses of the spindle encoder are emulated by the spindle drive. Beyond this it is advised to set it to 1 because of the above mentioned machining reasons.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y660	2nd spindle is active
Y661	2nd spindle orientation request
Y662	2nd spindle command signal enable
Y663	2nd spindle command signal with + polarity
Y664	2nd spindle binary command signal output (spindle JOG)
Y665	Synchronize 2nd spindle to the 1st
Y666	2nd spindle synchronization in counter direction
Y667	2nd spindle orientation in the shorter direction

Y660: 2nd spindle is active
The spindle to which commands M3, M4, M5, M11, ..., M18, M19, S are ececuted by the PLC is considered to be the active one. PLC program specifies the active spindle through flag Y660. If flag Y660 is low the first if it is high the second spindle is active. The NC always calculates the following values according to the active spindle:

displays the spindle revolution,
monitors the spindle speed fluctuation,
calculates feed per revolution according to the encoder of the active spindle,
displays the spindle gear range from RH063 or RH068 register and
the rotation state from RH062 or RH067 register.

Both spindles can be rotated at the same time, e.g.: During synchronization. The NC can handle both spindles parallel that is the

I650, I660; I651, I661; I652, I662; I656, I666, I657, I667 input flags
Y651, Y661; Y652, Y662; Y653, Y663; Y654, Y664 output flags
RH010, RH015; RH011, RH016 input registers and
RH060, RH065; RH061, RH066; RH062, RH067; RH063, RH068 output registers.

Y661:2nd spindle orientation request
If the spindle drive can be positioned, i.e. if the position control loop can be closed through the spindle drive, closing and orientation of spindle control loop can be required from the NC by switching flag Y661 on by means of statement

U661.

The PLC programmer determines the speed of zero pulse search through 2nd spindle jog command signal register RH066. If the orientation is finished (spindle is set on the zero pulse of encoder) the NC acknowledges the executed command by switching input flag I661 on.

Y662: 2nd spindle command signal enable
By setting this flag to 1 the command signal ramping is started.

Y663: 2nd spindle command signal with + polarity
The NC always takes the value entered into register RH065 as a positive number (+). The polarity of spindle command signal can be defined by switching flag Y663 to the appropriate state.:

With statement U663 in effect the spindle command signal has positive polarity,
With statement D663 in effect the spindle command signal has negative polarity.

Y664: 2nd spindle binary command signal output (spindle JOG)
If the flag is set to 0 command signal transfer is done from register RH065 by taking polarity flag Y663 and range limits set at parameters into account.
If the flag is set to 1 command signal transfer is done in binary form from register RH066. In case of +10V the value to be entered into the register is 7FFFh, while in case of -10V it is 8000h.

Y665: Synchronize 2nd spindle to the 1st
If the 2nd spindle is to be synchronized to the 1st one a command signal must be output to the 2nd spindle via register RH065 or RH066 equal to to the revolution of the 1st one and in the same or in the counter direction.
After I666 $n=n_S$ flag has been set to 1 set flag Y665 to 1 and wait for signal I661 (spindle loop closed) to be turned to 1.
– As a first step the zero pulse of the 2nd spindle is closed to that of the 1st one in the distance defined by parameter 5422 SPSHIFT2. The gain of the control loop is specified by parameter 5421 SYNCHR2. Then
– the NC closes the position control loop (I661=1) and from now on the pulses of the 1st spindle encoder become the input of the position control loop of the 2nd spindle and for it the SERVO parameters indexed by S2 are valid. If parameter 4510 FEEDFORWS2 is set to 128 the zero pulse of the 1st spindle is followed up with minimal error specified by parameter 5422 SPSHIFT2.

Y666: 2nd spindle synchronization in counter direction
If the value of this flag is 0 the NC rotates the 2nd spindle in the same direction as that of the 1st spindle otherwise in counter direction.

Y667: 2nd spindle orientation in the shorter direction

PLC flag	Parameter	Spindle movement during orientation
Y667=0	7210 ZPULSS2=0	The spindle searches the zero pulse always in the shorter direction, independently of the value written in register RH066 (sign of the binary number)
	7210 ZPULSS2=1	The spindle always moves to the zero pulse in the direction specified by the value of register RH066
Y667=1		The spindle searches the zero pulse always in the shorter direction, independently of the value written in register RH066

As a rule of thumb execution of command M19 must be specified if the spindle loop is open previously, value of Y667 is 0 if the spindle loop is closed Y667=1.
Explanation: In fine boring cycle G76 spindle must be oriented in the direction of spindle rotation, otherwise rotation in the opposite direction scrapes the surface of the bore or the tool tip can be damaged. In rigid tapping cycles G84.2, G84.3 if a series of taps are to be carried out repeteated orientation is made at closed spindle loop and orientation in the shorter direction can save time.
Attention: Parameter 7210 ZPULSS2 must be set to 1 if the pulses of the spindle encoder are emulated by the spindle drive. Beyond this it is advised to set it to 1 because of the above mentioned machining reasons.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y670	1st analog command signal with + polarity
Y671	1st analog command signal output binary
Y672	2nd analog command signal with + polarity
Y673	2nd analog command signal output binary
Y674	Piston turning
Y675	Chopping On
Y676	1st analog command signal output enable
Y677	2nd analog command signal output enable

Y670, Y672: 1st, 2nd analog command signal with + polarity
The command polarity of the 1st and 2nd analog output signals can be defined by switching flags Y670, Y672 to the appropriate state, provided command signal transfer by scaling from registers RH080, RH085:

With statement U670, U672 in effect the command signal has positive polarity,
With statement D60, D672 in effect the command signal has negative polarity.

Y671, Y673: 1st, 2nd analog command signal output binary
Command signal transfer of the 1st and 2nd analog output is done binarily according to the value written in output registers RH081, RH086.

If Y671=0 or Y673=0 the NC scales the value written into register RH080 or RH085 according to the appropriate parameters, it takes the output override value into account, ramps command signal output according to parameter ACC or DCC and thus outputs the command signal.
If Y671=1 or Y673=1 the NC transfers the value written into register RH081 or RH086 as command signal directly, without the above calculation.

Y674: Piston turning
If the flag is turned on (1) the control enters piston turning mode configured by registers RH190, ..., RH195. Before turning the flag off (0) it is recommended to reset ovality registers (RH192, RH193) to 0 and wait until oscillation of axis doing ovality stops. Then flag Y674 can be turned off. This function can be used with special mechanism developed for piston turning sold by NCT.

Y675: Chopping On
If PLC sets the flag NC starts chopping function the way defined in parameter subgroups 0281 CHOPAXF and 0301 CHOPPOS. Chopping can be started by programming command G81.1 in part program or by turning on a button mounted on machine control panel.
If PLC resets flag Y675, NC moves chopping axis from lower dead point to point R and stops it.

Y676, Y677: 1st, 2nd analog command signal output enable
The appropriate voltage is transferred to the output only in case the appropriate flag is set to 1.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y680	
Y681	
Y682	
Y683	
Y684	
Y685	
Y686	
Y687	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y690	
Y691	
Y692	
Y693	
Y694	
Y695	
Y696	
Y697	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y700	1st indexed message request
Y701	2nd indexed message request
Y702	3rd indexed message request
Y703	4th indexed message request
Y704	5th indexed message request
Y705	6th indexed message request
Y706	7th indexed message request
Y707	8th indexed message request

Y700, ..., Y707: 1st, ..., 8th indexed message request
8 different user messages, indexed according to the contents of register RH090, ..., RH097 can be displayed on the screen containing user messages with the help of flags Y700, ..., Y707. Of the maximum 8 messages only one, displayed in the 2nd line of screen, is active. (For reading the active message there is no need to switch over to the screen containing the user messages.)
The active message can be read at flags I700, ..., I707, of which the state of only one can be TRUE. The PLC programmer must take care of canceling the messages. E.g. if one message is for tool replacement it is useful to cancel the active message by means of START button. A message flag can be canceled (DY70n) before it becomes active in case the reason of the message has ceased. Naturally in this case it also is deleted from the screen listing the messages.
The message string must be entered into module :198. The strings are separated by commas ",". The end of module together with the last message is indicated by character $:

:198MESSAGE1,MESSAGE2,...,MESSAGE8$

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y710	1st message request
Y711	2nd message request
Y712	3rd message request
Y713	4th message request
Y714	5th message request
Y715	6th message request
Y716	7th message request
Y717	8th message request

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y790	65th message request
Y791	66th message request
Y792	67th message request
Y793	68th message request
Y794	69th message request
Y795	70th message request
Y796	71st message request
Y797	72nd message request

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y800	73rd message request
Y801	74th message request
Y802	75th message request
Y803	76th message request
Y804	77th message request
Y805	78th message request
Y806	79th message request
Y807	80th message request

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y890	145th message request
Y891	146th message request
Y892	147th message request
Y893	148th message request
Y894	149th message request
Y895	150th message request
Y896	151st message request
Y897	152nd message request

Y710, ..., Y897: 1st, ..., 152nd message request
152 different user message can be displayed on the screen containing user messages with the help of flags Y710, ..., Y897. Of the maximum 152 messages only one, displayed in the 2nd line of screen, is active. (For reading the active message there is no need to switch over to the screen containing the user messages.)
Due to this only one of flags I710, ..., I897 has TRUE state. It is the task of the PLC programmer to define the method of canceling the user messages. To cancel an error message also the RESET button, the state of which is sent through input flag I477 can be used. A message flag can be canceled (DY7nn) before it becomes active in case the reason of the message has ceased. Naturally in this case it also is deleted from the screen listing the messages.
The message string must be entered into module :199. The strings are separated by commas ",". The end of module together with the last message is indicated by character $:

:198MESSAGE1,MESSAGE2,...,MESSAGE152$

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y900	1st axis interpolator START
Y901	1st axis interpolator strobe signal
Y902	1st axis movement with feed
Y903	1st axis incremental movement
Y904	1st axis go to reference point
Y905	1st axis interpolator RESET
Y906	
Y907	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y910	2nd axis interpolator START
Y911	2nd axis interpolator strobe signal
Y912	2nd axis movement with feed
Y913	2nd axis incremental movement
Y914	2nd axis go to reference point
Y915	2nd axis interpolator RESET
Y916	
Y917	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y920	3rd axis interpolator START
Y921	3rd axis interpolator strobe signal
Y922	3rd axis movement with feed
Y923	3rd axis incremental movement
Y924	3rd axis go to reference point
Y925	3rd axis interpolator RESET
Y926	
Y927	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y930	4th axis interpolator START
Y931	4th axis interpolator strobe signal
Y932	4th axis movement with feed
Y933	4th axis incremental movement
Y934	4th axis go to reference point
Y935	4th axis interpolator RESET
Y936	
Y937	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y940	5th axis interpolator START
Y941	5th axis interpolator strobe signal
Y942	5th axis movement with feed
Y943	5th axis incremental movement
Y944	5th axis go to reference point
Y945	5th axis interpolator RESET
Y946	
Y947	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y950	6th axis interpolator START
Y951	6th axis interpolator strobe signal
Y952	6th axis movement with feed
Y953	6th axis incremental movement
Y954	6th axis go to reference point
Y955	6th axis interpolator RESET
Y956	
Y957	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y960	7th axis interpolator START
Y961	7th axis interpolator strobe signal
Y962	7th axis movement with feed
Y963	7th axis incremental movement
Y964	7th axis go to reference point
Y965	7th axis interpolator RESET
Y966	
Y967	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y970	8th axis interpolator START
Y971	8th axis interpolator strobe signal
Y972	8th axis movement with feed
Y973	8th axis incremental movement
Y974	8th axis go to reference point
Y975	8th axis interpolator RESET
Y976	
Y977	

☞ *The below flags are effective only in case of PLC controlled axes selected at flags Y630, ..., Y637.*

Y900, Y910, ..., Y970: 1st, 2nd, ..., 8th axis interpolator START
If the flag is set to 1 movement starts on the appropriate axis, provided the interpolator has valid movement command.
If the flag is set to 0 the movement stops (STOP). The interpolator stop flag (I900, I910, ..., I970) is set to 1 by the interpolator only after it has stopped with deceleration defined at parameter 470n ACCn. All movements cease on the axis when the appropriate 1st, ..., 8th axis in position flag I560, ..., I567 is set to 1.

Y901, Y911, ..., Y971: 1st, 2nd, ..., 8th axis interpolator strobe signal
The following flags and registers fully define movement commands for the interpolator:

Y902, Y912, ..., Y972: 1st, 2nd, ..., 8th axis movement with feed
Y903, Y913, ..., Y973: 1st, 2nd, ..., 8th axis incremental movement
RH150, RH151, ...: 1st, ... axis end position value
RH152, ...: 1st, ... axis feed rate value

After the necessary values have been entered into the above flags and registers on the axis to be moved the interpolator must be told to receive the movement parameters by setting the appropriate flag Y901, Y911, ..., Y971 to 1. The interpolator acknowledges the receipt of movement parameters by setting the appropriate flag I901, I911, ..., I971 to 0.

The movement can only be started in case the appropriate 1^{st}, 2^{nd}, ..., 8^{th} axis interpolator START flag Y900, Y910, ..., Y970 is set to 1.

Y902, Y912, ..., Y972: 1^{st}, 2^{nd}, ..., 8^{th} axis movement with feed
If the flag
=0 the interpolator moves on the appropriate axis at rapid traverse rate specified at parameter 468n RAPIDn.
=1 the interpolator moves on the appropriate axis at the value entered into the appropriate axis speed command register RH152, ...: 1^{st}, ... The interpolator restricts the feed rate value entered by the value defined at parameter 474n FEEDMAXn.

Y903, Y913, ..., Y973: 1^{st}, 2^{nd}, ..., 8^{th} axis incremental movement
If the flag
=0 the interpolator interprets the data entered into axis end position command register RH150, RH151, ...: 1^{st}, ... as absolute movement.
=1 the interpolator interprets the data entered into axis end position command register RH150, RH151, ...: 1^{st}, ... as incremental movement.

Y904, Y914, ..., Y974: 1^{st}, 2^{nd}, ..., 8^{th} axis go to reference point
If reference point return is to be executed on an axis, flag Y904, Y914, ..., Y974 belonging to the appropriate axis must be set to 1. The executed reference point return can be read at the appropriate flag I903, I913, ..., I973.
The reference point return is started with setting the appropriate START flag Y900, Y910, ..., Y970 to 1. The reference point return can be stopped and restarted by switching the START flag off and on.

Y905, Y915, ..., Y975: 1^{st}, 2^{nd}, ..., 8^{th} axis interpolator RESET
This flag must be set to 1 if an already started movement is to be stopped and the movement command to be canceled on one of the PLC controlled axes.

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y980	
Y981	
Y982	
Y983	
Y984	
Y985	
Y986	
Y987	

Flag Identity	Meaning of Flag if Value=1 (TRUE)
Y990	
Y991	
Y992	
Y993	
Y994	
Y995	
Y996	
Y997	

2.2.3 Registers from NC to PLC (Input Registers)

Reference to input registers can be done with string RH and three digits:

RHpqr

The value of the first digit:

p=0,1

The value range of the second digit (q) for input registers:

q=0,1,2,3,4

The third one is decimal, its range:

r=0,1,2,3,4,5,6,7,8,9

Input registers are 16-bit variables. The variables are always transferred in binary form, thus the value in register must be regarded as a binary number.
In the followings a detailed list of input registers is shown:

RH000	1st M function code (belonging to flag I520)

RH001	2nd M function code (belonging to flag I521)

RH002	3rd M function code (belonging to flag I522)

RH003	4th M function code (belonging to flag I523)

RH004	5th M function code (belonging to flag I524)

In a program block up to 5 M functions, which are to be transferred to the PLC can be used. According to the order written in the block the NC writes the first loaded code into register RH000 and sets flag I520 to 1, it writes the second M function into register RH001 and sets flag I521 to 1 and so on. The code is transferred in binary form.
The PLC programmer determines the order of the execution of the different M functions within the given block.

RH005	S function code (belonging to flag I525)

If S function is written in a program block the NC sets flag I525 to 1 and data S appears in input register RH005. The data is transferred in binary form.

RH006	T function code (belonging to flag I526)

If T function is written in a program block the NC sets flag I526 to 1 and the T code appears in input register RH006. The code is transferred in binary form.

RH007	"A" function code (belonging to flag I527)

If address A is selected for function (parameter state: 0183 **A.MISCEL**=1), and A function is written in a program block the NC sets flag I527 to 1 and the A code appears in input register RH007. The code is transferred in binary form.

RH008	"B" function code (belonging to flag I530)

If address B is selected for function (parameter state: 0186 **B.MISCEL**=1), and B function is written in a program block the NC sets flag I530 to 1 and the B code appears in input register RH008. The code is transferred in binary form.

RH009	"C" function code (belonging to flag I531)

If address C is selected for function (parameter state: 0189 **C.MISCEL**=1), and C function is written in a program block the NC sets flag I531 to 1 and the C code appears in input register RH009. The code is transferred in binary form.

RH010	1^{st} spindle current revolution

If the 1^{st} spindle is mounted with encoder and value of parameter 5023 **ENCODERS1** contains the resolution of the encoder the current revolution of spindle is measured by the control in cycles, and informs on its value at register RH010. The revolution value is transferred in rpm in binary form.
If the value of parameter 5023 **ENCODERS1** is 0 the control interprets it as no encoder is mounted on the spindle and writes the calculated revolution involving override and range limits. The value of this register occurs in the current S display.

RH011	1^{st} spindle modified programmed revolution

The PLC writes the programmed S code in programmed revolution register RH060. The NC calculates the command signal for the transferred spindle drive by modifying the contents of this register with the spindle override value, examines, whether the in such way calculated value is greater or less than the value clamped by parameter belonging to the current range. If yes, it executes the clampings and writes the in such way calculated value into register RH011. It writes the continuously altering value in the switched-on state of constant cutting rate calculation (G96) into register RH011. If the spindle is mounted with encoder the spindle can be supervised by the continuous comparing of RH011 and current revolution register RH010 in PLC.

RH012	G96 revolution on the active spindle

It is the value of the active spindle revolution in the switched-on state of constant surface speed (G96) involving position and the programmed maximum revolution (G92 S) calculated by the control. This value needs to be copied by the PLC program into the output register RH060 or RH065 for the spindle revolution calculated for programmed constant surface speed to be effective.

RH013	Programmed maximum revolution on the active spindle

It is the value of maximum spindle revolution defined by command G92 S. The NC takes the limit of RH013 into account by the value written in register RH012 in state G96, and only in state G96.

RH014	

RH015	2nd spindle current revolution

If the 2nd spindle is mounted with encoder and value of parameter 5024 **ENCODERS2** contains the resolution of the encoder the current revolution of spindle is measured by the control in cycles, and informs on its value at register RH015. The revolution value is transferred in rpm in binary form.
If the value of parameter 5024 **ENCODERS2** is 0 the control interprets it as no encoder is mounted on the spindle and writes the calculated revolution involving override and range limits. The value of this register occurs in the current S display.

RH016	2nd spindle modified programmed revolution

The PLC writes the programmed S code in programmed revolution register RH065. The NC calculates the command signal for the transferred spindle drive by modifying the contents of this register with the spindle override value, examines, whether the in such way calculated value is greater or less than the value clamped by parameter belonging to the current range. If yes, it executes the clampings and writes the in such way calculated value into register RH016. It writes the continuously altering value in the switched-on state of constant cutting rate calculation (G96) into register RH016. If the spindle is mounted with encoder the spindle can be supervised by the continuous comparing of RH016 and current revolution register RH015 in PLC.

RH017	

RH018	

RH019	

RH020	Active message code

If in the message field, i.e. in the 2^{nd} line of screen a message is displayed, no matter whether it comes from the NC or the PLC the message code can be read at register RH020. Error coding is contained by chapter 6.4 Listing of Global Messages on page 234. If flag I537 is set to 1 this code is valid, if it is 0 the code is invalid.

RH021	Year

The register contains the current year in 4 tetrades, in BCD form. E.g.: If the current year is 2013 the value of the register is: .2013

RH022	Month, Day

The register contains the current Month on the upper two tetrades while the current Day on the lower two ones, in BCD form. E.g.: If it is 27, October the value of the register is: .1027.

RH023	Hour, Minute

The register contains the current Hour on the upper two tetrades while the current Minute on the lower two ones, in BCD form. E.g.: If it is 32 past 4 p.m. the value of the register is: .1632.

RH024	Second

The register contains the current Second on the lower two tetrades, in BCD form. E.g.: .0018.

RH025	

RH026	Meanings of softkeys

In register RH026 the meanings of the softkeys belonging to the current screen (register RH027) can be found. If the upper byte of the register is 0, the softkeys contain the screen menu, if the value of the upper byte is 1 the action menu is seen on softkeys:

RH026=00xxh: screen menu

RH026=01xxh: action menu

Independent of the upper byte (screen menu or action menu) state the lower byte of register always shows the code of the previously selected action menu belonging to the screen. For detailed description see chapter 6.6 Codes of Screens and Softkeys on page 240.

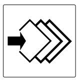
RH027	Screen code

Register RH027 contains the code of the displayed screen. Its lower byte is the number of screen group containing the current screen (e.g. POSITION), while its upper byte is the number of screen within the screen group (e.g. ABSOLUTE). For detailed description see chapter 6.6 Codes of Screen Menu and Action Menu Captions on page 240.

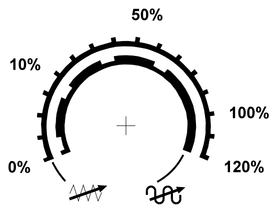
RH028	F% (feedrate override) input register

RH028	%
0	0
1	1
2	2
3	5
4	10
5	20
6	30
7	40
8	50
9	60
10	70
11	80
12	90
13	100
14	110
15	120

If Y527=1 (feed override from NC keyboard), Y531=1 (machine control board 1), or Y532=1 (machine control board 2) the state of feed override switch is sent by the NC to the PLC through register RH028. The contents of the register is binary. Below the percent equivalent of each value can be seen (the control works with the % value in the line of code). In the above cases the PLC programmer must take care of copying the value of input register RH028 to output register RH078.

If Y527=1 (switch F% operate from SW control panel) the feed rate can be modified by means of selecting one of screens OPERATOR’S PANE L, POSITION or CHECK.
Afterwards select action menu % F4 after pressing action menu key  . In this case captions **G–, G+, S–, S+, F–, F+** appear on softkeys. By pressing key F– the feed rate override value (i.e. value of register RH028) decreases, while with the help of key F+ value of register RH028 increases.

If Y532=1 a rotary switch is mounted on machine control board 2 for feedrate override state of which can be read from register RH028.



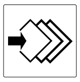
☞ ***Warning!***
Only one of flags Y527 and Y532 can be 1, i.e. feed rate override may be selected by the use of either SW control panel or machine control board switch!

RH029	S% (spindle speed override) input register

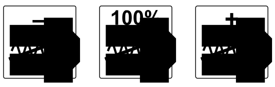
If Y526=1 (spindle override from NC keyboard), Y531=1 (machine control board 1), or Y532=1 (machine control board 2) the state of spindle override switch is sent by the NC to the PLC through register RH029. The contents of the register is binary. Below the percent equivalent of each value can be seen (the control works with the % value in the line of code).

RH029	%
0	50
1	60
2	70
3	80
4	90
5	100
6	110
7	120
8	130
9	140
10	150

In the above cases the PLC programmer must take care of copying the value of input register RH029 to output register RH079.

If Y526=1 (switch S% operate from SW control panel) the spindle override value can be modified by means of selecting one of screens OPERATOR'S PANE L, POSITION or CHECK. Afterwards select action menu % F^4 after pressing action menu key  . In this case captions **G–, G+, S–, S+, F–, F+** appear on softkeys. By pressing key S– the spindle override value (i.e. value of register RH029) decreases, while with the help of key S+ value of register RH028 increases.

If Y532=1 three push-buttons are mounted on machine control board 2 in order to set spindle %, with which the override value, i.e. that of register RH029 can be decreased or increased, as well as by the use of which 100% can be set.



☞ ***Warning!***
Only one of flags Y526 and Y532 can be 1, i.e. spindle override may be selected by the use of either SW control panel or machine control board switch!

RH030	Number of program under execution

The number of program under current execution. This may be the number of main program, subprogram or macro.

RH031	Number of program selected for automatic execution

This is always the number of the main program selected for automatic execution.

RH032	Number of program selected for execution in manual data input mode

This is always the number of the main program selected for execution in manual data input mode.

RH033	

RH034	

RH035	1st analog input on 1st INT board

RH036	2nd analog input on 1st INT board

RH037	3rd analog input on 1st INT board

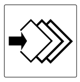
RH038	4th analog input on 1st INT board

1st INT (interface) board can optionally be equipped with AD (analog to digital) converter unit capable of receiving 4 different analog signals. Values of analog signals can be read through the above registers. Resolution of AD convert is 12 bits. It is calibrated according to the below table:

Input value in V	data read from register RH
+10V	.0000
0V	.0800
-9.995V	.0FFF

RH039	R% (rapid traverse override) input register

If Y525=1 (rapid traverse override from SW control panel) the control sends the rapid traverse override switch state to PLC in register RH039. If Y525=1 (switch R% operate from SW control panel) the rapid traverse override can be modified by means of selecting one of screens OPERATOR’S PANE L, POSITION or CHECK.

Afterwards select action menu % F^4 after pressing action menu key  . In this case captions **G–, G+, S–, S+, F–, F+** appear on softkeys. By pressing key G– the rapid traverse override value (i.e. value of register RH039) decreases, while with the help of key G+ value of register RH039 increases.

The register contents are in binary form. The percent correspondent of each value (acknowledged by the control for the given value) can be seen in the below two tables. If **RAPOVER** No. 1204=0 it is the first table, while if **RAPOVER** No. 1204>0 it is the second one

1204 **RAPOVER**=0	
RH039	%
0	0
1	1
2	2
3	5
4	10
5	20
6	30
7	40
8	50
9	60
10	70
11	80
12	90
13	100

1204 **RAPOVER**>0	
RH039	%
0	F0=RAPOVER
1	25
2	50
3	100

In the above cases the PLC programmer must take care of copying the value of input register RH039 to output register RH089.

RH040	G51.2 polygonal turning data P

RH041	G51.2 polygonal turning data Q

Polygonal turning can be programmed by specifying block G51.2 P_ Q_. The ratio of P/Q defines the ratio of revolution of the main spindle (workpiece) and the slave spindle (tool). Programmed absolute value of P is available in register RH040 while value Q in register in RH041. The revolution of the tool spindle is calculated according the formula below:

$$S_{toolspindle} = \frac{Q}{P}S = \frac{RH041}{RH040}S$$

The PLC program should turn the tool spindle to the revolution calculated before, then it should request synchronization via flags Y655 or Y665.

Command G50.2 turns polygonal turning off and flag I640goes to low. The PLC program should cancel the synchronization of the two spindles, then turn the tool spindle off.

RH042	Actual feed lower word

RH043	Actual feed higher word

Feed in mm/min or in inch/min can be calculated from the data in registers RH042, RH043 according the table below

	4764 INCRSYSTA=1	4765 INCRSYSTB=1	4766 INCRSYSTC
47	F[mm/min]=data/10^3	F[mm/min]=data/10^4	F[mm/min]=data/10^5
4763 INCHDET=1	F[inch/min]=data/10^4	F[inch/min]=data/10^5	F[inch/min]=data/10^6

RH044	

RH045	

RH046	

RH047	

RH048	

RH049	Code of valid push-button

If a key is pressed on data input keyboard the NC sets flag I536 to 1 for 1 PLC cycle and places the key code into register RH049. Key codes can be found in chapter 6.5 Listing of Push-button codes on page 237. If flag I536 is 1 the code herein is valid, however if it is 0 the code is invalid.

RH100	1st axis current position lower word

RH101	1st axis current position upper word

At the two above registers the position of the 1st axis registered in machine coordinate system can be read in output increment.

RH102	1st axis lag lower word

RH103	1st axis lag upper word

At the above two registers the lag value of the servo loop of the 1st axis can be read in output increment.

RH104	1st axis drive current

When applying NCT digital servo drive and XMU CAN digital measuring system board it contains the quotient of the actual and nominal current of the 1st axis (I/I_n) per mill (‰) with sign, in two's complement.

RH105	2^{nd} axis current position lower word

RH106	2^{nd} axis current position upper word

At the two above registers the position of the 2^{nd} axis registered in machine coordinate system can be read in output increment.

RH107	2^{nd} axis lag lower word

RH108	2^{nd} axis lag upper word

At the above two registers the lag value of the servo loop of the 2^{nd} axis can be read in output increment.

RH109	2^{nd} axis drive current

When applying NCT digital servo drive and XMU CAN digital measuring system board it contains the quotient of the actual and nominal current of the 2^{nd} axis (I/I_n) per mill (‰) with sign, in two's complement.

RH110	3^{rd} axis current position lower word

RH111	3^{rd} axis current position upper word

At the two above registers the position of the 3^{rd} axis registered in machine coordinate system can be read in output increment.

RH112	3^{rd} axis lag lower word

RH113	3^{rd} axis lag upper word

At the above two registers the lag value of the servo loop of the 3^{rd} axis can be read in output increment.

RH114	3rd axis drive current

When applying NCT digital servo drive and XMU CAN digital measuring system board it contains the quotient of the actual and nominal current of the 3rd axis (I/I_n) per mill (‰) with sign, in two's complement.

RH115	4th axis current position lower word

RH116	4th axis current position upper word

At the two above registers the position of the 4th axis registered in machine coordinate system can be read in output increment.

RH117	4th axis lag lower word

RH118	4th axis lag upper word

At the above two registers the lag value of the servo loop of the 4th axis can be read in output increment.

RH119	4th axis drive current

When applying NCT digital servo drive and XMU CAN digital measuring system board it contains the quotient of the actual and nominal current of the 4th axis (I/I_n) per mill (‰) with sign, in two's complement.

RH120	5th axis current position lower word

RH121	5th axis current position upper word

At the two above registers the position of the 5th axis registered in machine coordinate system can be read in output increment.

RH122	5th axis lag lower word

RH123	5th axis lag upper word

At the above two registers the lag value of the servo loop of the 5th axis can be read in output increment.

RH124	5th axis drive current

When applying NCT digital servo drive and XMU CAN digital measuring system board it contains the quotient of the actual and nominal current of the 5th axis (I/I_n) per mill (‰) with sign, in two's complement.

RH125	6th axis current position lower word

RH126	6th axis current position upper word

At the two above registers the position of the 6th axis registered in machine coordinate system can be read in output increment.

RH127	6th axis lag lower word

RH128	6th axis lag upper word

At the above two registers the lag value of the servo loop of the 6th axis can be read in output increment.

RH129	6th axis drive current

When applying NCT digital servo drive and XMU CAN digital measuring system board it contains the quotient of the actual and nominal current of the 6th axis (I/I_n) per mill (‰) with sign, in two's complement.

RH130	7th axis current position lower word

RH131	7th axis current position upper word

At the two above registers the position of the 7th axis registered in machine coordinate system can be read in output increment.

RH132	7th axis lag lower word

RH133	7th axis lag upper word

At the above two registers the lag value of the servo loop of the 7th axis can be read in output increment.

RH134	7th axis drive current

When applying NCT digital servo drive and XMU CAN digital measuring system board it contains the quotient of the actual and nominal current of the 7th axis (I/I_n) per mill (‰) with sign, in two's complement.

RH135	8th axis current position lower word

RH136	8th axis current position upper word

At the two above registers the position of the 8th axis registered in machine coordinate system can be read in output increment.

RH137	8th axis lag lower word

RH138	8th axis lag upper word

At the above two registers the lag value of the servo loop of the 8th axis can be read in output increment.

RH139	8th axis drive current

When applying NCT digital servo drive and XMU CAN digital measuring system board it contains the quotient of the actual and nominal current of the 8th axis (I/I_n) per mill (‰) with sign, in two's complement.

RH140	

RH141	

RH142	

RH143	

RH144	1st spindle drive current

When applying NCT digital main drive and XMU CAN digital measuring system board it contains the quotient of the actual and nominal current of the 1st spindle (I/I_n) per mill (‰) with sign, in two's complement.

RH145	

RH146	

RH147	

RH148	

RH149	2^{nd} spindle drive current

When applying NCT digital main drive and XMU CAN digital measuring system board it contains the quotient of the actual and nominal current of the 2^{nd} spindle (I/I_n) per mill (‰) with sign, in two's complement.

2.2.4 Registers from PLC to NC (Output Registers)

Reference to output registers can be done with character RH and three digits:

RHpqr

The value of the first digit:

p=0,1

The value range of the second digit (q) for output registers:

q=5,6,7,8,9

The third one is decimal, its range:

r=0,1,2,3,4,5,6,7,8,9

Input registers are 16-bit variables. The variables always have to be transferred to the NC in binary form.

In the followings a detailed list of output registers is shown:

RH050	Number of program to be executed

If the PLC selects a program in memory, its number is specified in this register. Afterwards flag Y600 or Y601 is set to 1 in function of the program execution being in automatic or manual data input mode.

RH051	Start address of data to be transmitted

RH052	Number of bytes to be transmitted

RH053	Code of transmitter periphery

If the PLC needs to transmit array through a periphery (e.g. through serial channel RS-232), it writes the data to be transmitted at inner variables F010, ..., F499. The array start address is specified in register RH051, the number of bytes to be transmitted, i.e. the record length is given in register RH052.

If e.g.area F400, ..., F463 is selected for data transmission the registers are filled up as follows:

```
,400
SRH051
,64
SRH052
```

The code of the periphery, through which the data is to be transmitted must be given in register RH053. If

RH053=1: data is transmitted through 1st serial channel

RH053=2: data is transmitted through 2nd serial channel.

RH054	Start address of received data

RH055	Number of received bytes

RH056	Code of receiver periphery

If the PLC needs to receive array from external device through a periphery (e.g. through serial channel RS-232), the incoming data are required at inner variables F010, ..., F499. The array start address is specified in register RH054, the number of bytes to be received, i.e. the record length is given in register RH055.
If e.g.area F300, ..., F363 is selected for data receive the registers are filled up as follows:

```
,300
SRH054
,64
SRH055
```

The code of the periphery, through which the data is to be received must be given in register RH056. If

RH053=1: data is received through 1st serial channel
RH053=2: data is received through 2nd serial channel.

RH057	"A" function current value

RH058	"B" function current value

RH059	"C" function current value

If address A, B, or C is selected for function (parameter state: 0183 **A.MISCEL**=1, 0186 **B.MISCEL**=1, or 0189 **C.MISCEL**=1), the current value A, B, C can be displayed at these registers on the appropriate screen.
The value copied from register RH007, RH008, or RH009 is written into register RH057, RH058, or RH059 after the appropriate command is executed. The number must be entered into the register in binary form.

RH060	1st spindle programmed S register

Command signal transfer to the 1st spindle is done through register RH060 after address S has been programmed.
First the command signal transfer has to be enabled by statement U652. The number entered into register RH060 (its value range: 0-65535) is regarded as an unsigned number by the NC. The polarity must be defined by setting flag Y653 (U653: positive, D653: negative). Flag Y654 must be set to 0 in order to transfer the command signal from register RH060.

Command signal output on the basis of code S (Y654=0)

If flag Y654 is set to 0 the NC transfers the value written into register RH060 into the D/A converter as command signal. The transfer is not done directly, but

- the number written into register is interpreted as spindle revolution (code S) and the command signal amount is calibrated according to the valid range code (register RH063) and parameter group SPINDLE,
- the spindle override value is taken into account,
- the command signal cannot be under or over the minimum or maximum value of range revolution specified at parameter group SPINDLE,
- the command signal is not transferred promptly, but reaches its size specified at parameter group SPINDLE through linear ramping,
- in the state of constant surface speed calculation (G96) the command signal is altered automatically in the function of the selected coordinate.

The value of revolution input register RH005 (data programmed at address S) must be copied into register RH060.

The initialization of register RH060 is the task of the PLC programmer.
Before inverting flag Y654 the PLC programmer must take care of the spindle being stopped.

RH061	1st spindle binary command register

Binary command signal output (spindle JOG)

If flag Y654 is set to 1 the value written into register RH061 is output to the D/A converter in direct binary form and transferred to the spindle drive by the NC as command signal. It can be used in case of gear range change for the fluctuation of spindle, as well as in spindle jog state for jogging the spindle.
After setting flag Y651 to 1 this register is used for setting the rate of zero pulse search in case of spindle orientation.
Interpretation of the numbers written into the register and their effect on the analog output:

- the value in case of +10 V is F000h,
- the value in case of +5 V is F7FFh,
- the value in case of +2.5 V is FBFFh,
- the value in case of 0 V is 0000h,
- the value in case of −2.5 V is 0400h,
- the value in case of −5 V is 0800h,
- the value in case of −10 V is1000h

RH062	1st spindle rotation code (M3, M4, M5, M19)

The revolution state of 1st spindle must be told the NC through register RH062. The change of revolution state can be initiated

- by command M3, M4, M5, or M19 written in the part program,
- from PLC, for example orientation before tool replacement (M19),
- or with the help of push-buttons M3, M4, M5 by the operator.

In all cases the appropriate rotation code 3, 4, 5, or 19 must be entered in binary form into register RH062. The initialization of the register is the task of the PLC programmer. The current rotation state is displayed as the value of this register.

RH063	1st spindle range code (M11, ..., M18)

The state of 1st spindle range must be told the NC through register RH063. Change of the state can be initiated

- by command M11, ..., M18 written in the part program,
- or from the PLC..

If there is no overlapping between the revolution ranges of spindle, i.e. if the maximum revolution the ith range is n, and the minimum revolution of the (i+1)th range is n+1, then the gear range change can be automatically generated on the basis of the programmed code S and there is no need to program M11, ..., M18.
In all cases the appropriate range code 11, ..., 18 must be entered in binary form in register RH063. The initialization of the register is the task of the PLC programmer. The current state is displayed by the NC through the register, as well as it takes the parameters used for calibrating spindle command signal transfer into account on the basis of the spindle range register.

RH064	Active tool code (T)

The number of the active tool must be entered in binary form in this register. The initialization of the register is the task of the PLC programmer. The current tool number is displayed by the NC through this register.

RH065	2nd spindle programmed S register

Command signal transfer to the 2nd spindle is done through register RH065 after address S has been programmed.
First the command signal transfer has to be enabled by statement U662. The number entered into register RH065 (its value range: 0-65535) is regarded as an unsigned number by the NC. The polarity must be defined by setting flag Y663 (U663: positive, D663: negative). Flag Y664 must be set to 0 in order to transfer the command signal from register RH065.

Command signal output on the basis of code S (Y664=0)

If flag Y664 is set to 0 the NC transfers the value written into register RH065 into the D/A converter as command signal. The transfer is not done directly, but

- the number written into register is interpreted as spindle revolution (code S) and the command signal amount is calibrated according to the valid range code (register RH068) and parameter group SPINDLE,
- the spindle override value is taken into account,
- the command signal cannot be under or over the minimum or maximum value of range revolution specified at parameter group SPINDLE,
- the command signal is not transferred promptly, but reaches its size specified at parameter group SPINDLE through linear ramping,

- in the state of constant surface speed calculation (G96) the command signal is altered automatically in the function of the selected coordinate.

The value of revolution input register RH005 (data programmed at address S) must be copied into register RH065.

The initialization of register RH065 is the task of the PLC programmer.
Before inverting flag Y664 the PLC programmer must take care of the spindle being stopped.

RH066	2^{nd} spindle binary command register

Binary command signal output (spindle JOG)

If flag Y664 is set to 1 the value written into register RH066 is output to the D/A converter in direct binary form and transferred to the spindle drive by the NC as command signal. It can be used in case of gear range change for the fluctuation of spindle, as well as in spindle jog state for jogging the spindle.
After setting flag Y661 to 1 this register is used for setting the rate of zero pulse search in case of spindle orientation.
Interpretation of the numbers written into the register and their effect on the analog output:

- the value in case of +10 V is F000h,
- the value in case of +5 V is F7FFh,
- the value in case of +2.5 V is FBFFh,
- the value in case of 0 V is 0000h,
- the value in case of −2.5 V is 0400h,
- the value in case of −5 V is 0800h,
- the value in case of −10 V is1000h

RH067	2^{nd} spindle rotation code (M3, M4, M5, M19)

The revolution state of 2^{nd} spindle must be told the NC through register RH067. The change of revolution state can be initiated

- by command M3, M4, M5, or M19 written in the part program,
- from PLC, for example orientation before tool replacement (M19),
- or with the help of push-buttons M3, M4, M5 by the operator.

In all cases the appropriate rotation code 3, 4, 5, or 19 must be entered in binary form into register RH067. The initialization of the register is the task of the PLC programmer. The current rotation state is displayed as the value of this register.

RH068	2^{nd} spindle range code (M11, ..., M18)

The state of 2^{nd} spindle range must be told the NC through register RH068. Change of the state can be initiated

- by command M11, ..., M18 written in the part program,
- or from the PLC..

If there is no overlapping between the revolution ranges of spindle, i.e. if the maximum revolution the i^{th} range is n, and the minimum revolution of the $(i+1)^{th}$ range is n+1, then the gear range change can be automatically generated on the basis of the programmed code S and there is no need to program M11, ..., M18.
In all cases the appropriate range code 11, ..., 18 must be entered in binary form in register RH068. The initialization of the register is the task of the PLC programmer. The current state is displayed by the NC through the register, as well as it takes the parameters used for calibrating spindle command signal transfer into account on the basis of the spindle range register.

RH069	

RH070	1st M group display

RH071	2nd M group display

RH072	3rd M group display

RH073	4th M group display

RH074	5th M group display

RH075	6th M group display

RH076	7th M group display

RH077	8th M group display

It is possible to display 8 different M groups on the FUNCTION screen of the control. The 8 different M functions are displayed in one line according to the numbering of the registers. If the contents of register RH070, ..., RH077 is 0 in the appropriate place of its group spaces are shown on the screen. If a number other than 0 is entered into the register the contents of the appropriate register is displayed beside character M of the appropriate column. The value range of the number displayed is 0-99. The number must be entered into the register in binary form.

RH078	F% (feed override) output register

The current feed rate override value must be entered into register RH078 in the following way:

RH078	%
0	0
1	1
2	2
3	5
4	10
5	20
6	30
7	40
8	50
9	60
10	70
11	80
12	90
13	100
14	110
15	120

Feed override value is validated by the NC on the basis of register RH078. Register value 0 (0%) refers to not only the feed rate but also to the rapid traverse override. The override value written in register RH078 is also effective for PLC axes.

If Y527=1 (feed override from SW control panel) or Y532=1 (from machine control board 2) the override can be read from register RH028, otherwise the PLC programmer must set it up e.g. decode it from switch and enter it into register RH078 in the enclosed format.

RH079	S% spindle speed override output register

The current spindle speed override value must be entered into register RH079 in the following way:

RH079	%
0	50
1	60
2	70
3	80
4	90
5	100
6	110
7	120
8	130
9	140
10	150

Spindle override value is validated by the NC on the basis of register RH079.

If Y526=1 (spindle override from SW control panel) or Y532=1 (from machine control board 2) the override can be read from register RH029, otherwise the PLC programmer must set it up e.g. decode it from switch and enter it into register RH079 in the enclosed format.

RH080	1st analog output scaled command signal

It is possible to create two analog output signal in the control. If the n^th physical axis is ready to work but not selected for axis handle, i.e. the value of parameter AXISTn No.444n is 0, then the appropriate analog output can be applied for signal transfer. The physical axis on which the 1st and 2nd analog output signal is transferred is specified at register COMMAND1 No. 0101 and COMMAND2 No. 0102 of parameter field by entering a number between 1 and 8 in the appropriate register. Scaling of the output (the value in case of 10V, minimum and maximum value transferred) can be done at parameter group **0121 ANALOG1** and **0141 ANALOG2** similarly to spindle output.
The 1st analog output scaled command signal transfer is done through register RH080. The number entered into register RH080 (its value range: 0-65535) is handled as an unsigned number by the NC. The command signal polarity must be specified by setting flag Y670 (U670: positive, D670: negative). If flag Y671 is set to 0 the command signal is transferred from this register.

Command signal transfer regarding scaling (Y671=0)

If flag Y671 is set to 0 the value entered into register RH080 is not transferred directly as command signal, but

- scales the value of register on the basis of the parameter,
- it takes the override value in register RH082 into account,
- the command signal cannot be under or over the minimum or maximum value specified at the given parameter,

- the command signal is not transferred promptly, but reaches its sixe specified at parameter through linear rising and falling edge.

RH081	1st analog output binary command signal

Binary command signal output (Y671=1)

If flag Y671 is set to 1 the value entered into register RH081 is transferred directly, in binary form into the D/A converter as command signal by the NC.

- the value in case of 10 V is FFFFh,
- the value in case of 0 V is 0000h,
- and at flagY670 the sign can be specified.

RH082	1st analog output % (override) value

The override value of the 1st analog output can be entered into register RH082. The override value must be given in %. If for example the contents of register RH082 is 100, in the 1st analog output the command signal of register RH080 is transferred.

RH083	

RH084	

RH085	2nd analog output scaled command signal

It is possible to create two analog output in the control. If the nth physical axis is ready to work but not selected for axis handle, i.e. the value of parameter 444n **AXISTn** is 0, then the appropriate analog output can be applied for signal transfer. The physical axis on which the 1st and 2nd analog output is transferred is specified at register 0101 **COMMAND1** and 0102 **COMMAND2** of parameter field by entering a number between 1 and 8 in the appropriate register. Scaling of the output (the value in case of 10V, minimum and maximum value transferred) can be done at parameter group **0121 ANALOG1** and **0141ANALOG2** similarly to spindle output.

The 2nd analog output scaled command signal transfer is done through register RH085. The number entered into register RH085 (its value range: 0-65535) is handled as an unsigned number by the NC. The command signal polarity must be specified by setting flag Y672 (U672: positive, D672: negative). If flag Y673 is set to 0 the command signal is transferred from this register.

Command signal transfer regarding scaling (Y673=0)

If flag Y673 is set to 0 the value entered into register RH085 is not transferred directly as command signal, but

- scales the number entered into register on the basis of the parameter,
- it takes the override value in register RH087 into account,
- the command signal cannot be under or over the minimum or maximum value specified at the given parameter,
- the command signal is not transferred promptly, but reaches its size specified at parameter through linear rising and falling edge.

RH086	2nd analog output binary command signal

Binary command signal output (Y673=1)

If flag Y673 is set to 1 the value entered into register RH086 is transferred directly, in binary form into the D/A converter as command signal by the NC.

- the value in case of 10 V is FFFFh,
- the value in case of 0 V is 0000h,
- and at flagY670 the sign can be specified.

RH087	2nd analog output % (override) value

The override value of the 2nd analog output signal can be entered into register RH087. The override value must be given in %. If for example the contents of register RH087 is 100, in the 2nd analog output the command signal referring to register RH085 is transferred.

RH088	Chopping Override Register

In register RH088 can be defined the override value of chopping that modifies the chopping rate defined in parameter 0282 CHOPRATE per cents (%). Unit of value is %. Range of data: 0% ... 200% in 1% steps.

RH089	R% (rapid traverse override) output register

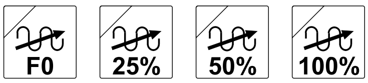
Rapid traverse override value is validated by the NC on the basis of register RH089. The register contents are binary. The percent correspondent of each value (acknowledged by the control for the given value) can be found in the below two tables. If 1204 **RAPOVER**=0 it is the first table, while if **RAPOVER** No. 1204>0 it is the second one

1204 **RAPOVER**=0	
RH089	%
0	0
1	1
2	2
3	5
4	10
5	20
6	30
7	40
8	50
9	60
10	70
11	80
12	90
13	100

1204 **RAPOVER**>0	
RH089	%
0	F0=RAPOVER
1	25
2	50
3	100

F0 is the value defined at parameter1204 **RAPOVER**. As seen in the enclosed table it has no 0% value, which is taken from feed override value.

If Y525=1 (rapid traverse override from SW control panel) the override can be read from register RH039, otherwise the PLC programmer must set it up e.g. decode it from switch and enter it into register RH089 in the enclosed format. If e.g. machine control board 2 is applied 4 free-purpose buttons can be mounted in the below form:



The override value can be selected by pressing the appropriate button.

The rapid traverse override value can also be decoded from feed override switch state.

RH090	Y700 message variable

RH091	Y701 message variable

RH092	Y702 message variable

RH093	Y703 message variable

RH094	Y704 message variable

RH095	Y705 message variable

RH096	Y706 message variable

RH097	Y707 message variable

RH090, ..., RH097: Y700, ..., Y707 message variable
If a message is to be displayed on screen indexed the appropriate value must be entered into the register of the appropriate message display. The value written into register must previously be converted into BCD format, if BCD number is to be displayed on screen. Otherwise the value found in register is displayed in hexadecimal form. It can be used for example for displaying the number of tool to be loaded in case of manual tool replacement.

RH098	

RH099	Key code from PLC

If the PLC needs to operate the NC through data input keyboard it sets flagY537 to 1. Afterwards it writes the appropriate key code into register RH099, than sets flag Y536 to 1 for 1 PLC cycle. Key codes can be found in chapter 6.5 Listing of Push-button Codes on page 237.

RH150	1st axis position command lower word

RH151	1st axis position command upper word

In case of PLC controlled axes the interpolator moves into the position entered here interpreted in incremental, or absolute value in function of the state of flag Y903. The dimensional unit of the position data is output increment.

RH152	1st axis feedrate command lower word

RH153	1st axis feedrate command upper word

In case of PLC controlled axes the axis moves at the rate entered here provided flag Y902 is set to 1. Interpretation of 1 unit of the rate parameter (RH152=1, RH153=0):

$$\frac{\textit{input increment}}{\text{min}}$$

☞ *Registers RH150, RH151, RH152, RH153 are effective only on the PLC controlled axes selected at flags Y630, ..., Y637.*

RH154	

RH155	2nd axis position command lower word

RH156	2nd axis position command upper word

In case of PLC controlled axes the interpolator moves into the position entered here interpreted in increment, or absolute value in function of the state of flag Y913. The interpretation of the position data is output increment.

RH157	2nd axis feedrate command lower word

RH158	2nd axis feedrate command upper word

In case of PLC controlled axes the axis moves at the rate entered here provided flag Y912 is set to 1. Interpretation of 1 unit of the rate parameter (RH157=1, RH158=0):

$$\frac{0.2\,input increment}{\min}$$

☞ *Registers RH155, RH156, RH157, RH158 are effective only on the PLC controlled axes selected at flags Y630, ..., Y637.*

RH159	

RH160	3rd axis position command lower word

RH161	3rd axis position command upper word

In case of PLC controlled axes the interpolator moves into the position entered here interpreted in increment, or absolute value in function of the state of flag Y923. The interpretation of the position data is output increment.

RH162	3rd axis feedrate command lower word

RH163	3rd axis feedrate command upper word

In case of PLC controlled axes the axis moves at the rate entered here provided flag Y922 is set to 1. Interpretation of 1 unit of the rate parameter (RH162=1, RH163=0):

$$\frac{0.2\,input increment}{\min}$$

☞ *Registers RH160, RH161, RH162, RH163 are effective only on the PLC controlled axes selected at flags Y630, ..., Y637.*

RH164	

RH165	4th axis position command lower word

RH166	4th axis position command upper word

In case of PLC controlled axes the interpolator moves into the position entered here interpreted in increment, or absolute value in function of the state of flag Y933. The interpretation of the position data is output increment.

RH167	4th axis feedrate command lower word

RH168	4th axis feedrate command upper word

In case of PLC controlled axes the axis moves at the rate entered here provided flag Y932 is set to 1. Interpretation of 1 unit of the rate parameter (RH167=1, RH168=0):

$$\frac{0.2 inputincrement}{\min}$$

☞ *Registers RH165, RH166, RH167, RH168 are effective only on the PLC controlled axes selected at flags Y630, ..., Y637.*

RH169	

RH170	5th axis position command lower word

RH171	5th axis position command upper word

In case of PLC controlled axes the interpolator moves into the position entered here interpreted in increment, or absolute value in function of the state of flag Y943. The interpretation of the position data is output increment.

RH172	5^{th} axis feedrate command lower word

RH173	5^{th} axis feedrate command upper word

In case of PLC controlled axes the axis moves at the rate entered here provided flag Y942 is set to 1. Interpretation of 1 unit of the rate parameter (RH172=1, RH173=0):

$$\frac{0.2\,input increment}{\min}$$

☞ *Registers RH170, RH171, RH172, RH173 are effective only on the PLC controlled axes selected at flags Y630, ..., Y637.*

RH174	

RH175	6^{th} axis position command lower word

RH176	6^{th} axis position command upper word

In case of PLC controlled axes the interpolator moves into the position entered here interpreted in increment, or absolute value in function of the state of flag Y953. The interpretation of the position data is output increment.

RH177	6^{th} axis feedrate command lower word

RH178	6^{th} axis feedrate command upper word

In case of PLC controlled axes the axis moves at the rate entered here provided flag Y952 is set to 1. Interpretation of 1 unit of the rate parameter (RH177=1, RH178=0):

$$\frac{0.2\,input increment}{\min}$$

☞ *Registers RH175, RH176, RH177, RH178 are effective only on the PLC controlled axes selected at flags Y630, ..., Y637.*

RH179	

RH180	7th axis position command lower word

RH181	7th axis position command upper word

In case of PLC controlled axes the interpolator moves into the position entered here interpreted in increment, or absolute value in function of the state of flag Y963. The interpretation of the position data is output increment.

RH182	7th axis feedrate command lower word

RH183	7th axis feedrate command upper word

In case of PLC controlled axes the axis moves at the rate entered here provided flag Y962 is set to 1. Interpretation of 1 unit of the rate parameter (RH182=1, RH183=0):

$$\frac{0.2 inputincrement}{\min}$$

☞ *Registers RH180, RH181, RH182, RH183 are effective only on the PLC controlled axes selected at flags Y630, ..., Y637.*

RH184	

RH185	8th axis position command lower word

RH186	8th axis position command upper word

In case of PLC controlled axes the interpolator moves into the position entered here interpreted in increment, or absolute value in function of the state of flag Y973. The interpretation of the position data is output increment.

RH187	8^{th} axis feedrate command lower word

RH188	8^{th} axis feedrate command upper word

In case of PLC controlled axes the axis moves at the rate entered here provided flag Y972 is set to 1. Interpretation of 1 unit of the rate parameter (RH187=1, RH188=0):

$$\frac{0.2 inputincrement}{\min}$$

☞ *Registers RH185, RH186, RH187, RH188 are effective only on the PLC controlled axes selected at flags Y630, ..., Y637.*

RH189	

RH190	Number of axis doing ovality

Write into this register the physical number of axis doing ovality during piston turning (Y674=1). It can be used only with digital CANXMU board, the number must be odd and the next physical axis must be left empty in point of view of data output. If e. g. axis 3 is doing ovality the following parameter values must be set: 4863 DIGITAL3=1, 4864 DIGITAL4=0 és RH190=3.

RH191	Position of longer diameter

The number specified by this register is equal to the distance between the zero pulse of spindle encoder and position of the longer diameter of ellipse in unit of encoder counts. This value is varied between mechanisms therefore it is recommended to get it from a CONST parameter.

RH192	Ovality lower word

RH193	Ovality higher word

When in piston turning mode (Y674=1) PLC must copy the position of axis doing ovality into these registers in modul :002.
If e. g. ovality is programmed on address "A" that is axis "A" is doing ovality and axis "A" is the physical axis № 4 the following parameters must be set: 4287 A=4, 4444 AXIST4=1, 4464

NOLOOP4=1, 4864 DIGITAL4=0.
Parameter NOLOOP is 1 because the NC does not close the position control loop on the axis doing ovality it is done by the drive. PLC must copy the position in modul :002 because the control during the execution of blocks

G1 X__ Z__ A__

continously changes the value of ovality (A).
In our case commands

LRH115
SRH192
LRH116
SRH193

are doing this task.

RH194	Barrellity lower word

RH195	Barrellity higher word

In piston turning mode (Y674=1) these registers are used if axis X must be clamped when the oscillation of axis doing ovality reacts on the position of axis X. Then barrel shape must be programmed by means of axis doing ovality.
It is the best if barrel shape is programmed on address "U" therefore set parameters: 4284 U=3 (RH190=3), 4444 AXIST3=1, 4464 NOLOOP4=0, 4864 DIGITAL3=1.
Before setting piston turning mode (Y674) axis U works like a normal NC axis.
Before setting piston turning mode by command U674 position control loop must be opened by the instruction U622. From now on position of axis U must be copied into registers above in modul :002. In this case:

LRH110
SRH194
LRH111
SRH195

NC can be programmed by command block G1 U__ Z__ A__.
After resetting piston turning mode (D674), PLC program must wait until oscillation of axis doing ovality stops then close position control loop by command D622 in our case.

RH196	

RH197	

RH198	

RH199	

2.3 Local Variables of PLC Program

1000 bytes of the PLC program form the freely available RAM area. Reference can be made to the bytes of this area by means of character F and 3 decimal digits:

Fpqr

pqr=000,001,...999

If a bit within the byte is to be referred to a fourth digit must be entered into the end of the number (s), and the value of s is octal:

Fpqrs

s=0,1,...,7

The selected area is basically divided into two parts. Variables from F000 to F499 are automatically vacanted when the power is turned on. The contents of variables from F500 to F999 are preserved upon power-off.

Most variables are freely available, however there are ones with special availability. The following table shows the usage of these variables .

Division of Local Variables

Variable Identity	Usage	Type
F000	Auxiliary register OP	Volatile variables
F001		
F002	Reserved for later use	
F003		
F004	Status register	
F005		
F006	Reserved for later use	
F007		
F008	Message register of operations	
F009		
F010	Freely available working area	
....		
F499		
F500	Tool pot table	Non-volatile Variables
....		
F(500+MAGAZIN*2+1)		
F[500+(MAGAZIN+1)*2]	Freely available table of PLC program	
....		
F[500+(MAGAZIN+2+PLC_TAB)*2]		
F[500+(MAGAZIN+4+PLC_TAB)*2]	Freely available working area	
....		
F999		

2.3.1 Auxiliary Register OP and Status Register

F000, F001: Auxiliary register OP
In case of multiplication of the contents of OP (statement *L[variable]), if the result does not have enough room into register OP, the high-words of the product can be found at this register: the low-byte at F000, the high-byte at F001.
In case of division of the contents of OP (statement /L[variable]) the low-byte of the remainder can be found in variable F000, while the high-byte in variable F001.

F004, F005: Status register
In the course of PLC program execution the following flags are set in function of the given statement:

Flag Identity	Meaning of Flag if Value=1 (TRUE)
F0040	Carry
F0041	
F0042	
F0043	
F0044	
F0045	
F0046	Result of statement: zero
F0047	Sign

Flag Identity	Meaning of Flag if Value=1 (TRUE)
F0050	
F0051	
F0052	
F0053	Overflow
F0054	
F0055	
F0056	
F0057	

F0040: Carry
The flag is set (=1) in the following cases:
- carry has been done in case of statement +,
- and borrow in case of statement -

F0046: Result of statement: zero
This flag is set to 1 if the result of statements +, -, ADDnnn, SUBnnn, CMPnnn is zero.

F0047: Sign
This flag is set to 1 if bit No. 15 of OP is 1 in case of statements +, -, ADDnnn and SUBnnn.

F0053: Overflow
This flag is set to 1 if the result of statement MULnnn is overflown.

F008, F009: Message register of statements

Flag Identity	Meaning of Flag if Value=1 (TRUE)
F0080	Syntax error
F0081	Data not found
F0082	Not BCD number
F0083	Overflow in case of statement *
F0084	
F0085	
F0086	
F0087	Sign of BCD number

F0080: Syntax error
This flag is set if error occurs during program execution in case of statements, where no fundamental syntax examination can be done in the course of compilation.
These statements:
LFInnn, SFInnn, /, HFnnn, PFnnn, MRnnn, MWnnn, ADDnnn, SUBnnn, MULnnn, DIVnnn, CMPnnn.
Details of the flag can be found in the description of the given statement.

F0081: Data not found
This flag is set to 1 if the data searched for cannot be found in case of search statements HFnnn, PFnnn.

F0082: Not BCD number
The flag is set if
- in the course of statement BIN the contents of OP is not BCD,
- in indirect address statements no address BCD is found.

F0083: Overflow in case of statement *
This flag is set to 1 if the result of * (multiplication) does not have enough room into register OP and the high-words can be found at addresses F000, F001.

F0087: Sign of BCD number
If a BCD number is to be converted to binary form by means of statement BIN the sign of BCD number must be entered into flag F0087:
- F0087=0: positive BCD number,
- F0087=1: negative BCD number.

2.3.2 Tool Pot Table

F500, ..., F[501+2*MAGAZINE]: Tool pot table
In case not local coded tool handle or random access magazine handle is to be used, a tool pot table is needed, in which the pot of the magazine and number of the tool found in it can be selected.

Note
Local coded tool handle means, *that reference to the tool is made at address T by the pot number of the magazine, in which the desired tool can be found.*
*If **tool reference** is **not local coded,** a table is needed, which shows, which tool number can be found in which pot of the magazine.*
Random access magazine handle *means, that the position of tools in magazine is not fixed. The returning tool (taken from spindle) is not taken back into the position it was taken out, but into the nearest vacant position in magazine, in the simplest case in place of the selected (new) tool.*

The tool pot table can be found among the SETTING screens on the TOOL POT TABLE screen, and can be filled out from the NC keyboard. For the NC sends always the code at address T to the PLC and the magazine handling should be done entirely in the PLC, the tool pot table is fully accessible for the PLC for writing and reading. Above all special handling commands ease the work of the PLC programmer.

The length of tool pot table can be set in parameter MAGAZINE No. 0061. In parameter MAGAZINE the number of tool pots in the magazine is to be entered. Row 0 of table shows the code of tool in spindle, i.e. the spindle is pot No. 0. The table has word-structure, therefore the length of table is 2*MAGAZINE+2 byte.

Reference to the row of the table can be made from the PLC program at address F and with the appropriate number. When editing, the serial numbering of the table is from 0 to the MAGAZINE value i.e. it corresponds to the word numbers. E.g. reference to the 3rd row of the table can be made in the PLC program by F506. The row numbers identify the pot numbers of the magazine.
.

Row No. 0, i.e. pot No. 0 indicates the spindle.
In all cases two data must be specified by every pot when editing:
- the number of the tool in pot,
- the width code of the tool.

The data structure is as follows:

```
            F(500+2i+1) | F(500+2i)

              1111 11
              5432 1098 7654 3210
ith row      [xxxx|xxxx|xxxx|xxxx]
              ||└───────────────┴───── the number of the tool in pot
              └┴────────────────────── the width code of the tool
```

The tool number is given in the memory in binary form.

Usage of width code

In case of local access magazine handle, i.e. the tool taken out is taken back into from where it was taken, the tool width, i.e. how many pots are being occupied is of no interest.
In case of random access magazine handle position may also have to be ensured for tools, that are more-than-one tool-pot-wide. Therefore tool positions, in which extra wide tools can be placed, should have to be selected. This is why a width code must be given to each tool in the tool pot table.
This is needed, for in case of random access magazine handle the returning tool can be taken to the place of the selected one, should the two tools have the same width code. If however the width of the two tools differ, the returning tool cannot be taken back in place of the selected one. In this case the - to the replacement - nearest vacant position, of which the width equals to that of the returning tool must be searched for.
The following width codes are enabled the table manager (under address L):

1 (normal width),
3, 5 or 7.

The tool that has a width of 3 occupies both to the left and to the right 1-1, that of the width of 5 occupies 2-2, while that of the width of 7 reserves 3-3 positions in the magazine. This way special pots can be selected in the magazine, into which the extra wide tools are placed.

The value for tool width entered into the table may be 1, 3, 5, or 7, the display and meaning of which is as follows:

coding in memory		The value in the table and the position reservation of tool in the magazine
15. bit	14. bit	
0	0	1
0	1	3
1	0	5
1	1	7

In case of extra wide tools to the pot number, into which the tool is taken also the tool number and the tool width code must be entered. As for the 1, 2 or 3 pot numbers before and after it, to the tool number 0, while to the width the appropriate width code is to be entered. If a tool is taken from the magazine to the spindle in row 0 the tool number and the width code also has to be entered, and the tool number is to be deleted in the row, from which the tool was taken. However the width code must be preserved in the table, for to show the returning tool, that the pots are reserved for extra wide tools.

2.3.3 Freely available Table of PLC Program

The length of freely available table can be entered into the parameter field at parameter PLC_TAB No. 0062, which can be found among the SETTING screens on the PLC TABLE screen. The table can be edited from the NC control and the data of the table can be accessed from the PLC program at address F and by entering the appropriate number. The freely available PLC table has also a word-structure, as is the tool pot table, this should be remembered when making references at address F. The length of the table is 2*PLC_TAB byte.

The freely available table is directly after the tool pot table in the memory:

start address: F[502+2*MAGAZINE]
end address: F[501+2*MAGAZINE+2*PLC_TAB]

If the value of parameter MAGAZINE is 0 the start and end addresses are as follows:

start address: F500
end address: F[499+PLC_TAB*2]

The serial numbering of the table in SETTING mode is from 1 to the PLC_TAB value and the value range of the data of the table:

0-65535

The usage of the table is freely available. Here for example data concerning the pot from where the tool in spindle was taken out or the number and width code of tool in each tool replacing stands can be stored.

2.4 Local Registers of PLC Program

2.4.1 Up/Down Counters

There are 32 pieces of 16-bit up/down counters available for the PLC programmer. The contents of the counter can be loaded and interrogated from the program. The value of counter can be incremented or decremented by means of PLC statements. As for the contents of the counter the condition testing statement can be given.
Reference to the counter can be made with its address (Q) and a two-digit decimal number:

Qnn
nn=00...31

2.4.2 20-msec Timers

There are 50 pieces of 20-msec timers available for the PLC programmer. The contents of the timer can be loaded and interrogated from the program. The contents of the timer is automatically decreased by one in every 20 msec. If the timer is terminated, i.e. its contents equals to 0, it does not turn over, but remains at 0 in the forthcoming timing units.
Reference to the 20-msec timer can be made with its address (T) and a two-digit decimal number:

Tnn
nn=00...49

2.4.3 Second Timers

There are 100 pieces of 16-bit 1-sec timers available for the PLC programmer. The contents of the timer can be loaded and interrogated from the program. The contents of the timer is automatically decreased by one in every 1 sec. If the timer is terminated, i.e. its contents equals to 0, it does not turn over, but remains at 0 in the forthcoming timing units.
Reference to the 1-sec timer can be made with its address (H) and a two-digit decimal number:

Hnn
nn=00...99

2.4.4 Minute Timers

There are 10 16-bit minute timers available for the PLC programmer. The contents of the timer can be loaded and interrogated from the program. The contents of the timer is automatically decreased by one in every minute. If the timer is terminated, i.e. its contents equals to 0, it does not turn over, but remains at 0 in the forthcoming timing units.
Reference to the minute timer can be made with its address (M) and a one-digit decimal number:

Mn
n=0...9

2.4.5 PLC Constants

There are 40 pieces of 16-bit constants available for the PLC programmer. The constants can be found at parameter groups **0001 CONST** and **0011 CONST2**. The difference between the two groups is that the first 10 constants i.e. those of group **0001 CONST** are operator's parameters, while those of group **0011 CONST2** are not.
The constants can be edited from the NC keyboard. Reference to the constant can be made in the program with its address (PR) and a three-digit number (the first digit is always 0):

RP0pq
pq=1...40

3 Standard Modules of PLC Program

3.1 Module :000

Module :000 is executed on level No. 0, i.e. in the rest time of T msec after the PLC module level No. 1 (module :001) has been executed (see: chapter 1.2 on page 8). Module :000 is not obligatorily executed in one time slice, its execution can extend over more time slices. In case level No. 0 has been terminated the rest time of PLC is returned to the NC. The module start is defined by label

```
:000
```

and its end by statement

```
J0
```

in the source language text of PLC program.
In module :000 the state of interface input lines and input flags is updated only in the first PLC time slice following the termination of the module (statement J0). *Thus in the same PLC time slice the result of condition test Innn may differ depending on the test done in module :001 or :000.*
Module :000 (level No. 0) can be used for tasks, the execution of which takes longer time.

3.2 Module :001

Module :001, i.e. the PLC level No. 1 is executed from the beginning in every PLC time slice, that is in every 20 msec. The execution of this module is mandatory in every PLC time slice . In case it is not done error message PLC TIMEOUT1 is displayed by the control. The beginning of module :001 is indicated by label

```
:001
```

and its end by statement

```
J1
```

in the source language text of PLC program.
In module :001 the state of interface input lines and input flags is updated in every PLC time slice.
It follows that module :001 (level 1.) is advisable to use for supervisory activities. Such actions may be the test of alarms, limits, signals of reference position switches and machine NC control buttons, as well as receiving the commands sent by the NC in the course of block execution.

3.3 Module :002

Module :002 makes it possible to react input signals in extreme situations as fast as possible..
Module :002 is called by the NC in every 5 (2) msec provided module call is enabled. Module :002 must be executed in 5 (2) msec, otherwise error message PLC TIMEOUT2 is displayed by the NC. The beginning of level No. 2 is indicated by label

```
:002
```

while its end by statement

```
J2
```

in the source language text of the PLC program. Call of module :002 is enabled or disabled by flag Y546.
Naturally in this module direct loading (Ppqr) and storing statements (UOpqr, DOpqr) are to be used.

3.4 Module :197

If output flag Y524 is 1 (PLC defined buttons with softkeys) signals of the 8 freely available softkeys offered by the NC are transferred through flags I500, ..., I507 by the NC (if Y524=0 these softkeys are not offered by the NC.). The caption of softkeys can be defined by the PLC programmer in module

:197.

The strings are separated by commas

,

and the last string together with module :197 is terminated by character

$.

The maximum length of captions is 9 characters. E.g.:

```
:197PLC1,PLC2,PLC3,PLC4,PLC5,PLC6,PLC7,PLC8$
```

The statuses of softkeys can be switched through flags Y500, ..., Y507.

3.5 Module :198

With the help of flags Y700, ..., Y707 8 different user messages indexed according to the contents of register RH090, ..., RH097 can be displayed on the screen containing user messages. Of the maximum 8 messages only the one displayed in the 2nd line of screen is active. (There is no need to switch over to the screen containing user messages in order to read the active message.)
The active message can be read at flags I700, ..., I707 of which the state of only one can be TRUE. The PLC programmer must take care of canceling the message. E.g. if one message is for tool replacement it is useful to cancel the active message by means of START button. A message display can be deleted (DY7nn) before it becomes active in case the reason of the message has ceased. Naturally in this case it also is deleted from the screen listing the messages.
The message strings must be entered into module

:198.

The strings are separated by commas

,

The maximum length of message strings is 20 characters. The end of module together with the last message is indicated by character

$.

E.g.:

```
:198MESSAGE1,MESSAGE2,...,MESSAGE8$
```

3.6 Module :199

152 different user message can be displayed on the screen containing user messages with the help of flags Y710, ..., Y897. Of the maximum 152 messages only one, displayed in the 2nd line of screen, is active. (For reading the active message there is no need to switch over to the screen containing the user messages.)
Due to this only one of flags I710, ..., I897 has TRUE state. It is the task of the PLC programmer to define the method of canceling the user messages. To cancel an error message also the RESET button, the signal of which is sent through input flag I477 can be used. A message flag can be canceled (DY7nn or DY8nn) before it becomes active in case the reason of the message has ceased. Naturally in this case it also is deleted from the screen listing the messages.
The message string must be entered into module

:199.

The strings are separated by commas

,

The maximum length of message strings is 20 characters. The end of module together with the last message is indicated by character

$

E.g.:

```
:199MESSAGE1,MESSAGE2,...,MESSAGE152$
```

3.7 Module :200

The information part of PLC program can be written in module :200. The information part, i.e. the text written in module :200 as well as the date and time of the compilation of the program, which is automatically generated by the compiler is displayed by selecting the SERVICE - PLC screen on the control.
The information text must be written in module

:200.

The end of module is indicated by character

$.

4 Instruction Set of PLC Program Language

4.1 Switch Statements

Upqr: switching interface output line or output flag Ypqr on

Switching interface output line on

Statement

Upqr (p=0,1,2,3)

switches the appropriate interface output Ypqr on, i.e. 24V occurs in the output line. The statement switches directly only the in-RAM-stored flag of interface output line on. The actual switch-on of the interface output appears only at the end of PLC time slice, when the output lines are updated from RAM flags by the NC. Therefore there is a lag between the statement execution and the switch-on of output line, the maximum time length of which is T msec (see: chapter 1.2 on page 8).

Switching output flag on

Statement

Upqr (p=4,5,6,7,8,9)

sets the appropriate output flag Ypqr to 1, i.e. to TRUE state.

Dpqr: switching interface output line or output flag Ypqr off

Switching interface output line off

Statement

Dpqr (p=0,1,2,3)

switches the appropriate interface output line Ypqr off. The statement switches directly only the in-RAM-stored flag of interface output line off. The actual switch-off of the interface output line appears only at the end of PLC time slice, when the output lines are updated from its RAM flags by the NC. Therefore there is a lag between the statement execution and the switch-off of output line, the maximum time length of which is T msec (see: chapter 1.2 on page 8).

Switching output flag off

Statement

Dpqr (p=4,5,6,7,8,9)

sets output flag Ypqr to 0, i.e. to FALSE state.

UFnnni: switching i^{th} bit of local variable on.

Statement

UFnnni (i=0,1,...,7)

sets the i^{th} bit of local variable Fnnn to 1, i.e. to TRUE state.

DFnnni: switching i^{th} bit of local variable off.

Statement

DFnnni (i=0,1,...,7)

sets the i^{th} bit of local variable Fnnn to 0, i.e. to FALSE state.

UOpqr: switching interface output line Ypqr on directly.
Statement

UOpqr (p=0,1,2,3)

switches the appropriate interface output line Ypqr on directly. The statement switches directly the interface output, i.e. not the in-RAM-stored flag of interface output line flag on. Contrary to statement Upqr the execution of statement UOpqr is five times slower, therefore it is advisable to use statement UOpqr in case prompt intervention is necessary in the output line. The statement can be applied only for interface output lines excluding output flags.

DOpqr: switching interface output line Ypqr off directly.
Statement

DOpqr (p=0,1,2,3)

switches the appropriate interface output line Ypqr off directly. The statement switches directly the interface output, i.e. not the in-RAM-stored flag of interface output line off. Contrary to statement Dpqr the execution of Statement DOpqr is five times slower, therefore it is advisable to use statement DOpqr in case prompt intervention is necessary in the output line. The statement can be applied only for interface output lines excluding output flags.

4.2 Condition Testing Statements

There may be two kinds of conditional program branches:

<condition> [true branch of condition] **E** [false branch of condition] **Z**
In case the <condition> is true the program execution is continued with statements of true branch between <condition> and marker E, than the program execution is continued with the statements after marker Z.
Otherwise, if <condition> is not true the program execution is continued with statements of false branch between marker E and marker Z, than the program execution is continued with the statements after marker Z.

<condition> [true branch of condition] **Z**
In case the <condition> is true the statements of true branch are executed, than the program execution is continued with the statements after marker Z.
Otherwise, if <condition> is not true the program execution is continued with the statements after marker Z, thus the statements between <condition> and marker Z are not executed.

E: else marker of the FALSE branch of condition, the use of which is not obligatory. If it is lacking the program searches for the FALSE path after the end marker of conditional program branch.

Z: end marker of conditional program branch, the use of which is obligatory. In the program the number of markers Z should be as much as the number of opening conditions. If there are less markers Z in the program than the number of opening conditions the compiler sends message "ERROR 17" and the cursor is flashed at the beginning of the erroneous condition. If there are more markers Z in the program than the number of opening conditions, then the compiler sends message "ERROR 2".

4.3 Creating Conditions with Flags

Ipqr: state test of interface input line or input flag Ipqr

State test of interface input line

The first statement of conditional program branch

Ipqr [Ipqr=true branch] *E* [Ipqr=false branch] *Z*, or
Ipqr [Ipqr=true branch] *Z*
p=0,1,2,3

performs state test of interface input line. If 24V occurs in the input line the condition is TRUE, if the input line is interrupted the condition is FALSE. The statement tests the in-RAM-stored synchronized flag interface input line.

State test of input flag

The first statement of conditional program branch

Ipqr [Ipqr=1 branch] *E* [Ipqr=0 branch] *Z*, or
Ipqr [Ipqr=1 branch] *Z*
p=4,5,6,7,8,9

performs state test of input flag Ipqr. The statement tests the synchronized state of input flags.

Note

The result of the state test of input lines or input flags also depends on whether the state test is done in module :000 or :001. In module :000 the in-RAM-stored flag of input lines is updated at the beginning of the first PLC time slice following the execution of statement J0, while in module :001 at the beginning of every PLC time slice.

Example:

I002 U012 E D012 Z

If there is 24V in input line I002 output line Y012 is switched on, if not, output line Y012 is switched off.

Ypqr: state test of interface output line or output flag Ypqr

State test of interface output line

The first statement of conditional program branch

Ypqr [Ypqr=true branch] *E* [Ypqr=false branch] *Z*, or
Ypqr [Ypqr=true branch] *Z*
p=0,1,2,3

performs state test on the in-RAM-stored flag of interface output line Ypqr. Therefore the state test can signal switched-on or switched-off state even when the output line is physically not switched on or off yet. If the output line is on the condition is TRUE, if the output line is interrupted the condition is FALSE.

State test of output flag

The first statement of conditional program branch

Ypqr [Ypqr=true branch] *E* [Ypqr=false branch] *Z*, or
Ypqr [Ypqr=true branch] *Z*
p=4,5,6,7,8,9

performs state test on output flag Ypqr.

Vpqr: Change test of interface input line or input flag Ipqr

Change test of interface input line

The first statement of conditional program branch

Vpqr [Ipqr changed branch] *E* [Ipqr not changed branch] *Z*, or

Vpqr [Ipqr changed branch] *Z*

p=0,1,2,3

performs change test of interface input line Ipqr. The current state of the in-RAM-stored flag of the interface input line is compared to the 20-msec earlier state, provided the change test has occurred in module :001. If the change test appears in module :000 the current synchronized state is compared to the previous state. The condition is TRUE if change had occurred.

Change test of input flag

The first statement of conditional program branch

Vpqr [Ipqr changed branch] *E* [Ipqr not changed branch] *Z*, or

Vpqr [Ipqr changed branch] *Z*

p=4,5,6,7,8,9

performs state test on the edge of input flag Ipqr. The function of the statement corresponds to that of interface input line.

Ppqr: direct state test of interface input line

The first statement of conditional program branch

Ppqr [Ipqr=true branch] *E* [Ipqr=false branch] *Z*, or

Ppqr [Ipqr=true branch] *Z*

p=0,1,2,3

performs direct state test of interface input line Ipqr. If 24V occurs in the input line the condition is TRUE, if the input line is interrupted the condition is FALSE. The statement tests directly the input line of interface board, not the flag stored in RAM. Naturally the statement cannot be applied for testing input flags.

Fnnni: State test of the ith bit of local variable Fnnn

The first statement of conditional program branch

Fnnni [Fnnni=true branch] *E* [Fnnni=false branch] *Z*, or

Fnnni [Fnnni=true branch] *Z*

i=0,1,...,7

performs state test on the ith bit of local variable Fnnn. If it is 1 the condition is TRUE.

N<condition>**: complemented state test of flag**

The state and change tests can be performed also on the complemented state of flags provided operator N is used:

NIpqr [Ipqr=false branch] *E* [Ipqr=true branch] *Z*, or

NIpqr [Ipqr=false branch] *Z*

NYpqr [Ypqr=false branch] *E* [Ypqr=true branch] *Z*, or

NYpqr [Ypqr=false branch] *Z*

NVpqr [Ipqr not changed branch] *E* [Ipqr changed branch] *Z*, or

NVpqr [Ipqr not changed branch] *Z*

NPpqr [Ipqr=false branch] *E* [Ipqr=true branch] *Z*, or

NPpqr [Ipqr=false branch] *Z*

NFnnni [Fnnni=false branch] *E* [Fnnni=true branch] *Z*, or

NFnnni [Fnnni=false branch] *Z*

Naturally direct state test can also be applied for these interface input lines.

4.4 Combination of Conditions with Logic Gates on Flags

(<1st condition> **A** <2nd condition>**): logical AND of two conditions**

The first statement of conditional program branch

(<1st condition> *A* <2nd condition>) [true branch] *E* [false branch] *Z*

(<1st condition> *A* <2nd condition>) [true branch] *Z*

performs state test of the two conditions combined with AND gate. The condition between parentheses (,) is true if both elements are TRUE. For example:

(I002 A Y014) UF0103 Z

If 24V occurs in input line I002 and output line Y014 is on, then bit 3 of variable F010 is switched to 1.

(<1st condition> **O** <2nd condition>**): logical OR of two conditions**

The first statement of conditional program branch

(<1st condition> *O* <2nd condition>) [true branch] *E* [false branch] *Z*

(<1st condition> *O* <2nd condition>) [true branch] *Z*

performs state test of the two conditions combined with OR gate. The condition between parentheses (,) is true if at least one of the conditions is TRUE. For example:

(I002 O Y014) UF0103 Z

If 24V occurs in input line I002 or output line Y014 is on, then bit 3 of variable F010 is switched to 1.

(<1st condition> **X** <2nd condition>**): logical eXclusive or of two conditions**

The first statement of conditional program branch

(<1st condition> *X* <2nd condition>) [true branch] *E* [false branch] *Z*

(<1st condition> *X* <2nd condition>) [true branch] *Z*

performs state test of the two conditions combined with EXCLUSIVE OR gate. The condition between parentheses (,) is true if one of the conditions is TRUE, while the other one is FALSE. For example:

(I002 X Y014) UF0103 Z

If 24V occurs in input line I002 and output line Y014 is off, then bit 3 of variable F010 is switched to 1.

(..): parentheses, combining more conditions into one condition.

More conditions can be combined by means of open and close parentheses. The maximum number of combined conditions is not defined and the logic gates combining the conditions can also be miscellaneous. When calculating a condition its result is calculated going from left to right. Condition

```
(I001 A Y012 A F1002 O I002)
```

is TRUE if I001, Y012 and F1002 are also true, or I002 is TRUE.

The maximum nesting depth is 8 parentheses. In this case the evaluation is started from the deepest parenthesis going from left to right. In Statement

```
((I001 O I002) A (Y015 A F1006))
```

first the result of condition (I001 O I002) than of condition (Y015 A F1006) is calculated, afterwards the two results are combined.

The open and close parentheses should always be in pairs.

4.5 Loading constant into register OP

,nnnnn: loading decimal constant into register OP

Decimal constant ,nnnnn written in the PLC program is converted by the compiler into binary form and loaded into register OP. The value range of the constant to be loaded:

,nnnnn = 0 - 65535,

that is only positive constant can be entered into OP. If the decimal constant is preceded by statement

<, >, =, <=, >=, +, −, *, /, N, A, O, X

marker "," of the decimal constant must not be entered before the constant, otherwise the compiler detects error.

.nnnn: loading hexadecimal constant into register OP

Hexadecimal constant .nnnnn written in the PLC program is converted by the compiler into binary form and loaded into register OP.. "." (point) indicates the hexadecimal constant. The value range of the constant to be loaded:

.nnnn = .0000 - .FFFF

The hexadecimal constant written into OP is always regarded by PLC statements as an unsigned number, thus .FFFF > .0 . Marker "." of the hexadecimal constant must always be entered before the constant.

4.6 Loading value of variable into register OP

Statement L loads the value of the addressed word or flag variable into register OP. After statement L reference to the variable can only be made by the identity number following the address of the variable. That is why this statement is called the direct loading of register OP.
If 3 digits are entered after the address of the variable (4 digits after address F) reference is made to the bit variable, the value of which is loaded into bit No. 0 of register OP. Bits No. 1...15 of register OP are cleared.
If 2 digits are entered after the address of the variable (3 digits after address F) reference is made to the word variable, the value of which is loaded into OP.
Indirect loading can be used in case of local variables Fnnn. In statement LFInnn value of Fnnn is aaa, which is referred to local variable Faaa and value of Faaa is loaded into register OP. That is why this statement is called indirect loading.
When loading the OP directly, i.e. in case of statement L reference can be made to the following variables:

LIpqr: bit-loading of the state of interface input line or input flag into OP

Loading state of interface input line into OP

Statement

LIpqr

p=0,1,2,3

loads the in-RAM-stored synchronized flag of the qr^{th} input line of the 1^{st}, ..., 4^{th} interface board specified by index p into bit No. 0 of the OP.

Loading input flag into OP

Statement

LIpqr

p=4,5,6,7

loads the in-RAM-stored synchronized flag of the pqr^{th} input line into OP.

Note
The same as in case of state test Ipqr.

LIpq: loading two neighboring bytes of interface input lines or input flags into OP

Loading a word from interface input lines into OP

Statement

LIpq

p=0,1,2,3

loads the in-RAM-stored synchronized bytes of the q^{th} and $(q+1)^{th}$ input bytes of the 1^{st}, ..., 4^{th} interface board specified by index p into OP.

Loading of the state of input flag into OP

Statement

LIpq

p=4,5,6,7

loads the in-RAM-stored synchronized bytes of the q^{th} and $(q+1)^{th}$ input bytes into OP.

Note
The same as in case of state test Ipqr.

LYpqr: bit-loading of the state of interface output line or output flag into OP

Loading of the state of interface output line into OP

Statement

LYpqr

p=0,1,2,3

loads the in-RAM-stored flag of the qr^{th} output line of the 1^{st}, ..., 4^{th} interface board specified by index p into bit No. 0 of the OP.

Loading of the state of output flag into OP

Statement

LYpqr

p=4,5,6,7

loads the in-RAM-stored flag of the pqr^{th} output line into OP.

LYpq: loading two neighboring bytes of interface output lines or output flags into OP

Loading a word from interface output lines into OP

Statement

LYpq

p=0,1,2,3

loads the in-RAM-stored bytes of the q^{th} and $(q+1)^{th}$ output bytes of the 1^{st}, ..., 4^{th} interface board specified by index p into OP.

Loading a word from output flags into OP

Statement

LYpq

p=4,5,6,7

loads the in-RAM-stored bytes of the q^{th} and $(q+1)^{th}$ output bytes into OP.

LVpqr: bit-loading of the change flag of interface input line or input flag into OP

Loading change state of interface input line into OP

Statement

LVpqr

p=0,1,2,3

tests the change of the in-RAM-stored synchronized flag of the qrth input line of the 1st, ..., 4th interface board according to the previous state. The current state of the interface input line is compared to the 20-msec-earlier state, provided the statement has occurred in module :001. If the statement occurs in module :000 the current synchronized state is compared to the previous synchronized state. The contents of OP is set to 1 if change has been detected.

Loading change state of input flag into OP

LVpqr

p=4,5,6,7,8,9

The same as in case of the interface input line.

LVpq: loading two neighboring bytes of change flags of interface input line or input flags into OP

Loading a word from change flags of interface input lines into OP

Statement

LVpq

p=0,1,2,3

tests the in-RAM-stored synchronized flag of the qth and (q+1)th input bytes of the 1st, ..., 4th interface board according to the previous state. The current state of the interface input line is compared to the 20-msec-earlier state, provided the statement has occurred in module :001. If the statement occurs in module :000 the current synchronized state is compared to the previous synchronized state. The bits, where change has been detected, are set to 1.

Loading a word from input flags into OP

LVpq

p=4,5,6,7,8,9

The same as in case of the interface input line.

LPpqr: direct bit loading of interface input line into OP

Statement

LPpqr

p=0,1,2,3

loads the qrth output line of the 1st, ..., 4th interface board specified by index p by testing directly the input line of the interface board. Naturally the statement cannot be applied in case of input flags.

LPpq: loading two neighboring bytes of interface input lines into OP directly

Statement

LPpq

p=0,1,2,3

loads the qth and (q+1)th output line of the 1st, ..., 4th interface board specified by index p by testing directly on the input line of the interface board, therefore it does not use in-RAM-stored synchronized flags of input lines. Naturally the statement cannot be applied in case of input flags.

LFpqri: loading the i^th bit of local variable into OP

Statement

LFpqri

loads the i^th bit of local variable Fpqr to bit No. 0 of register OP.

LFpqr: loading two neighboring bytes of local area into OP

Statement

*LFpqr*4

loads bytes Fpqr and Fpq(r+1) from local area into register OP

LRHinn: loading the contents of input or output register into OP

Statement

LRHinn

i=0, 1

nn=0, ..., 99

loads the contents of the addressed input or output register into register OP.

LQnn: loading the contents of up/down counter into OP

Statement

LQnn

nn=00, ..., 31

loads the contents of the addressed up/down counter into register OP.

LTnn: loading the contents of 20-msec timer into OP

Statement

LTnn

nn=00, ..., 49

loads the contents of the addressed 20-msec timer into register OP.

LHnn: loading the contents of second timer into OP

Statement

LHnn

n=00, ..., 99

loads the contents of the addressed second timer into register OP.

LMn: loading the contents of minute timer into OP

Statement

LMn

n=0, ..., 9

loads the contents of the addressed minute timer into register OP.

LRP0nn: loading PLC constant into OP

Statement

LRP0nn

nn=1, ..., 40

loads the contents of the addressed PLC constant into register OP.

LFInnn, loading indirect addressed word of local area into OP

This statement is for loading indirect addressed word of local area of the PLC program into OP. After the statement name (LFI) the address of the local variable, where the address of the data to be loaded can be found, needs to be entered with 3 decimal digits.

nnn: address of a local variable, where the address of the local variable to be loaded into OP can be found.

Flags to be set:

F0080: syntax error. The value of variable Fnnn is not in the range of 000...999.

F0082: the value of variable Fnnn is not decimal.

Example for the use of statement LFInnn:

```
 LFI128      ;loading the number and width code of the called tool
(F0080       ;if syntax error
OF0082)      ;or not decimal number
 U733        ;LOADING ERROR, message strobe set
E            ;if OK
 SF102       ;saving code of called tool
 ...
Z            ;end of syntax error condition
```

NL[variable], NLFInnn, loading complemented contents of the variable into OP

Statements NL[variable] (see types of variables above) and NLFInnn load the complemented value of the tested variable into register OP.

4.7 Storing Value from Register OP into Variable

Statement S stores the contents of register OP into the specified word or flag variable. Following statement name S reference to a variable can only be made by the identity number next to the address of the variable. That is why the statement is called direct storing.

If 3 digits are entered after the address of the variable (4 digits after address F) reference is made to a flag and bit No. 0 of register OP is stored into the specified flag.

If 2 digits are entered after the address of the variable (3 digits after address F) reference is made to a word variable and the contents of register OP is stored into the specified word.

Indirect loading can be used in case of local variables Fnnn. In statement SFInnn value aaa of Fnnn is referred to local variable Faaa and is executed in statement SFaaa. That is why this statement is called indirect storing.

In case of statement S the possible statement combinations are as follows:

SYpqr: storing bit No. 0 of the OP into interface output line or output flag

Storing bit No. 0 of the OP into interface output line

Statement

SYpqr

p=0,1,2,3

stores bit No. 0 of register OP into the in-RAM-stored flag of the qrth output line of the 1st, ..., 4th interface board specified by index p.

Storing bit No. 0 of the OP into output flag

Statement

SYpqr

p=4,5,6,7,8,9

stores bit No. 0 of register OP into the in-RAM-stored flag of the pqrth output flag.

SYpq: storing the contents of OP into two neighbouring bytes of interface output lines or output flags

Storing OP into a word of interface output lines

Statement

SYpq

p=0,1,2,3

stores the contents of register OP into in-RAM-stored bytes of the q^th^ and (q+1)^th^ output byte of the 1^st^, ..., 4^th^ interface board specified by index p.

Storing OP into a word of output flags

Statement

SYpq

p=4,5,6,7,8,9

stores the contents of register OP into the pq^th^ and p(q+1)^th^ output flag.byte.

SOpqr: storing bit No. 0 of OP directly into interface output line

Statement

SOpqr

p=0,1,2,3

stores the contents of bit No. 0 of register OP directly (by skipping the in-RAM-stored flags of the output lines) to the qr^th^ output line of the 1^st^, ..., 4^th^ interface board specified by index p. Contrary to statement SYpqr the execution of statement SOpqr is five times slower, therefore it is advisable to use statement SOpqr in case prompt intervention is necessary in the output line Naturally the statement cannot be applied in case of output flags.

SOpq: storing the contents of OP directly into two neighboring bytes of interface output lines

Statement

SOpq

p=0,1,2,3

stores the contents of register OP directly (by skipping the in-RAM-stored flags of the output lines) to the q^th^ and (q+1)^th^ output lines of the 1^st^, ..., 4^th^ interface board specified by index p. Contrary to statement SYpq the execution of statement SOpq is five times slower, therefore it is advisable to use statement SOpq in case prompt intervention is necessary in the output line Naturally the statement cannot be applied in case of output flags.

SFpqri: storing bit No. 0 of OP into the i^th^ bit of local variable

Statement

SFpqri

stores bit No. 0 of register OP into the i^th^ bit of the Fpqr byte of local area.

SFpqr: storing the contents of OP into two neighboring bytes of local area

Statement

SFpqr

stores the contents of register OP into the Fpqr and Fpq(r+1) byte of local area.

SRHinn: storing the contents of OP into output register

Statement

SRHinn

i=0, 1

nn=50, ..., 99

stores the contents of register OP into the addressed output register. Naturally the statement cannot be used in case of nn<50 (input register).

SQnn: storing the contents of OP into up/down counter

Statement

SQnn

nn=00, ..., 31

stores the contents of register OP into the addressed up/down counter.

STnn: storing the contents of OP into 20-msec timer

Statement

STnn

nn=00, ..., 49

stores the contents of register OP into the addressed 20-msec timer

SHnn: storing the contents of OP into second timer

Statement

SHnn

n=00, ..., 99

stores the contents of register OP into the addressed second timer.

SMnn: storing the contents of OP into minute timer

Statement

SMn

n=0, ..., 9

stores the contents of register OP into the addressed minute timer.

SFInnn, storing the contents of OP into indirectly addressed word of local area

This statement stores the contents of OP indirectly to one of the local variables. After the statement name (SFI) the address of the local variable, where the address of the data to be loaded can be found, needs to be entered with 3 decimal digits.

nnn: address of a local variable, where the address of the local variable, the contents of which is to be loaded into OP can be found.

Flags to be set:

F0080: syntax error. The value of variable Fnnn is not in the range of 000...999.

F0082: the value of variable Fnnn is not decimal.

Example for the use of Statement SFInnn:

```
 LF102        ;number of the called tool
 A.C000       ;preserving width code, cutting tool number
 SFI128       :clearing the called tool from tool pot table
(F0080        ;if syntax error
OF0082)       ;or not decimal number
 U732         ;STORING ERROR, message strobe set
E             ;if OK
 ...
Z             ;end of syntax error condition
```

NS[variable], NSFInnn, storing complemented contents of OP into variable
Statements NS[variable] (see types of variables above) and NSFInnn store the complemented value of register OP into the specified variable.

4.8 Arithmetic Statements with Register OP

+: adding constant or value of variable into register OP (sum into OP)
Constant or value of variable can be added to the contents of register OP:

Adding decimal constant into OP (OP=OP+decimal number)

Statement

+ *nnnnn* (nnnnn=0...65535)

adds decimal constant nnnnn to the contents of OP. The result can be found into register OP.

Adding hexadecimal constant into OP (OP=OP+hexadecimal number)

Statement

+ *.nnnn* (.nnnn=0000h...FFFFh)

adds hexadecimal constant .nnnn to the contents of OP. The result can be found into register OP.

Adding value of variable into OP (OP=OP+variable)

Statement

+ *L*[variable], or
+ *LFInnn*

adds the value of variable to the contents of OP in binary form. The result can be found into register OP. For syntax reasons the identity of variable must be substituted for the expression "loading value of variable into register OP" in the statement. This is formally the application of prefix L. Reference can be made to all the variables, the value of which can be loaded into OP: +LIpq, +LYpq, +LVpq, +LPpq, +LFpqr, +LRHipq, +LQnn, +LTnn, +LHnn, +LMn, +LRP0nn, +LFInnn.

Adding complemented value of variable into OP (OP=OP+Nvariable)

Statement

+ *NL*[variable]
+ *NLFInnn*

complements the value of variable (without changing the contents of the variable) and adds the result to the contents of OP in binary form. The result of addition can be found into register OP. For syntax reasons the identity of variable must be substituted for the expression "loading value of variable into register OP" in the statement. This is formally the application of prefix L. Reference can be made to all the variables, the value of which can be stored into OP:
+NLIpq, +NLYpq, +NLVpq, +NLPpq, +NLFpqr, +NLRHipq, +NLQnn, +NLTnn, +NLHnn, +NLMn, +NLRP0nn, +NLFInnn.

The following status flags can be tested after addition:

F0040=1, if carry has occurred
F0046=1, if OP=0 (result of statement is zero)
F0047=1, if OP<0 (result of statement is less than zero, i.e. bit No. 15 of OP is 1)

+: adding value of register OP into variable (Sum in variable)

Adding value of register OP into variable (variable=variable+OP)

Statement

+ *S*[variable], or
+ *SFInnn*

adds the contents of OP into the value of variable in binary form. The result can be found in the

variable (contents of OP remains unchanged). For syntax reasons the identity of variable must be substituted for the expression “storing value of variable into register OP” in the statement. This is formally the application of prefix S. Reference can be made to all the variables, to which reference with Statement S can be made:
+SYpq, +SOpq, +SFpqr, +SRHipq, +SQnn, +STnn, +SHnn. +SMn, +SFInnn.

Adding value of register OP into the bit-negated value of variable (variable=Nvariable +OP)

Statement

+ *NS*[variable]

+ *NSFInnn*

complements the value of variable and adds the contents of OP into the result of addition in binary form. The result can be found in the variable. For syntax reasons the identity of variable must be substituted for the expression “storing value of variable into register OP” in the statement. This is formally the application of prefix S. Reference can be made to all the variables, to which reference with Statement S can be made:
+NSYpq, +NSOpq, +NSFpqr, +NSRHipq, +NSQnn, +NSTnn, +NSHnn, +NSMn, +NSFInnn.

The following status flags can betested after addition:

F0040=1,if carry has occurred

F0046=1, if OP=0 (result of statement is zero)

F0047=1, if OP<0 (result of statement is less than zero, i.e. bit No. 15 of OP is 1)

–: subtracting constant or value of variable from register OP (difference into OP)

Constant or value of variable can be subtracted from the contents of register OP:

Subtracting decimal constant from OP (OP=OP-decimal constant)

Statement

- *nnnnn* (nnnnn=0...65535)

adds the two’s complement of decimal constant nnnnn to the contents of OP. The result can be found into register OP.

Subtracting hexadecimal constant from OP (OP=OP-hexadecimal constant)

Statement

- *.nnnn* (.nnnn=0000h...FFFFh)

adds the two’s complement of hexadecimal constant .nnnn to the contents of OP. The result can be found into register OP.

Subtracting value of variable into OP (OP=OP-variable)

Statement

- *L*[variable], or

- *LFInnn*

adds the two’s complement of the value of variable to the contents of OP. The result can be found into register OP. For syntax reasons the identity of variable must be substituted for the expression “loading value of variable into register OP” in the statement. This is formally the application of prefix L. Reference can be made to all the variables, the value of which can be stored into OP:
-LIpq, -LYpq, -LVpq, -LPpq, -LFpqr, -LRHipq, -LQnn, -LTnn, -LHnn, -LMn, -LRP0nn, -LFInnn.

Subtracting complemented value of variable from OP (OP=OP-Nvariable)

Statement

- *NL*[variable]
- *NLFInnn*

complements the value of variable (without changing the contents of the variable) and subtracts the result from the contents of OP in binary form. The result of subtraction can be found into register OP. For syntax reasons the identity of variable must be substituted for the expression "loading value of variable into register OP" in the statement. This is formally the application of prefix L. Reference can be made to all the variables, the value of which can be stored into OP: -NLIpq, -NLYpq, -NLVpq, -NLPpq, -NLFpqr, -NLRHipq, -NLQnn, -NLTnn, -NLHnn, -NLMn, -NLRP0nn, -NLFInnn.

The following status flags can be tested after subtraction:

F0040=1 ,if carry has occurred
F0046=1, if OP=0 (result of statement is zero)
F0047=1, if OP<0 (result of statement is less than zero, i.e. bit No. 15 of OP is 1)

–: subtracting value of register OP from variable (Sum in variable)

Subtracting value of register OP from variable (variable=variable-OP)

Statement

- *S*[variable], or
- *SFInnn*

subtracts the contents of OP from the value of variable in binary form. The result can be found in the variable (contents of OP remains unchanged). For syntax reasons the identity of variable must be substituted for the expression "storing value of variable into register OP" in the statement. This is formally the application of prefix S. Reference can be made to all the variables, to which reference with statement S can be made:
-SYpq, -SOpq, -SFpqr, -SRHipq, -SQnn, -STnn, -SHnn. -SMn, -SFInnn.

Subtracting value of register OP from the complemented value of variable (variable = Nvariable -OP)

Statement

- *NS*[variable]
- *NSFInnn*

complements the value of variable and subtracts the contents of OP from the result in binary form. The result of subtraction can be found in the variable. For syntax reasons the identity of variable must be substituted for the expression "storing value of variable into register OP" in the statement. This is formally the application of prefix S. Reference can be made to all the variables, to which reference with statement S can be made:
-NSYpq, -NSOpq, -NSFpqr, -NSRHipq, -NSQnn, -NSTnn, -NSHnn, -NSMn, -NSFInnn.

The following status flags can be tested after subtraction:

F0040=1, if carry has occurred
F0046=1, if OP=0 (result of operation is zero)
F0047=1, if OP<0 (result of operation is less than zero, i.e. bit No. 15 of OP is 1)

***: multiplying constant or value of variable by register OP**

The contents of register OP can be multiplied by constant or value of variable. The multiplication regards both the multiplicator and multiplicand as positive unsigned numbers. For it may take 32 bits to multiply two 16-bit numbers the lower word of the product is placed into register OP. In case overflow occurs, i.e. the product needs to store more than 16 bits the bits with higher local value can be found in bytes F000 and F001. The bits with 31...24 local value are in byte F001 byte, while those with 23...16 local value are in byte F000.

Multiplying decimal constant by OP (OP=OP*decimal constant)

Statement

* *nnnnn* (nnnnn=0...65535)

multiplies decimal constant nnnnn by the contents of OP. The result can be found into register OP, in case of overflow at variables F000, F001.

Multiplying hexadecimal constant by OP (OP=OP*hexadecimal constant)

Statement

* *.nnnn* (.nnnn=0000h...FFFFh)

multiplies hexadecimal constant .nnnn by the contents of OP. The result can be found into register OP, in case of overflow at variables F000, F001.

Multiplying value of variable by OP (OP=OP*variable)

Statement

* *L*[variable], or

* *LFInnn*

multiplies the value of variable by the contents of OP. The result of multiplication can be found into register OP, in case of overflow at variables F000, F001. For syntax reasons the identity of variable must be substituted for the expression "loading value of variable into register OP" in the statement. This is formally the application of prefix L. Reference can be made to all the variables, the value of which can be stored into OP:

*LIpq, *LYpq, *LVpq, *LPpq, *LFpqr, *LRHipq, *LQnn, *LTnn, *LHnn, *LMn, *LRP0nn, *LFInnn.

Multiplying complemented value of variable by OP (OP=OP*Nvariable)

Statement

* *NL*[variable]

* *NLFInnn*

complements the value of variable (without changing the contents of the variable) and multiplies the result by the contents of OP in binary form. The result of multiplication can be found into register OP, in case of overflow at variables F000, F001. For syntax reasons the identity of variable must be substituted for the expression "loading value of variable into register OP" in the statement. This is formally the application of prefix L. Reference can be made to all the variables, the value of which can be stored into OP:

*NLIpq, *NLYpq, *NLVpq, *NLPpq, *NLFpqr, *NLRHipq, *NLQnn, *NLTnn, *NLHnn, *NLMn, *NLRP0nn, *NLFInnn.

The following status flag can be tested after multiplication:

F0083=1, if OP is overflown. Its meaning: the result of multiplication does not have enough room into OP, the bits with higher local values can be found at addresses F000, F001.

/: division

The contents of registers F001, F000 and OP can be divided by constant or value of variable. F001 byte contains bits 31...24, while F000 byte bits 23...16 of the dividend. The division regards both the divisor and dividend as positive unsigned numbers. The result of the statement can be stored into two 16-bit registers. The OP contains the quotient and variables F000 and F001 contain the remainder. The bits 15...8 of the remainder are in byte F001, while bits 7...0 are in byte F000.

☞ *Note: before using instruction / always to be considered whether the contents of variables F000 and F001 are the part of the dividend and if not they must be zeroed.*

Dividing OP by decimal constant (OP=OP/decimal constant)

Statement

/ nnnnn (nnnnn=0...65535)

divides the contents of registers F000, F0001 and OP by decimal constant nnnnn. The quotient can be found into register OP, while the remainder at variables F000, F001.

Dividing OP by hexadecimal constant (OP=OP/hexadecimal constant)

Statement

/ .nnnn (.nnnn=0000h...FFFFh)

divides the contents of registers F000, F0001 and OP by hexadecimal constant .nnnn. The quotient can be found into register OP, while the remainder at variables F000, F001.

Dividing OP by value of variable (OP=OP/variable)

Statement

/ L[variable], or

/ LFInnn

divides the contents of registers F000, F0001 and OP by the value of variable. The quotient can be found into register OP, while the remainder at variables F000, F001. For syntax reasons the identity of variable must be substituted for the expression "loading value of variable into register OP" in the statement. This is formally the application of prefix L. Reference can be made to all the variables, the value of which can be stored into OP:

/LIpq, /LYpq, /LVpq, /LPpq, /LFpqr, /LRHipq, /LQnn, /LTnn, /LHnn, /LMn, /LRP0nn, /LFInnn.

Dividing OP by complemented value of variable (OP=OP/Nvariable)

Statement

/ NL[variable]

/ NLFInnn

complements the value of variable (without changing the contents of the variable) and divides the contents of registers F000, F0001 and OP by the result in binary form. The quotient can be found into register OP, while the remainder at variables F000, F001. For syntax reasons the identity of variable must be substituted for the expression "loading value of variable into register OP" in the statement. This is formally the application of prefix L. Reference can be made to all the variables, the value of which can be stored into OP:

/NLIpq, /NLYpq, /NLVpq, /NLPpq, /NLFpqr, /NLRHipq, /NLQnn, /NLTnn, /NLHnn, /NLMn, /NLRP0nn, /NLFInnn.

The following status flag can be tested after multiplication:

F0080=1, if the divisor is zero, i.e. division is to be done by 0.

<<nn: shifting contents of OP into the left

Statement

<<nn (0< nn <15)

shifts the contents of OP into the left with the specified number of bits while filling vacanted bit positions with zero. The statement equals to division by 2^{nn}

>>nn: shifting contents of OP into the right

Statement

>>*nn* (0< nn <15)

shifts the contents of OP into the right with the specified number of bits while filling vacanted bit positions with zero. The statement equals to multiplication by 2^{nn}

BIN: converting the contents of register OP from BCD into binary form

The maximal value of register OP in BCD can be 9999, If negative BCD value is to be converted flag F0087 must be set to 1 before issuing Statement BIN. Thus

F0047 = 1 (OP<0)

has a meaning for the convert.

The following status flag can be tested after BIN statement:

F0082=1, if the number to be converted into binary form is not BCD

F0046=1, if OP=0 (result of statement is zero)

F0047=1, if OP<0 (result of statement is less than zero, i.e. bit No. 15 of OP is 1)

BCD: converting the contents of register OP from binary form into BCD

It converts the binary contents of register OP into BCD. The result of conversion, i.e. the value range of the contents of OP: -9999 < OP < 9999. The sign of the BCD number can be read at status flag F0047. After conversion the state of status flags must be evaluated.

The following status flag can be tested after BCD statement:

F0046=1, if OP=0 (result of statement is zero)

F0047=1, if the BCD number into OP is negative

F0053=1, overflow, i.e. the binary contents of OP: OP<-9999, or OP>9999.

[...]: parenthesing of the arithmetic operations executed into register OP

The arithmetic statements executed into register OP can be connected optionally, as for e.g.:

LF020 + LF022 * LF024
SF026

The execution order of statements goes from left to right. In the above example first bytes F020, F021 are stored into OP, adds to bytes F022, F023, than multiplies the result into OP by the contents of bytes F024, F025. This calculated OP contents is stored into variables F026, F027. If the above execution order is unsatisfactory, parentheses need to be used.

The maximum nesting depth of parenthesed arithmetic expressions is 8. Calculation of the value of OP is started from the deepest parenthesed expression:

[[LF020 + LF022] * LF024]
SYF026

In the above example first the addition is calculated, than the sum is multiplied by the contents of bytes F024, F025. The value of the result is stored into bytes F026, F027.

Note: in the arithmetic statement chain there may also be logic statements.

4.9 Logic Statements with Register OP

A: logical AND, result of statement into register OP

Constant or value of variable and the contents of register OP can be gated with AND:

Decimal constant and OP gated with AND (OP=OP A decimal constant)

Statement

A nnnnn (nnnnn=0...65535)

gates decimal constant nnnnn and the contents of OP with AND. The statement is executed for each bit: bit No. 0 of OP with bit N.0 of constant, and so on. The result can be found into register OP.

Hexadecimal constant and OP gated with AND (OP=OP A hexadecimal constant)

Statement

A .nnnn (.nnnn=0000h...FFFFh)

gates hexadecimal constant .nnnn and the contents of OP with AND. The statement is executed for each bit: bit No. 0 of OP with bit N.0 of constant, and so on. The result can be found into register OP.

Value of variable and OP gated with AND (OP=OP A variable)

Statement

A L[variable], or

A LFInnn

gates the value of variable and the contents of OP with AND.

If 3 digits are entered after the address of the variable (4 digits after address F) reference is made to the flag and only bit No. 0 of register OP participates in the statement.

If 2 digits are entered after the address of the variable (3 digits after address F) reference is made to the word-variable In this case the statement is executed for each bit: bit No. 0 of OP with bit No. 0 of the value of variable, and so on. The result can be found into register OP. For syntax reasons the identity of variable must be substituted for the expression "loading value of variable into register OP" in the statement. This is formally the application of prefix L. Reference can be made to all the variables, the value of which can be stored into OP:

ALIpq, ALYpq, ALVpq, ALPpq, ALFpqr, ALRHipq, ALQnn, ALTnn, ALHnn, ALMn, ALRP0nn, ALFInnn.

Complemented value of variable and OP gated with AND (OP=OP A Nvariable)

Statement

A NL[variable]

A NLFInnn

complements the value of variable (without changing the contents of the variable) and gates the result with AND to the contents of OP in the above mentioned way. The result of statement A can be found into register OP. For syntax reasons the identity of variable must be substituted for the expression "loading value of variable into register OP" in the statement. This is formally the application of prefix L. Reference can be made to all the variables, the value of which can be stored into OP:

ANLIpq, ANLYpq, ANLVpq, ANLPpq, ANLFpqr, ANLRHipq, ANLQnn, ANLTnn, ANLHnn, ANLMn, ANLRP0nn, ANLFInnn.

A: logical AND, result of statement in variable

OP and value of variable gated with AND (variable=variable A OP)

Statement

A S[variable], or

A SFInnn

gates the contents of OP and the value of variable with AND.

If 3 digits are entered after the address of the variable (4 digits after address F) reference is made to the flag and only bit No. 0 of register OP participates in the statement.

If 2 digits are entered after the address of the variable (3 digits after address F) reference is made to the word-variable In this case the statement is executed for each bit: bit No. 0 of OP with bit No. 0 of the value of variable, and so on. The result can be found in the variable. For syntax reasons the identity of variable must be substituted for the expression “storing value of variable into register OP” in the statement. This is formally the application of prefix S. Reference can be made to all variables with statement S:

ASYpq, ASOpq, ASFpqr, ASRHipq, ASQnn, ASTnn, ASHnn, ASMn, ASFInnn.

OP and complemented value of variable gated with AND (variable=Nvariable A OP)

Statement

A NS[variable]

A NSFInnn

complements the value of variable (without changing the contents of the variable) and gates the contents of OP and the result with AND in the above mentioned way. The result of statement A can be found in the variable. For syntax reasons the identity of variable must be substituted for the expression “storing value of variable into register OP” in the statement. This is formally the application of prefix S. Reference can be made to all variables with statement S:

ANSYpq, ANSOpq, ANSFpqr, ANSRHipq, ANSQnn, ANSTnn, ANSHnn, ANSMn, ANSFInnn.

O: logical OR, result of statement into register OP

Constant or value of variable and the contents of register OP can be gated with OR:

Decimal constant and OP gated with OR (OP=OP O decimal constant)

Statement

O nnnnn (nnnnn=0...65535)

gates decimal constant nnnnn and the contents of OP with OR. The statement is executed for each bit: bit No. 0 of OP with bit N.0 of constant, and so on. The result can be found into register OP.

Hexadecimal constant and OP gated with OR (OP=OP O hexadecimal constant)

Statement

O .nnnn (.nnnn=0000h...FFFFh)

gates hexadecimal constant .nnnn and the contents of OP with OR. The statement is executed for each bit: bit No. 0 of OP with bit N.0 of constant, and so on. The result can be found into register OP.

Value of variable and OP gated with OR (OP=OP O variable)

Statement

O L[variable], or

O LFInnn

gates the value of variable and the contents of OP with OR in binary form.

If 3 digits are entered after the address of the variable (4 digits after address F) reference is made to the variable in bit operation and only bit No. 0 of register OP participates in the statement.

If 2 digits are entered after the address of the variable (3 digits after address F) reference is made to the word-variable In this case the statement is executed for each bit: bit No. 0 of OP with bit

No. 0 of data, and so on. The result can be found into register OP. For syntax reasons the identity of variable must be substituted for the expression “loading value of variable into register OP” in the statement. This is formally the application of prefix L. Reference can be made to all the variables, the value of which can be stored into OP:
OLIpq, OLYpq, OLVpq, OLPpq, OLFpqr, OLRHipq, OLQnn, OLTnn, OLHnn, OLMn, OLRP0nn, OLFInnn.

Complemented value of variable and OP gated with OR (OP=OP O Nvariable)

Statement

O NL[variable]

O NLFInnn

complements the value of variable (without changing the contents of the variable) and gates the result and the contents of OP with OR in binary form in the above mentioned way. The result of statement O can be found into register OP. For syntax reasons the identity of variable must be substituted for the expression “loading value of variable into register OP” in the statement. This is formally the application of prefix L. Reference can be made to all the variables, the value of which can be stored into OP:
ONLIpq, ONLYpq, ONLVpq, ONLPpq, ONLFpqr, ONLRHipq, ONLQnn, ONLTnn, ONLHnn, ONLMn, ONLRP0nn, ONLFInnn.

O: logical OR result of statement in variable

OP and value of variable gated with OR(variable=variable O OP)

Statement

O S[variable], or

O SFInnn

gates the contents of OP and the value of the variable gated with OR.
If 3 digits are entered after the address of the variable (4 digits after address F) reference is made to the flag and only bit No. 0 of register OP participates in the statement.
If 2 digits are entered after the address of the variable (3 digits after address F) reference is made to the word-variable In this case the statement is executed for each bit: bit No. 0 of OP with bit No. 0 of the value of variable, and so on. The result can be found in the variable. For syntax reasons the identity of variable must be substituted for the expression “storing value of variable into register OP” in the statement. This is formally the application of prefix S. Reference can be made to all variables with statement S:
OSYpq, OSOpq, OSFpqr, OSRHipq, OSQnn, OSTnn, OSHnn, OSMn, OSFInnn.

OP and complemented value of variable gated with OR (variable=Nvariable O OP)

Statement

O NS[variable]

O NSFInnn

complements the value of variable (without changing the contents of the variable) and gates the contents of OP and the result with OR in binary form in the above mentioned way. The result can be found in the variable. For syntax reasons the identity of variable must be substituted for the expression “storing value of variable into register OP” in the statement. This is formally the application of prefix S. Reference can be made to all variables with statement S:
ONSYpq, ONSOpq, ONSFpqr, ONSRHipq, ONSQnn, ONSTnn, ONSHnn, ONSMn, ONSFInnn.

X: Logical eXclusive or, result of statement into register OP

Constant or value of variable and the contents of register OP can be gated with EXCLUSIVE OR:

Decimal constant and OP gated with EXCLUSIVE OR (OP=OP X decimal constant)

Statement

X nnnnn (nnnnn=0...65535)

gates decimal constant nnnnn and the contents of OP with EXCLUSIVE OR. The statement is executed for each bit: bit No. 0 of OP with bit N.0 of constant, and so on. The result can be found into register OP.

Hexadecimal constant and OP gated with EXCLUSIVE OR (OP=OP X hexadecimal constant)

Statement

X .nnnn (.nnnn=0000h...FFFFh)

gates hexadecimal constant .nnnn and the contents of OP with EXCLUSIVE OR. The statement is executed for each bit: bit No. 0 of OP with bit N.0 of constant, and so on. The result can be found into register OP.

Value of variable and OP gated with EXCLUSIVE OR (OP=OP X variable)

Statement

X L[variable], or

X LFInnn

gates the value of variable and the contents of OP with EXCLUSIVE OR.

If 3 digits are entered after the address of the variable (4 digits after address F) reference is made to the flag and only bit No. 0 of register OP participates in the statement.

If 2 digits are entered after the address of the variable (3 digits after address F) reference is made to the work-variable In this case the statement is executed for each bit: bit No. 0 of OP with bit No. 0 of the value of variable, and so on. The result can be found into register OP. For syntax reasons the identity of variable must be substituted for the expression “loading value of variable into register OP” in the statement. This is formally the application of prefix L. Reference can be made to all the variables, the value of which can be stored into OP:

XLIpq, XLYpq, XLVpq, XLPpq, XLFpqr, XLRHipq, XLQnn, XLTnn, XLHnn, XLMn, XLRP0nn, XLFInnn.

Complemented value of variable and OP gated with EXCLUSIVE OR (OP=OP X Nvariable)

Statement

X NL[variable]

X NLFInnn

complements the value of variable (without changing the contents of the variable) and gates the result and the contents of OP with EXCLUSIVE OR in the above mentioned way. The result can be found into register OP. For syntax reasons the identity of variable must be substituted for the expression “loading value of variable into register OP” in the statement. This is formally the application of prefix L. Reference can be made to all the variables, the value of which can be stored into OP:

XNLIpq, XNLYpq, XNLVpq, XNLPpq, XNLFpqr, XNLRHipq, XNLQnn, XNLTnn, XNLHnn, XNLMn, XNLRP0nn, XNLFInnn.

X: logical eXclusive or, result of statement in variable

OP and value of variable gated with EXCLUSIVE OR (variable=variable X OP)

Statement

X S[variable], or

X SFInnn

gates the contents of OP and the value of variable with EXCLUSIVE OR.

If 3 digits are entered after the address of the variable (4 digits after address F) reference is made to the flag and only bit No. 0 of register OP participates in the statement.

If 2 digits are entered after the address of the variable (3 digits after address F) reference is made to the word-variable In this case the statement is executed for each bit: bit No 0 of OP with bit No. 0 of the value of variable, and so on. The result can be found in the variable. For syntax reasons the identity of variable must be substituted for the expression "storing value of variable into register OP" in the statement. This is formally the application of prefix S. Reference can be made to all variables with statement S:

XSYpq, XSXpq, XSFpqr, XSRHipq, XSQnn, XSTnn, XSHnn, XSMn, XSFInnn.

OP and complemented value of variable gated with EXCLUSIVE OR (variable=Nvariable X OP)

Statement

X NS[variable]

X NSFInnn

complements the value of variable (without changing the contents of the variable) and gates the contents of OP and the result with EXCLUSIVE OR in the above mentioned way. The result can be found in the variable. For syntax reasons the identity of variable must be substituted for the expression "storing value of variable into register OP" in the statement. This is formally the application of prefix S. Reference can be made to all variables with statement S:

XNSYpq, XNSXpq, XNSFpqr, XNSRHipq, XNSQnn, XNSTnn, XNSHnn, XNSMn, XNSFInnn.

[...]: parenthesing logic statements executed into register OP

Logic statements executed into register OP can be connected optionally, as e.g.:

```
LI000 A LY022 O LF0012
SY001
```

The execution order of statements goes from left to right. In the above example the contents of OP is 1 if both input line I000 and output line Y022 are set to 1, or the value of F0012 is 1. This OP contents is stored into output line Y001. If this execution order is unsatisfactory, parentheses need to be used.

The maximum nesting depth of parenthesed logic expressions is 8. Calculation of the value of OP is started from the deepest parenthesis:

```
[LI000 A [LY022 O LF0012]]
SY001
```

In the above example first the deepest OR gate is calculated, than the two results are gated with AND and the result is stored into output line Y001.

The above discussed statement are also valid for in word-variables if result of logic statements are into register OP.

Note: in logic statement chain there may also be arithmetic statement.

4.10 Relational Expressions with Register OP

<: is the contents of OP less than...

The condition, that the contents of register OP is less than the constant or the value of variable, can be tested. The condition test regards both the constant and the value of variable as an unsigned number, i.e. considers condition .0 < .FFFF to be true.

Decimal constant (OP < decimal constant)

The first statement of conditional program branch

< *nnnnn* [true branch] *E* [false branch] *Z*
< *nnnnn* [true branch] *Z*
(nnnnn=0...65535)

tests, whether the value of OP is less than constant nnnn (true), or not (false), and the forthcoming conditional program branches are executed on the basis of the result.

Hexadecimal constant (OP < hexadecimal constant)

The first statement of conditional program branch

< *.nnnn* [true branch] *E* [false branch] *Z*
< *.nnnn* [true branch] *Z*
(.nnnn=.0000 ... .FFFF)

tests, whether the value of OP is less than constant .nnnn (true), or not (false), and the forthcoming conditional program branches are executed on the basis of the result.

Value of variable (OP < variable)

The first statement of conditional program branch

< *L*[variable] [true branch] *E* [false branch] *Z*
< *LFInnn* [true branch] *E* [false branch] *Z*
< *L*[variable] [true branch] *Z*
< *LFInnn* [true branch] *Z*

tests, whether the value of OP is less than the value of variable (true), or not (false), and the forthcoming conditional program branches are executed on the basis of the result. For syntax reasons the identity of variable must be substituted for the expression "loading value of variable into register OP" in the statement. This is formally the application of prefix L. Reference is made to all variables, the value of which can be stored into OP.
LIpq, LYpq, LVpq, LPpq, LFpqr, LRHipq, LQnn, LTnn, LHnn, LMn, LRP0nn, LFInnn.

Complemented value of variable (OP < Nvariable)

The first statement of conditional program branch

< *NL*[variable] [true branch] *E* [false branch] *Z*
< *NLFInnn* [true branch] *E* [false branch] *Z*
< *NL*[variable] [true branch] *Z*
< *NLFInnn* [true branch] *Z*

complements the value of variable (without changing the contents of variable), than compares the result with the contents of OP, whether the value of OP is less than the result (true) or not (false), and the forthcoming conditional program branches are executed on the basis of its result. For syntax reasons the identity of variable must be substituted for the expression "loading value of variable into register OP" in the statement. This is formally the application of prefix L. Reference is made to all variables, the value of which can be stored into OP:
NLIpq, NLYpq, NLVpq, NLPpq, NLFpqr, NLRHipq, NLQnn, NLTnn, NLHnn, NLMn, NLRP0nn, NLFInnn.

>: is the contents of OP greater than...

The condition, that the contents of register OP is greater than the constant or the value of variable, can be tested. The condition test regards both the constant and the variable as an unsigned number, i.e. considers condition .0 > .FFFF to be true.

Decimal constant (OP > decimal constant)

The first statement of conditional program branch

> *nnnnn* [true branch] *E* [false branch] *Z*

> *nnnnn* [true branch] *Z*

(nnnnn=0...65535)

tests, whether the contents of OP is greater than decimal constant nnnnn, or not, and the forthcoming conditional program branches are executed on the basis of the result.

Hexadecimal constant (OP > hexadecimal constant)

The first statement of conditional program branch

> *.nnnn* [true branch] *E* [false branch] *Z*

> *.nnnn* [true branch] *Z*

(.nnnn=.0000 ... .FFFF)

tests, whether the contents of OP is greater than constant .nnnn, or not, and the forthcoming conditional program branches are executed on the basis of the result.

Value of variable (OP > variable)

The first statement of conditional program branch

> *L*[variable] [true branch] *E* [false branch] *Z*

> *LFInnn* [true branch] *E* [false branch] *Z*

> *L*[variable] [true branch] *Z*

> *LFInnn* [true branch] *Z*

tests, whether the contents of OP is greater than the value of variable, or not, and the forthcoming conditional program branches are executed on the basis of the result. For syntax reasons the identity of variable must be substituted for the expression “loading value of variable into register OP” in the statement. This is formally the application of prefix L. Reference is made to all variables, the value of which can be stored into OP:

LIpq, LYpq, LVpq, LPpq, LFpqr, LRHipq, LQnn, LTnn, LHnn, LMn, LRP0nn, LFInnn.

Complemented value of variable (OP > Nvariable)

The first statement of conditional program branch

> *NL*[variable] [true branch] *E* [false branch] *Z*

> *NLFInnn* [true branch] *E* [false branch] *Z*

> *NL*[variable] [true branch] *Z*

> *NLFInnn* [true branch] *Z*

complements the value of variable (without changing the contents of variable), than compares the result with the contents of OP, whether the value of OP is greater than the result, or not, and the forthcoming conditional program branches are executed on the basis of its result. For syntax reasons the identity of variable must be substituted for the expression “loading value of variable into register OP” in the statement. This is formally the application of prefix L. Reference is made to all variables, the value of which can be stored into OP:

NLIpq, NLYpq, NLVpq, NLPpq, NLFpqr, NLRHipq, NLQnn, NLTnn, NLHnn, NLMn, NLRP0nn, NLFInnn.

=: is the contents of OP equal to...

The condition, that the contents of register OP is equal to the constant or the value of variable, can be tested. The condition test regards both the constant and the value of variable as an unsigned number, i.e. considers condition .0 = .FFFF to be true.

Decimal constant (OP = decimal constant)

The first statement of conditional program branch

= *nnnnn* [true branch] *E* [false branch] *Z*

= *nnnnn* [true branch] *Z*

(nnnnn=0...65535)

tests, whether the contents of OP is equal to constant nnnnn, or not, and the forthcoming conditional program branches are executed on the basis of the result.

Hexadecimal constant (OP = hexadecimal constant)

The first statement of conditional program branch

= *.nnnn* [true branch] *E* [false branch] *Z*

= *.nnnn* [true branch] *Z*

(.nnnn=.0000 ... .FFFF)

tests, whether the contents of OP is equal to constant .nnnn, or not, and the forthcoming conditional program branches are executed on the basis of the result.

Value of variable (OP = variable)

The first statement of conditional program branch

= *L*[variable] [true branch] *E* [false branch] *Z*

= *LFInnn* [true branch] *E* [false branch] *Z*

= *L*[variable] [true branch] *Z*

= *LFInnn* [true branch] *Z*

tests, whether the contents of OP is equal to the value of variable, or not, and the forthcoming conditional program branches are executed on the basis of the result. For syntax reasons the identity of variable must be substituted for the expression “loading value of variable into register OP” in the statement. This is formally the application of prefix L. Reference is made to all variables, the value of which can be stored into OP:

LIpq, LYpq, LVpq, LPpq, LFpqr, LRHipq, LQnn, LTnn, LHnn, LMn, LRP0nn, LFInnn.

Complemented value of variable (OP = Nvariable)

The first statement of conditional program branch

= *NL*[variable] [true branch] *E* [false branch] *Z*

= *NLFInnn* [true branch] *E* [false branch] *Z*

= *NL*[variable] [true branch] *Z*

= *NLFInnn* [true branch] *Z*

complements the value of variable (without changing the contents of variable), than compares the result with the contents of OP, whether the value of OP is equal to the result, or not, and the forthcoming conditional program branches are executed on the basis of the result. For syntax reasons the identity of variable must be substituted for the expression “loading value of variable into register OP” in the statement. This is formally the application of prefix L. Reference is made to all variables, the value of which can be stored into OP:

NLIpq, NLYpq, NLVpq, NLPpq, NLFpqr, NLRHipq, NLQnn, NLTnn, NLHnn, NLMn, NLRP0nn, NLFInnn.

<=: is the contents of OP less than or equal to...

The condition, that the contents of register OP is less than or equal to the constant or the value of variable, can be tested. The condition test regards both the constant and the value of variable as an unsigned number, i.e. considers condition .0 <= .FFFF to be true.

Decimal number (OP <= decimal number)

The first statement of conditional program branch

<= *nnnnn* [true branch] *E* [false branch] *Z*

<= *nnnnn* [true branch] *Z*

(nnnnn<=0...65535)

tests, whether the contents of OP is less than or equal to decimal constant nnnnn, or not, and the forthcoming conditional program branches are executed on the basis of the result.

Hexadecimal number (OP <= hexadecimal number)

The first statement of conditional program branch

<= *.nnnn* [true branch] *E* [false branch] *Z*

<= *.nnnn* [true branch] *Z*

(.nnnn<=.0000 ... .FFFF)

tests, whether the contents of OP is less than or equal to constant .nnnn, or not, and the forthcoming conditional program branches are executed on the basis of the result.

Value of variable (OP <= variable)

The first statement of conditional program branch

<= *L*[variable] [true branch] *E* [false branch] *Z*

<= *LFInnn* [true branch] *E* [false branch] *Z*

<= *L*[variable] [true branch] *Z*

<= *LFInnn* [true branch] *Z*

tests, whether the contents of OP is less than or equal to the value of variable, or not, and the forthcoming conditional program branches are executed on the basis of the result. For syntax reasons the identity of variable must be substituted for the expression “loading value of variable into register OP” in the statement. This is formally the application of prefix L. Reference is made to all variables, the value of which can be stored into OP:

LIpq, LYpq, LVpq, LPpq, LFpqr, LRHipq, LQnn, LTnn, LHnn, LMn, LRP0nn, LFInnn.

Complemented value of variable (OP <= Nvariable)

The first statement of conditional program branch

<= *NL*[variable] [true branch] *E* [false branch] *Z*

<= *NLFInnn* [true branch] *E* [false branch] *Z*

<= *NL*[variable] [true branch] *Z*

<= *NLFInnn* [true branch] *Z*

complements the value of variable (without changing the contents of variable), than compares the result with the contents of OP, whether the value of OP is less than or equal to the result, or not, and the forthcoming conditional program branches are executed on the basis of the result. For syntax reasons the identity of variable must be substituted for the expression “loading value of variable into register OP” in the statement. This is formally the application of prefix L. Reference is made to all variables, the value of which can be stored into OP:

NLIpq, NLYpq, NLVpq, NLPpq, NLFpqr, NLRHipq, NLQnn, NLTnn, NLHnn, NLMn, NLRP0nn, NLFInnn.

>=: is the contents of OP greater than or equal to...

The condition, that the contents of register OP is greater than or equal to the constant or the value of variable, can be tested. The condition test regards both the constant and the value of variable as an unsigned number, i.e. considers condition .0 >= .FFFF to be true.

Decimal number (OP >= decimal number)

The first statement of conditional program branch

>= *nnnnn* [true branch] *E* [false branch] *Z*

>= *nnnnn* [true branch] *Z*

(nnnnn>=0...65535)

tests, whether the contents of OP is greater than or equal to decimal constant nnnnn, or not, and the forthcoming conditional program branches are executed on the basis of the result.

Hexadecimal number (OP >= hexadecimal number)

The first statement of conditional program branch

>= *.nnnn* [true branch] *E* [false branch] *Z*

>= *.nnnn* [true branch] *Z*

(.nnnn>=.0000 ... .FFFF)

tests, whether the contents of OP is greater than or equal to constant .nnnn or not, and the forthcoming conditional program branches are executed on the basis of the result.

Value of variable (OP >= variable)

The first statement of conditional program branch

>= *L*[variable] [true branch] *E* [false branch] *Z*

>= *LFInnn* [true branch] *E* [false branch] *Z*

>= *L*[variable] [true branch] *Z*

>= *LFInnn* [true branch] *Z*

tests, whether the contents of OP is greater than or equal to the value of variable, or not, and the forthcoming conditional program branches are executed on the basis of the result. For syntax reasons the identity of variable must be substituted for the expression “loading value of variable into register OP” in the statement. This is formally the application of prefix L. Reference is made to all variables, the value of which can be stored into OP:

LIpq, LYpq, LVpq, LPpq, LFpqr, LRHipq, LQnn, LTnn, LHnn, LMn, LRP0nn, LFInnn.

Complemented value of variable (OP >= Nvariable)

The first statement of conditional program branch

>= *NL*[variable] [true branch] *E* [false branch] *Z*

>= *NLFInnn* [true branch] *E* [false branch] *Z*

>= *NL*[variable] [true branch] *Z*

>= *NLFInnn* [true branch] *Z*

complements the value of variable (without changing the contents of variable), than compares the result with the contents of OP whether the value of OP is greater than or equal to the result, or not, and the forthcoming conditional program branches are executed on the basis of the result. For syntax reasons the identity of variable must be substituted for the expression “loading value of variable into register OP” in the statement. This is formally the application of prefix L. Reference is made to all variables, the value of which can be stored into OP:

NLIpq, NLYpq, NLVpq, NLPpq, NLFpqr, NLRHipq, NLQnn, NLTnn, NLHnn, NLMn, NLRP0nn, NLFInnn.

4.11 Goto Statements

:nnn: label

Labels can be written in the PLC program. After goto statements the execution of program is always continued from the specified label. The subroutines in the PLC program can be identified with labels. Also the three main modules of the PLC program (:000, :001 and :002) are identified with labels.

The address of label is ":". 3-decimal-digit identity number nnn follows the address. The value range of the identity number:

000-200.

Th following labels are reserved, i.e. their use is standard:

:000 module 0
:001 module 1
:002 module 2
:197 module of softkey captions of PLC action menu
:198 module of message strings
:199 module of error message strings
:200 information module of PLC program

Other labels are freely available.

J0, J1, J2: closing statements of modules

Statement **J0** indicates the end of and closes module :000.

As the effect of statement J0 the PLC returns the control to the NC. In the next time slice after module :001 has been executed the execution of module :000 is started from the beginning of the module by the use of statement J1.

Statement **J1** indicates the end of and closes module :001.

As the effect of statement J1 the control is transferred to module :000. The execution of module :000 is continued, where it was interrupted in the previous time slice, except if statement J0 has been reached in the preceding time slice. In this case the execution of module :000 is started from its beginning. If the execution of module :001 or :002 is not finished within its time slice emergency state is generated by the control by means of error message PLC TIMEOUT1 or PLC TIMEOUT2 and loses signal NC READY. The error is fatal, can only be canceled by turning the machine off.

The use of both statements is obligatory at the end of he appropriate module.

Statement **J2** indicates the end of and closes module :002.

$: closing message modules

Modules :197, :198, :199, :200 must be closed with character $.

Gnnn: direct goto statement

As the effect of this statement the control is transferred to label :nnn of PLC program without condition. The program execution is continued from here.

The usable values nnn: 0, 3-196

GFnnn: indirect goto statement

As the effect of this statement the control is transferred to label :nnn of PLC program without condition to the label of the PLC program, the code number of which can be found at local variable Fnnn. The program execution is continued from here.

The value range of variable Fnnn: 3-196

Flags to be set:
F0080: syntax error the value of variable Fnnn is not in value range 3-196.
F0082: the value of variable Fnnn is not decimal.

Cnnn: direct subroutine call
As the effect of this statement the control is transferred to subroutine :nnn without condition. As the effect of the first statement R, which is found by the program in the course of execution the statement following statement Cnnn is returned.
The value range of identity number of label: 3-196

CFnnn: indirect subroutine call
As the effect of this statement the control is transferred to the subroutine, the identity number of which is the contents of variable Fnnn. As the effect of the first statement R, which is found by the program in the course of execution the statement following statement Cnnn is returned.
The value range of data found at address nnn: 3-196
Flags to be set:
F0080: syntax error: the value of variable of Fnnn is not in value range 3-196.
F0082: the value of variable Fnnn is not decimal.

R: return from subroutine
As the effect of statement R the program execution is continued from the statement following the last subroutine call statement (Cnnn, CFnnn) before reaching statement R. It is usable only in the valid label subroutine :003...:196.

4.12 Use of Up/Down Counters

UQnn: incrementing the contents of the nnth up/down counter
Statement

UQnn

increases the contents of the nnth up/down counter by one. If the contents of the counter is 65535 by means of statement UQnn it becomes 0.

DQnn: decrementing the contents of the nnth up/down counter
Statement

DQnn

decreases the contents of the nnth up/down counter by one. If the contents of the counter is 0 by means of statement DQnn it becomes 65535.

Qnn: state test of the nnth up/down counter
The following condition tests can be initiated on the state of the nnth up/down counter:

Qnn [Qnn ≠ 0] *E* [Qnn = 0] *Z*
Qnn [Qnn ≠ 0] *Z*

Complemented test of the contents of the counter is also possible:

NQnn [Qnn = 0] *E* [Qnn ≠ 0] *Z*
NQnn[Qnn = 0] *Z*

4.13 Condition Test on Timers

Tnn: condition test on the state of the nnth 20msec timer

Condition test can be initiated on the state of the nnth 20-msec 16-bit timer. There are two results of the test of the condition: true if the timer is running, false if the timer is terminated.

Tnn [running: Tnn>0] *E* [terminated: Tnn=0] *Z*
Tnn [running: Tnn>0] *Z*

Negated call of the timer is also possible:

NTnn [terminated: Tnn=0] *E* [running: Tnn>0] *Z*
NTnn [terminated: Tnn=0] *Z*

Running of timer is worked by the NC program.

Hnn: condition test on the state of the nnth second timer

Condition test can be initiated on the state of the nnth 1-sec 16-bit timer. There are two results of the condition test: true if the timer is running, false if the timer is terminated.

Hnn [running: Hnn>0] *E* [terminated: Hnn=0] *Z*
Hnn [running: Hnn>0] *Z*

Negated call of the timer is also possible:

NHnn [terminated: Hnn=0] *E* [running: Hnn>0] *Z*
NHnn [terminated: Hnn=0] *Z*

Running of the timer is worked by the NC program.

Mn: condition test on the state of the nnth minute timer

Condition test can be initiated on the state of the nnth minute 16-bit timer. There are two results of the condition test: true if the timer is running, false if the timer is terminated.

Mn [running: Mn>0] *E* [terminated: Mn=0] *Z*
Mn [running: Mn>0] *Z*

Negated call of the timer is also possible:

NMn [terminated: Mn=0] *E* [running: Mn>0] *Z*
NMn [terminated: Mn=0] *Z*

Running of the timer is worked by the NC program.

4.14 Search Statements

HFnnn: Search for the Contents of OP in Tables

This statement searches for the contents of register OP in the indicated table, which can be found in the PLC local area. After the statement name (HF) the address of the local variable, where the registers controlling the statement begin must be entered with three decimal digits. The parameter area of the statement is 10 bytes. The parameter area of the statement must be placed in the freely available working area.

Description of the statement:

nnn: address of a local variable, where the parameter area used in the statement starts.

Address of registers	Meaning of registers
nnn	Format register
nnn+2	Start address of table
nnn+4	Length of table
nnn+6	Mask register
nnn+8	Address of found data

Format register

The format register can be found at address nnn of the parameter area. In this register the number of bytes, into which the searched item is stored can be given.

Length of register: 1 word

Possible contents of register: 1, 2.

If a byte is searched for, the searched data must be placed in the lower byte of OP.

Start address of table

The start address of the defined table must be entered at address nnn+2 of the parameter area. The value of start address must be given in decimal form.

Length of table

The length of the indicated table must be entered in two bytes, at address nnn+4 of the parameter area. The length is specified in byte units. If for example the table is in the area of F300-F349 the value to be written into register is 50. The length of table must be entered in binary form.

Mask register

It is found at address nnn+6. The search statement compares the contents of OP to the items of table according to the following relation:

$$OP=TABLE(i^{th}\text{ item}) \text{ AND } MASK$$

The i^{th} item of the table and the MASK register are gated with AND, the result is compared to the contents of OP.

Address of found data

If in the course of search the searched item is found in the table the address of data is written in this register. The address of the found item is put in this register in decimal form.

After executing the statement the following flags can be tested

F0080: syntax error: the start address of table is not decimal
The lower byte of format register is not 1 or 2, or the address values are not in range 000...999.

F0081: Data not found. If the searched data is not found in the defined table flag F0081 is set to 1, else it is set to 0.

Sample for the use of statement HFnnn:

```
.0002       ;format of search is in word operation
SF120       ;storing into format register
.0500       ;start address of tool pot table
SF122       ;storing into start address
LRP039      ;length of magazine: number of tool pots
*2          ;transforming to byte number,
            ;because items of tool pot table are words
+2          ;adding tool pot No. 0: length of table
SF124       ;entering length
.3FFF       ;mask: width code (14th, 15th bit)is cut off tool pot :table
            data
```

```
      SF126       ;entering mask
      LF024       ;code of called tool is loaded into OP
      HF120       ;searching for address of called tool in table
   F0080          ;if syntax error in search
      U735        ;SEARCH ERROR WITH H error message strobe on,
   E              ;otherwise no syntax error
    F0081         ;if data not found: MANUAL REPLACEMENT
                  ;description of manual replacement actions
    E             ;if data found
                  ;description of auto replacement actions
      LF128       ;address of tool is loaded into OP
      BIN         ;converting to binary form
      -500        ;subtracting start address of tool pot table
      /2          ;creating item number (word)
      SF104       ;position of found tool in magazine
                  ;
    Z             ;end of condition data not found
   Z              ;end of condition search error
```

PFnnn: search for free pot with the appropriate width in tool pot table

This statement searches in the tool pot table for free tool position of the specified width code into register OP by starting from the specified item of table in one direction (if magazine has only one direction), or two directions (if magazine can be rotated in two directions).

The statement can be used in case of random access magazine handle, when tools reserving more tool pots can also be positioned in the magazine, and the method mentioned in case of tool pot table can be used for coding width. In this case the returning tool cannot be placed into the pot, in which the replacement is to be done if the width code of tool in spindle and returning tool is not the same.

The statement first examines, whether the width code into OP (width of returning tool) equals to the width code of the pot. If yes, this pot number is defined for the returning tool. If their width code differ the above statement searches for the nearest free tool position, the width code of which equals to the returning tool in only positive direction or in both directions.

After the statement name (PF) the address of the local variable, where the registers controlling the statement begin must be entered with three decimal digits. The parameter area of the statement is 6 bytes.

The parameter area must be placed in the freely available working area. The form of register OP must be as follows:

```
      1111 11
      5432 1098 7654 3210
[OP]  [xxxx|xxxx|xxxx|xxxx]
       ||└──────────────┴──── x: do not care (position of returning tool)
       └┴──────────────────── width code of returning tool
```

Description of the statement:

nnn: address of a local variable, where the parameter area used in the statement starts.

Address of registers	Meaning of registers
nnn	Format register
nnn+2	Address of tool pot table, from where the search is started = (number of the pot, in which the replacement is to be done)*2+500
nnn+4	address of found item

Format register:
The format register can be found at address nnn of the parameter area. Both the lower and upper bytes of the register are used.
Length of register: 1 word
The contents of nnnth byte is always 2 (word).
Byte nnn+1 0: search only in positive direction
1: search in both directions

Address of tool pot table, from where the search is started
It can be found at address nnn+2. The search is started from the address of tool pot table, which corresponds to the contents of address nnn+2. The address can be calculated from the number of the pot, in which the replacement is to be done with the help of the following relation:

(number of the pot, in which the replacement is to be done)*2+500

In the format register the nearest free tool position with the appropriate width code is searched for in both directions or only in one direction as a function of the magazine. If in the course of searching the maximum position has been reached in positive direction the search is continued from position No. 1, while if the minimum is reached in negative direction it is continued from the maximum position (Specified at parameter MAGAZINE).
The address, from where the search is started must always be entered in decimal form.

Address of found item
If in the course of search the position with the appropriate width code is found the address of the free position is written into this register in decimal form. The returning tool is to be placed in this pot.
The number of the found free pot corresponds to the number of the pot, in which the replacement is to be done, if the width code of that pot corresponds to that of the returning tool.
In the course of search the contents of OP and the contents of the table is compared according to the following relation:

(OP AND C000h) = TABLE(ith item)

After executing the statement the following status flags can be tested
F0080: syntax error: the start address of table is not decimal
The lower byte of format register is not 2, its upper byte is not 0 or 1, or the address values are not in range 000...999.
F0081: Data not found. If the searched data is not found in the selected table flag F0081 is set to 1, else it is set to 0.

Sample for the use of statement PFnnn:

```
      .0102       ;searching for data in word item in both directions
      SF130       ;storing into format
      LF110       ;current magazine position (opposing spindle) into OP
      *2          ;transforming into byte
      +500        ;adding start address of tool pot table
      BCD         ;converting to BCD form for search
      SF132       ;search for free position is started from this address
      LF500       ;number and width code of tool in spindle into OP
      PF130       ;searching for free pot for tool with the above width
    F0080         ;if syntax error in search
      U736        ;SEARCH ERROR WITH P error message strobe on,
    E             ;else if no syntax error
     F0081        ;if data not found
      U737        ;NO FREE POSITION error message strobe on
     E            ;data found
      LF134       ;number of found pot into OP
      BIN         ;converting to binary form
      -500        ;subtracting start address of tool pot table
```

```
      /2           ;creating item number (word)
      SF108        ;position of returning tool in magazine
     Z             ;end of condition data not found
    Z              ;end of search error
```

4.15 Reading and writing the memory of NC

MRnnn: reading the NC memory

This statement is for reading the NC memory. Memory areas reachable for the PLC: macro variables and parameters. After the statement name (MR) the start address of the register area controlling the statement must be entered with three decimal digits. The register area of the statement is 8 bytes.

Description of the statement:

nnn: start address of the local area containing the registers used in the statement.

Address of registers	Meaning of registers
nnn	Format register
nnn+2	Segment register
nnn+4	Index register
nnn+6	Start address of the data to be loaded

<u>Format register:</u>

The format register can be found at address nnn of the register area. Length of register: 1 word. In the lower byte of the register the size of the allocated area in the bytes, into which the data to be loaded is stored into the local area, can be given.

Possible contents of byte nnn: 1, 2, or 4.

If a flag is loaded from the parameter area and 2 bytes are reserved for it the flag is in bit No. 0 of the lower byte. Remember, that in case of filling a register if byte data is read bytes must be reserved for it, if word data is read a word must be reserved for it, and so on. In case of reading flags the reserved byte number is of no importance.

The upper byte of the register is only used when loading macro variables #1...#999. These variables are in floating point format in the NC memory but in PLC programs there are only integer variables. Therefore the value of the parameter must be transferred as an integer whereby the decimal point is shifted by the number of possible places after the point (shift count).

possible contents of address nnn+1, i.e. shift count: 0,1,...,8

E.g.: if the value at address nnn is 4, the value in variable #100 is 1 and
the value of shift count is 3, then 1,000 can be entered with three decimal places.
The resulting integer is 1000.
If however the value of shift count is 0, the resulting integer is 1.

<u>Segment register:</u>

In this register the segment of the NC memory, to which the loading statement is referred to must be specified.

Possible values of address *nnn+2*:

=1 macro variables
=2 parameters

Index register:
The index register contains the reference number to be loaded within the indicated NC memory segment.
When loading

Macro variables

it is the reference number of the macro variable (the number after signal #).

Possible values of address *nnn+4*:

1...999
2000...
The loading of macro variables #1000... #1999 is not possible.
When loading

NC parameters

it is the reference number of the parameter.
The contents of index register is always a BCD number

Start address of the area allocated for the data
The start address of local area, into where the data is loaded can be found at address nnn+6. Bytes with lower local value are loaded into lower addresses, while those with higher local value are loaded into higher addresses. The data written here is regarded by the compiler as a decimal number, similarly to number nnn in statement LFnnn or SFnnn.
Start address of the data to be written is always a BCD number.

After executing the NC memory the reading of the state of the following status flags can be tested:
F0080: Syntax error in statement
If the registers used for the statement are filled out correctly::

- the lower byte of format register is 1, 2, or 4, and the allocated area corresponds to the size of data to be read,
- the shift count in case of parameter is within value range 0...8,
- both segment and index registers refer to readable NC memory area,
- the address register refers to the address range of freely available local variables.

Else flag F0080 is set to 1.
F0082: not BCD number
The flag is set to 1 if the value of index or address register is not in BCD form.

Sample for reading macro variable #180 into the PLC:

```
Location:
F200...F206 -     registers of statement MR200
F270...F273 -     data loaded from #180

      .0304       ;number of decimal digits =3, format =4 (4 byte)
      SF200       ;storing into format register
      .0001       ;index of macro variables
      SF202       ;storing into segment register
      .0180       ;line number of macro variable #180
      SF204       ;storing into index register
      .0270       ;load data at address F270...F273
      SF206       ;storing into address register
      MR200       ;loading macro variable
(F0080            ;if syntax error
OF0082)           ;or addresses are not in BCD form
      U720        ;MACRO READING ERROR message strobe on
Z                 ;end of condition
                  ;syntax error
```

MWnnn: overwriting data in the NC memory

This statement is for overwriting data in the NC memory. Memory areas reachable for the PLC : macro variables and parameters. After the statement name (MR) the start address of the register area controlling the statement must be entered with three decimal digits. The register area of the statement is 8 bytes. The register and data areas must be placed in the freely available working area.

Description of the statement:

nnn: start address of the local area containing the registers area used in the statement.

Address of registers	Meaning of registers
nnn	Format register
nnn+2	Segment register
nnn+4	Index register
nnn+6	Start address of the data area to be stored

Format register:

The format register can be found at address nnn of the register area. Length of register: 1 word. In the lower byte of the register the size of the transferred data to be stored is stored among the common variables, can be given.

possible contents of byte Fnnn: 1, 2, or 4.

If a flag is transferred from the data and 2 bytes are reserved for it the flag must be placed into bit No. 0 of the lower byte. Remember, that in case of filling a register if byte data is transferred bytes must be reserved for it, if word data is transferred a word must be reserved for it, and so on. In case of flags the reserved byte number is of no importance.

The upper byte of the register is only used when overwriting macro variables #1...#999. These variables are in floating point format in the NC memory but in PLC programs there are only integer variables. Therefore the value of the data must be transferred as an integer whereby the decimal point is shifted by the number of possible places after the point (shift count).

possible value of address nnn+1, i.e. shift count: 0,1,...,8

E.g.: if the value at address nnn is 4, the value of the data is 1000 and
the shift count is 3, then #100=1,
in case the shift count is 0, #100=1000..

Segment register:

In this register the segment of the NC memory, to which the overwriting statement is referred to must be specified.

Possible values of address *nnn+2*:

=1 macro variables

=2 NC parameters

Index register:

The index register contains the reference number to be stored within the selected memory segment.

When overwriting

Macro variables
it is the reference number of the macro variable (the number after signal #).
Possible values of address *nnn+4*:
1...999
2000...
The overwriting of macro variables #1000... #1999 is not possible.
When overwriting
NC parameters
it is the reference number of the parameter.
The contents of index register is always a BCD number

Start address of the area allocated for the data
The start address of the local area, into where the data is stored can be found at address nnn+6. Bytes with lower local value are stored into lower addresses, while those with higher local value are stored into higher addresses. The data written here is regarded by the compiler as a decimal number, similarly to number nnn in statement LFnnn or SFnnn.
Start address of the data to be stored is always a BCD number.

After executing the NC memory the overwriting of the state of the following status flags can be tested:
F0080: Syntax error in statement
If the registers used for the statement are filled out correctly:
- the lower byte of format register is 1, 2, or 4, and the location corresponds to the size of data to be stored,
- the upper byte is within value range 0...8,
- both segment and index registers refer to writeable memory area,
- the address register refers to the address range of freely available local variables.

Else flag F0080 is set to 1.
F0082: not BCD number
The flag is set to 1 if the value of index or start address register is not in BCD form.

Sample for storing macro variable #180 into the PLC:

```
Location:
F210...F216 -      parameters of statement MR210
F298...F301 -      data overwritten into #183

      .0304        ;number of decimal digits =3, format =4 (4 byte)
      SF210        ;storing into format register
      .0001        ;index of macro variables
      SF212        ;storing into segment register
      .0183        ;reference number of macro variable #183
      SF214        ;storing into index register
      .0298        ;load data from address F270...F273
      SF216        ;storing into address register
      MR210        ;overwriting macro variable
(F0080             ;if syntax error
OF0082)            ;or addresses are not in BCD form
      U721         ;MACRO WRITING ERROR message strobe on
Z                  ;end of condition
                   ;syntax error
```

4.16 Arithmetic Operations

Beside the 16-bit unsigned arithmetic operations executed into register OP arithmetic operations with 1, 2 or 4 byte numbers or signed numbers are also available.

ADDnnn: addition: A + B = C

This statement is for adding 1, 2, or 4 byte numbers, signed numbers, or the two's complement of the numbers. After the statement name (ADD) the start address of the register area controlling the statement must be entered with three decimal digits. The register area of the statement is 8 bytes. The register and data areas must be placed in the freely available working area.
Description of the statement:
nnn: start address of the local area containing the registers used in the statement..

Address of registers	Meaning of registers
nnn	Format register
nnn+2	Start address of 1st addable (A)
nnn+4	Start address of 2nd addable (B)
nnn+6	Start address of sum (C)

Format register:
The format register can be found at address nnn of the register area. In this register the number of bytes, in which the numbers of statement are reserved can be given.
Length of register: 1 word
Possible contents of register: 1, 2, or 4.

Start address of 1st addable (A):
The start address of the 1st addable can be found at address nnn+2 of the register area. This address must point to the local variable, at which the value of 1st addable can be found. At this address the number of bytes specified at format register is taken into account during the addition in order to calculate the result. Bytes with lower local value are at the lower addresses, while those with higher local values are at higher addresses.
Start address of 1st addable is always a BCD number.

Start address of 2nd addable (B):
The start address of the 2nd addable can be found at address nnn+4 of the register area. This address must point to the local variable, at which the value of 2nd addable can be found. At this address the number of bytes specified at format register is taken into account during the addition in order to calculate the result. Bytes with lower local value are at the lower addresses, while those with higher local values are at higher addresses.
Start address of 2nd addable is always a BCD number.

Start address of sum (C):
The start address of the sum can be found at address nnn+6 of the register area. This address must point to the local variable, at which the value of the sum can be found. At this address the number of bytes specified at format register is taken into account during the addition in order to calculate the result. Bytes with lower local value are at the lower addresses, while those with higher local values are at higher addresses.
Start address of the sum is always a BCD number.

After the execution of addition the state of the following status flags can be tested:
F0080: Syntax error in statement
If the registers used for the statement are filled out correctly::
- the contents of format register is 1, 2, or 4,
- the address registers refer to the address range of usable local variables.

Else flag F0080 is set to 1.
F0082: not BCD number
The flag is set to 1 if the values of address registers are not in BCD form.
F0046: The result is 0.
F0047: The result is negative
F0053: Overflow
If the result of addition does not have enough room at the bytes, the number of which is specified at format register further bytes are not overwritten, but flag F0053 is set to 1.

Example for the use of statement ADDnnn

```
Location:
F220...F226 -      input registers of statement ADD220
F270...F273 -      1st addable
F274...F277 -      2nd addable
F282...F285 -      sum

      .0004        ;length of numbers =4 (4 bytes)
      SF220        ;storing into addition format register
      .0270        ;start address of 1st addable: F270(...F273)
      SF222        ;storing into 1st addable address register
      .0274        ;start address of 2nd addable: F274(...F277)
      SF224        ;storing into 2nd addable address register
      .0282        ;start address of sum: F282(...F285)
      SF226        ;storing into sum address register
      ADD220       ;addition
(F0080             ;if syntax error
OF0082             ;or addresses are not in BCD form
OF0053)            ;or overflow
      U722         ;ADDITION ERROR message strobe on
Z                  ;end of condition
                   ;syntax error
```

SUBnnn: subtraction: A - B = C

This statement is for subtracting 1, 2, or 4 byte numbers, signed numbers, or the two's complement of the numbers. After the statement name (SUB) he start address of the register area controlling the statement must be entered with three decimal digits. The register area of the statement is 8 bytes. The register and data areas must be placed in the freely available working area.
Description of the statement:
nnn: start address of the local area containing the registers used in the statement..

Address of registers	Meaning of registers
nnn	Format register
nnn+2	Start address of subtractand (A)
nnn+4	Start address of subtractor (B)
nnn+6	Start address of difference (C)

Format register:
The format register can be found at address nnn of the register area. In this register the number of bytes, in which the numbers of statement are shown can be given.
Length of register: 1 word
Possible contents of register: 1, 2, or 4.

Start address of subtractand (A):
The start address of the subtractand can be found at address nnn+2 of the register area. This address must point to the local variable, at which the value of subtractand can be found. At this address the number of bytes specified at format register is taken into account during the subtraction in order to calculate the result. Bytes with lower local value are at the lower addresses, while those with higher local values are at higher addresses.
Start address of subtractand is always a BCD number.

Start address of subtractor (B):
The start address of the subtractor can be found at address nnn+4 of the register area. This address must point to the local variable, at which the value of subtractor can be found. At this address the number of bytes specified at format register is taken into account during the subtraction in order to calculate the result. Bytes with lower local value are at the lower addresses, while those with higher local values are at higher addresses.
Start address of subtractor is always a BCD number.

Start address of difference (C):
The start address of the difference can be found at address nnn+6 of the register area. This address must point to the local variable, at which the value of the difference can be found. At this address the number of bytes specified at format register is taken into account during the subtraction in order to calculate the result. Bytes with lower local value are at the lower addresses, while those with higher local values are at higher addresses.
Start address of the difference is always a BCD number.

After the execution of subtraction the state of the following status flags can be tested:
F0080: Syntax error in statement
If the registers used for the statement are filled out correctly::
- the contents of format register is 1, 2, or 4,
- the address registers refer to the address range of usable local variables.

Else flag F0080 is set to 1.
F0082: not BCD number
The flag is set to 1 if the values of address registers are not in BCD form.
F0046: The result is 0.
F0047: The result is negative
F0053: Overflow

If the result of subtraction does not have enough room at the bytes, the number of which is specified at format register further bytes are not overwritten, but flag F0053 is set to 1.

Example for the use of statement SUBnnn

```
Location:
F230...F236 -     input registers of statement SUB230
F270...F273 -     subtractand
F274...F277 -     subtractor
F286...F289 -     difference

      .0004       ;length of numbers =4 (4 bytes)
      SF230       ;storing into subtraction format register
      .0270       ;start address of subtractand: F270(...F273)
      SF232       ;storing into subtractand address register
      .0274       ;start address of subtractor: F274(...F277)
      SF234       ;storing into subtractor address register
      .0286       ;start address of difference: F286(...F289)
      SF236       ;storing into difference address register
      SUB230      ;subtraction
(F0080            ;if syntax error
OF0082            ;or addresses are not in BCD form
OF0053)           ;or overflow
      U723        ;SUBTRACTION ERROR message strobe on
Z                 ;end of condition
                  ;syntax error
```

MULnnn: multiplication: A * B = C

This statement is for multiplying 1, 2, or 4 byte numbers, signed numbers, or the two's complement of the numbers. After the statement name (MUL) the start address of the register area controlling the statement must be entered with three decimal digits. The register area of the statement is 8 bytes.

Description of the statement:

nnn: start address of a local area containing the registers used in the statement.

Address of registers	Meaning of registers
nnn	Format register
nnn+2	Start address of multiplicand (A)
nnn+4	Start address of multiplicator (B)
nnn+6	Start address of product (C)

Format register:

The format register can be found at address nnn of the register area. In this register the number of bytes, in which the numbers of statement are shown can be given.

Length of register: 1 word

Possible contents of register: 1, 2, or 4.

Start address of multiplicand (A):

The start address of the multiplicand can be found at address nnn+2 of the register area. This address must point to the local variable, at which the value of multiplicand can be found. At this address the number of bytes specified at format register is taken into account during the multiplication in order to calculate the result. Bytes with lower local value are at the lower addresses, while those with higher local values are at higher addresses.

Start address of multiplicand is always a BCD number.

Start address of multiplicator (B):
The start address of the multiplicator can be found at address nnn+4 of the register area. This address must point to the local variable, at which the value of multiplicator can be found. At this address the number of bytes specified at format register is taken into account during the multiplication in order to calculate the result. Bytes with lower local value are at the lower addresses, while those with higher local values are at higher addresses.
Start address of multiplicator is always a BCD number.

Start address of product (C):
The start address of the product can be found at address nnn+6 of the register area. This address must point to the local variable, at which the value of the product can be found. At this address the number of bytes specified at format register is taken into account during the multiplication in order to calculate the result. Bytes with lower local value are at the lower addresses, while those with higher local values are at higher addresses.
Start address of the product is always a BCD number.

After the execution of multiplication the state of the following status flags can be tested:
F0080: Syntax error in statement
If the registers used for the Statement are filled out correctly::
- the contents of format register is 1, 2, or 4,
- the address registers refer to the address range of usable local variables.

Else flag F0080 is set to 1.
F0082: not BCD number
The flag is set to 1 if the values of address registers are not in BCD form.
F0046: The result is 0.
F0047: The result is negative
F0053: Overflow
If the result of multiplication does not have enough room at the bytes, the number of which is specified at format register further bytes are not overwritten, but flag F0053 is set to 1.

Example for the use of statement MULnnn

```
Location:
F240...F246 -      input registers of statement MUL240
F282...F285 -      multiplicand
F278...F281 -      multiplicator
F290...F297 -      product

      .0004        ;length of numbers =4 (4 bytes)
      SF240        ;storing into multiplication format register
      .0282        ;start address of multiplicand: F282(...F285)
      SF242        ;storing into multiplicand address register
      .0278        ;start address of multiplicator: F278(...F281)
      SF244        ;storing into multiplicator address register
      .0290        ;start address of product: F290(...F297)
      SF246        ;storing into product address register
      MUL240       ;multiplication
(F0080             ;if syntax error
OF0082             ;or addresses are not in BCD form
OF0053)            ;or overflow
      U724         ;MULTIPLICATION ERROR message strobe on
Z                  ;end of condition
                   ;syntax error
```

DIVnnn: division: A / B = C

This statement is for dividing 1, 2, or 4 byte numbers, signed numbers, or the two's complement of the numbers. After the statement name (DIV) the start address of the register area controlling the statement must be entered with three decimal digits. The register area of the statement is 8 bytes.

Description of the statement:

nnn: start address of the local area containing the registers used in the statement.

Address of registers	Meaning of registers
nnn	Format register
nnn+2	Start address of dividend (A)
nnn+4	Start address of divisor (B)
nnn+6	Start address of quotient (C) and remainder

Format register:

The format register can be found at address nnn of the register area. In this register the number of bytes, in which the numbers of statement are shown can be given.

Length of register: 1 word

Possible contents of register: 1, 2, or 4.

Start address of dividend (A):

The start address of the dividend can be found at address nnn+2 of the register area. This address must point to the local variable, at which the value of dividend can be found. At this address the number of bytes specified at format register is taken into account during the division in order to calculate the result. Bytes with lower local value are at the lower addresses, while those with higher local values are at higher addresses.

Start address of dividend is always a BCD number.

Start address of divisor (B):

The start address of the divisor can be found at address nnn+4 of the register area. This address must point to the local variable, at which the value of divisor can be found. At this address the number of bytes specified at format register is taken into account during the division in order to calculate the result. Bytes with lower local value are at the lower addresses, while those with higher local values are at higher addresses.

Start address of divisor is always a BCD number.

Start address of quotient (C) and remainder (R):

The start address of the result can be found at address nnn+6 of the register area. This address must point to the local variable, at which the value of the result can be found. At this address the number of bytes specified at format register is taken into account during the division in order to calculate the result. Bytes with lower local value are at the lower addresses, while those with higher local values are at higher addresses.

Start address of the quotient is always a BCD number.

After the execution of division the state of the following status flags can be tested:

F0080: Syntax error in statement

If the registers used for the statement are filled out correctly:

- the contents of format register is 1, 2, or 4,

\- the address registers refer to the address range of usable local variables.
Else flag F0080 is set to 1.
F0082: not BCD number
The flag is set to 1 if the values of address registers are not in BCD form.
F0046: The result is 0.
F0047: The result is negative

Example for the use of statement DIVnnn

```
Location:
F250...F256 -     input registers of statement DIV250
F290...F297 -     dividend
F286...F289 -     divisor
F298...F301 -     quotient
F302...F305 -     remainder

      .0004       ;length of numbers =4 (4 bytes)
      SF250       ;storing into division format register
      .0290       ;start address of dividend: F290(...F297)
      SF252       ;storing into dividend address
      .0286       ;start address of divisor: F286(...F289)
      SF254       ;storing into divisor address
      .0298       ;start address of quotient: F298(...F301, of remainder:
                  ;F302...F305)
      SF256       ;storing into quotient address
      DIV250      ;division
(F0080            ;if syntax error
OF0082)           ;or addresses are not in BCD form
      U725        ;DIVISION ERROR message strobe on
Z                 ;syntax error
                  ;end of condition
```

CMPnnn: comparing binary data
This statement is for comparing 1, 2, or 4 byte numbers, signed numbers, or the two's complement of the numbers. After the statement name (CMP) the start address of the register area controlling the statement must be entered with three decimal digits. The register area of the statement is 6 bytes.
Description of the statement:
nnn: start address of the local area containing the registers used in the statement.

Address of registers	Meaning of registers
nnn	Format register
nnn+2	Start address of basic data
nnn+4	Start address of compared data

<u>Format register:</u>
The format register can be found at address nnn of the register area. In this register the number of bytes, in which the numbers of statement are shown can be given.
Length of register: 1 word
Possible contents of register: 1, 2, or 4.

<u>Start address of basic data:</u>
The start address of the entered data can be found at address nnn+2 of the register area. This address must point to the local variable, at which the basic data can be found. At this address the number of bytes specified at format register is taken into account during the comparison in order

to calculate the result. Bytes with lower local value are at the lower addresses, while those with higher local values are at higher addresses.
Start address of entered data is always a BCD number.

Start address of compared data:
The start address of the compared data can be found at address nnn+4 of the register area. This address must point to the local variable, at which the compared data can be found. At this address the number of bytes specified at format register is taken into account during the comparison in order to calculate the result. Bytes with lower local value are at the lower addresses, while those with higher local values are at higher addresses.
Start address of compared data is always a BCD number.

The result of comparison can be read in the state of the status flags:
F0080: Syntax error in statement
If the registers used for the statement are filled out correctly:
- the contents of format register is 1, 2, or 4,
- the address registers refer to the address range of usable local variables.

Else flag F0080 is set to 1.
F0082: not BCD number
The flag is set to 1 if the values of address registers are not in BCD form.
F0046: The result is 0, i.e. the two data is equal
F0047: The result is negative, the basic data is less than the compared data
F0053: Overflow
If the result of comparison does not have enough room at the bytes, the number of which is specified at format register further bytes are not overwritten, but flag F0053 is set to 1.

Example for the use of statement CMPnnn

```
Location:
F260...F264 -     input registers of statement CMP260
F298...F301 -     entered data
F270...F273 -     compared data

      .0004       ;length of number =4 (4 bytes)
      SF260       ;storing into comparison format register
      .0298       ;start address of entered data: F298(...F301)
      SF262       ;storing into entered data address
      .0270       ;start address of compared data: F270(...F273)
      SF264       ;storing into compared data address
      CMP260      ;comparison
(F0080            ;if syntax error
OF0082            ;or addresses are not in BCD form
OF0053)           ;or overflow
      U726        ;COMPARISON ERROR message strobe on
E                 ;if no error
 F0046
      U727        ;EQUAL TO message strobe on
 E
  F0047
      U730        ;LESS THAN message strobe on
  E
      U731        ;GREATER THAN message strobe on
  Z
 Z
Z                 ;end of condition
                  ;syntax error
```

5 Compiling and Loading PLC Program into NC Control

The PLC source program is a text file, which is to be compiled for the NC control. The NC control is only able to execute the statements of the compiled program.
The source program can contain any number of comments. Comments can be used in two ways

; comment ${}^{C}R{}^{L}F$

i.e. comment start ";" is closed by carriage return (C_r) or line feed (L_f). The other possibility

/* comment */

is when brackets are added to the comment as seen above. This comment can contain however many lines.
The PLC program is to be loaded into the control compiled and in binary form.

PLC compiler Pe*.exe runs on MS DOS operating system of IBM PC or compatible computer. In place of character * the version number of the compiler can be replaced. The compiler regards only text files with extension *.plc as PLC programs, therefore it only loads those ones.
The following stipulations exist in connection with the length of the PLC program:

- The text length of the source program without comments and spaces, i.e. which is displayed by the compiler when compiling cannot be longer than 64 kB.
- The compiling is done at the lower 640 kB of the PC (Conventional Memory). The compiler program, the PLC source program and the operating system must have room in this memory. If in the course of compiling memory problem occurs DOS or Norton Commander must be directed to HMA (High Memory Area) or UMA (Upper Memory Area).

After starting the compiler the following menu items are offered:

F^{1} HELP:	starting the help
F^{2} LIBR:	selecting drive or directory. The selection is done by means of keys <up>, <down>, <right>, <left> and <ENTER>.
F^{8} COLOUR:	changing the colors of screen
F^{9} LANGUA:	languages to be selected: ENGLISH, DEUTSCH, MAGYAR
F^{10} QUIT:	exit from program

If a menu item has been selected the menu can be returned by the use of <Esc> (except for QUIT).

If (after selecting drive and directory) the program to be compiled has been selected (the PLC source must be saved to the directory with extension *.plc). After the highlighting bar has been set to the program key <ENTER> needs to be pressed. In this case the compiler compiles the program automatically, provided if no error has been found in it. The program statements are displayed on the screen (without comments). In case of error beginning with the erroneous statement the text is not formatted, but is displayed on screen in input format. The error message can be read on the bottom of screen. The error code list and their meanings can be read in the APPENDIX in chapter 6.3 Error Messages of the PLC Compiler on page 231.

If compiling is completed a file with extension *.bin beside the source with extension *.plc is created, which can be sent to the NC control. At the same time the compiler writes the time of compiling in form of

[year] [month] [day] [hour] [minute]

together with the version number of the compiler in the binary file. The above information is displayed on screen Service—PLC. Make sure, that the version of the software in the control and of the PLC compiler is the same. On the above mentioned screen also the information data entered by the programmer in module :200 can be read.

In this state the following actions are available by means of softkeys:

F^{1} HELP: starting the help
F^{2} COM1: the compiled PLC program (file *.bin) is sent to the control, provided the serial port of the PC is connected to input RS232C of the control. If the number of port is to be changed keys <1>, <2>, <3>, <4> must be used. ***This function can be used just in case of NCT98 and NCT99 controls.***
F^{3} MODUL↓: the list goes to the label of the next module in the displayed text
F^{4} MODUL↑: the list goes to the label of the previous module in the displayed text
F^{5} COND: If the cursor stands on the beginning of a condition, it goes to the condition closing Z, if it stands on a Z, it goes to the beginning of the state test.
F^{6} STAT: Here different statements and labels can be selected and the program evaluates, whether these references are in the text or not.
F^{8} ↓↑SEAR: it searches for the entered text. The search direction can be selected by the use of keys ↓and ↑.
F^{9} VALUE: If the PLC is connected to the control through serial interface the program perpetually updates the values of variables in the statements on the right side of screen. This gives help for the debugging of PLC program.
F^{10} QUIT: exit from program

If a menu item has been selected the menu can be returned by the use of <Esc> (except for QUIT).

In case of NCT98 and 99 controls the compiled program (with extension .bin) must be loaded.
For all bytes are halved in order to transfer them on serial line the length of the compiled binary file is two times the size the location the binary PLC program reserves in the control memory.

In case of NCT2000, 990, 100, 101, 104 and 115 controls the source code, that is the text file (with extension .plc) must be loaded.
The compilation of PLC program happens in the control after loading it. If the source code is syntactically erroneous the critical part is displayed and the same messages are produced as in case of version running on PC. Before loading a PLC program it is avised to check it by compiling it on a PC.

6 APPENDIX

6.1 Summary of the Variables of the Connection between PLC and NC

I400 Ref posit setting mode push-button
I401 Handle mode push-button
I402 Incremental jog mode push-button
I403 Jog mode push-button
I404
I405 Manual data input mode push-button
I406 Automatic mode push-button
I407 Edit mode push-button

I410 1st axis selector softkey
I411 2nd axis selector softkey
I412 3rd axis selector softkey
I413 4th axis selector softkey
I414 5th axis selector softkey
I415 6th axis selector softkey
I416 7th axis selector softkey
I417 8th axis selector softkey

I420 1 increment push-button
I421 10 increment push-button
I422 100 increment push-button
I423 1000 increment push-button
I424
I425
I426 Auto tool length measure softkey
I427 JOG rapid traverse push-button

I430 JOG 1 push-button
I431 JOG 2 push-button
I432 JOG 3 push-button
I433 JOG 4 push-button
I434 JOG 5 push-button
I435 JOG 6 push-button
I436 JOG 7 push-button
I437 JOG 8 push-button

Y400 Ref posit setting mode lamp
Y401 Handle mode lamp
Y402 Incremental jog mode lamp
Y403 Jog mode lamp
Y404
Y405 Manual data input mode lamp
Y406 Automatic mode lamp
Y407 Edit mode lamp

Y410 1st axis selected lamp
Y411 2nd axis selected lamp
Y412 3rd axis selected lamp
Y413 4th axis selected lamp
Y414 5th axis selected lamp
Y415 6th axis selected lamp
Y416 7th axis selected lamp
Y417 8th axis selected lamp

Y420 1 increment lamp
Y421 10 increment lamp
Y422 100 increment lamp
Y423 1000 increment lamp
Y424
Y425
Y426 Auto tool length measure lamp
Y427 JOG rapid traverse lamp

Y430 JOG X axis + direction selected
Y431 JOG Y axis + direction selected
Y432 JOG Z axis + direction selected
Y433 JOG + direction selected
Y434 JOG X axis – direction selected
Y435 JOG Y axis – direction selected
Y436 JOG Z axis – direction selected
Y437 JOG – direction selected

I440 Test push-button	**Y440** Test lamp
I441 Machine lock push-button	**Y441** Machine lock lamp
I442 Dry run push-button	**Y442** Dry run lamp
I443 Block restart push-button	**Y443** Block restart lamp
I444 Block return push-button	**Y444** Block return lamp
I445 Conditional stop push-button	**Y445** Conditional stop lamp
I446 Cond block skip push-button	**Y446** Conditional block skip lamp
I447 Single block push-button	**Y447** Single block lamp
I450 1st user’s push-button	**Y450** JOG 1 push-button lamp
I451 2nd user’s push-button	**Y451** JOG 2 push-button lamp
I452 3rd user’s push-button	**Y452** JOG 3 push-button lamp
I453 4th user’s push-button	**Y453** JOG 4 push-button lamp
I454 5th user’s push-button	**Y454** JOG 5 push-button lamp
I455 6th user’s push-button	**Y455** JOG 6 push-button lamp
I456 7th user’s push-button	**Y456** JOG 7 push-button lamp
I457 8th user’s push-button	**Y457** JOG 8 push-button lamp
I460 9th user’s push-button	**Y460** 1st axis lock selected
I461 10th user’s push-button	**Y461** 2nd axis lock selected
I462 11th user’s push-button	**Y462** 3rd axis lock selected
I463 12th user’s push-button	**Y463** 4th axis lock selected
I464 13th user’s push-button	**Y464** 5th axis lock selected
I465 14th user’s push-button	**Y465** 6th axis lock selected
I466 15th user’s push-button	**Y466** 7th axis lock selected
I467 16th user’s push-button	**Y467** 8th axis lock selected
I470 Start push-button	**Y470** Start state lamp
I471 Stop push-button	**Y471** Stop state lamp
I472 function lock push-button	**Y472** function lock lamp
I473	**Y473** Manual handle feed
I474 M3 push-button	**Y474** M3 of control board 2 lamp
I475 M4 push-button	**Y475** M4 of control board 2 lamp
I476 M5 push-button	**Y476** M5 of control board 2 lamp
I477 RESET push-button	**Y477** RESET from PLC
I480 1st user’s push-button	**Y480** 1st user’s push-button's lamp
I481 2nd user’s push-button	**Y481** 2nd user’s push-button's lamp
I482 3rd user’s push-button	**Y482** 3rd user’s push-button's lamp
I483 4th user’s push-button	**Y483** 4th user’s push-button's lamp
I484 5th user’s push-button	**Y484** 5th user’s push-button's lamp
I485 6th user’s push-button	**Y485** 6th user’s push-button's lamp
I486 7th user’s push-button	**Y486** 7th user’s push-button's lamp
I487 8th user’s push-button	**Y487** 8th user’s push-button's lamp

I490	**Y490**
I491	**Y491**
I492	**Y492**
I493	**Y493**
I494	**Y494**
I495	**Y495**
I496	**Y496**
I497	**Y497**
I500 PLC defined softkey 1	**Y500** PLC defined softkey 1 lamp
I501 PLC defined softkey 2	**Y501** PLC defined softkey 2 lamp
I502 PLC defined softkey 3	**Y502** PLC defined softkey 3 lamp
I503 PLC defined softkey 4	**Y503** PLC defined softkey 4 lamp
I504 PLC defined softkey 5	**Y504** PLC defined softkey 5 lamp
I505 PLC defined softkey 6	**Y505** PLC defined softkey 6 lamp
I506 PLC defined softkey 7	**Y506** PLC defined softkey 7 lamp
I508 PLC defined softkey 8	**Y508** PLC defined softkey 8 lamp
I510 first call of module :001	**Y510** conditional block 2 skip
I511 automatic operation interrupted	**Y511** conditional block 3 skip
I512	**Y512** conditional block 4 skip
I513	**Y513** conditional block 5 skip
I514	**Y514** conditional block 6 skip
I515	**Y515** conditional block 7 skip
I516	**Y516** conditional block 8 skip
I517 parts required=parts count	**Y517** conditional block 9 skip
I520 1st M function strobe	**Y520** Mode selection with softkeys
I521 2nd M function strobe	**Y521** Axis selection with softkeys
I522 3rd M function strobe	**Y522** Increment selection with softkeys
I523 4th M function strobe	**Y523** State selection with softkeys
I524 5th M function strobe	**Y524** PLC defined buttons with softkeys
I525 S function strobe	**Y525** R% with softkeys
I526 T function strobe	**Y526** S% with softkeys
I527 A function strobe	**Y527** F% with softkeys
I530 B function strobe	**Y530** Jog buttons from NC keyboard
I531 C function strobe	**Y531** Selection of mach control board 1
I532 Chopping Function Strobe	**Y532** Selection of mach control board 2
I533	**Y533**
I534	**Y534**
I535	**Y535**
I536 Valid push-b. code in reg RH049	**Y536** Valid push-button code from PLC
I537 Message on screen	**Y537** Data input from PLC

I540 Status of Machine on output
I541 Status of NC Ready signal
I542 Machine on output disabled
I543 module :000 start
I544
I545 programmed ref posit setting (G28)
I546 executable block in buffer
I547 STOP request from NC

I550 interpolator stopped
I551 interpolator empty
I552 override disabled
I553 spindle rotation request
I554 thread cutting (G33)
I555 Thread cutting cycle (G76, G78)
I556
I557

I560 1st axis in position
I561 2nd axis in position
I562 3rd axis in position
I563 4th axis in position
I564 5th axis in position
I565 6th axis in position
I566 7th axis in position
I567 8th axis in position

I570 1st axis lubrication request
I571 2nd axis lubrication request
I572 3rd axis lubrication request
I573 4th axis lubrication request
I574 5th axis lubrication request
I575 6th axis lubrication request
I576 7th axis lubrication request
I577 8th axis lubrication request

I580
I581
I582
I583
I584
I585
I586
I587

Y540 Machine on request
Y541 No input synchronization in :000
Y542 Feed hold
Y543 General security gate enable
Y544 Interrupt macro call enable
Y545 Free purpose user's timer enable
Y546 :002 call enable
Y547 FIN: functions executed by PLC

Y550 1st axis on reference switch
Y551 2nd axis on reference switch
Y552 3rd axis on reference switch
Y553 4th axis on reference switch
Y554 5th axis on reference switch
Y555 6th axis on reference switch
Y556 7th axis on reference switch
Y557 8th axis on reference switch

Y560 1st axis on + limit switch
Y561 2nd axis on + limit switch
Y562 3rd axis on + limit switch
Y563 4th axis on + limit switch
Y564 5th axis on + limit switch
Y565 6th axis on + limit switch
Y566 7th axis on + limit switch
Y567 8th axis on + limit switch

Y570 1st axis on – limit switch
Y571 2nd axis on – limit switch
Y572 3rd axis on – limit switch
Y573 4th axis on – limit switch
Y574 5th axis on – limit switch
Y575 6th axis on – limit switch
Y576 7th axis on – limit switch
Y577 8th axis on – limit switch

Y580 Tool sensor pressed in X+ direction
Y581 Tool sensor pressed in X– direction
Y582 Tool sensor pressed in Z+ direction
Y583 Tool sensor pressed in Z– direction
Y584
Y585
Y586
Y587

I590	**Y590** Axis 1 synchron slave on
I591	**Y591** Axis 2 synchron slave on
I592	**Y592** Axis 3 synchron slave on
I593	**Y593** Axis 4 synchron slave on
I594	**Y594** Axis 5 synchron slave on
I595	**Y595** Axis 6 synchron slave on
I596	**Y596** Axis 7 synchron slave on
I597	**Y597** Axis 8 synchron slave on
I600	**Y600** Program selection for automatic mode
I601	**Y601** Program selection for MDI mode
I602 Program execution in DNC	**Y602** Program execution in DNC
I603 Program execution in NCT DNC	**Y603** Program execution in NCT DNC
I604 Message acknowledged	**Y604** Message strobe
I605 Transmission error	**Y605** Open input channel
I606 Data transmitted from memory	**Y606** Transmittable data in memory
I607 Data received in memory	**Y607** PLC received data from memory
I610 1st axis motion request	**Y610** 1st axis motion disable
I611 2nd axis motion request	**Y611** 2nd axis motion disable
I612 3rd axis motion request	**Y612** 3rd axis motion disable
I613 4th axis motion request	**Y613** 4th axis motion disable
I614 5th axis motion request	**Y614** 5th axis motion disable
I615 6th axis motion request	**Y615** 6th axis motion disable
I616 7th axis motion request	**Y616** 7th axis motion disable
I617 8th axis motion request	**Y617** 8th axis motion disable
I620 1st axis rapid traverse request	**Y620** 1st axis loop open
I621 2nd axis rapid traverse request	**Y621** 2nd axis loop open
I622 3rd axis rapid traverse request	**Y622** 3rd axis loop open
I623 4th axis rapid traverse request	**Y623** 4th axis loop open
I624 5th axis rapid traverse request	**Y624** 5th axis loop open
I625 6th axis rapid traverse request	**Y625** 6th axis loop open
I626 7th axis rapid traverse request	**Y626** 7th axis loop open
I627 8th axis rapid traverse request	**Y627** 8th axis loop open
I630	**Y630** 1st axis motion by PLC
I631	**Y631** 2nd axis motion by PLC
I632	**Y632** 3rd axis motion by PLC
I633	**Y633** 4th axis motion by PLC
I634	**Y634** 5th axis motion by PLC
I635	**Y635** 6th axis motion by PLC
I636	**Y636** 7th axis motion by PLC
I637	**Y637** 8th axis motion by PLC

I640 G51.2: polygonal turning
I641 polyg. turn., reverse direction (Q<0)
I642
I643
I644
I645
I646
I647

I650 1st spindle command ramping ready
I651 1st spindle orientation ready
I652 1st spindle in position
I653 State G96 on active spindle
I654 State G25 on active spindle
I655 State G25 on active spindle
I656 1st spindle $n=n_S$
I657 1st spindle n=0

I660 2nd spindle command ramping ready
I661 2nd spindle orientation ready
I662 2nd spindle in position
I663 1st sp. synchronized to the 2nd one
I664 2nd sp. synchronized to the 1st one
I665
I666 2nd spindle $n=n_S$
I667 2nd spindle n=0

I670 1st analog command ramping ready
I671
I672 2nd analog command ramping ready
I673
I674
I675 Chopping Function Code
I676 Axis Is Chopping
I677 Chopping Axis on Point R

I680
I681
I682
I683
I684
I685
I686
I687

Y640 1st axis encoder check off
Y641 2nd axis encoder check off
Y642 3rd axis encoder check off
Y643 4th axis encoder check off
Y644 5th axis encoder check off
Y645 6th axis encoder check off
Y646 7th axis encoder check off
Y647 8th axis encoder check off

Y650 Active spindle rotates
Y651 1st spindle orientation request
Y652 1st spindle command signal enable
Y653 1st spindle com signal with + polarity
Y654 1st spindle binary com signal outp
Y655 Synchronize 1st spindle to the 2nd
Y656 1st sp. synchr. in counter direction
Y657 1st sp. orient. in the shorter direction

Y660 2nd spindle is active
Y661 2nd spindle orientation request
Y662 2nd spindle command signal enable
Y663 2nd spindle com signal with + polarity
Y664 2nd spindle binary com signal outp
Y665 Synchronize 2nd spindle to the 1st
Y666 2nd sp. synchr. in counter direction
Y667 2nd sp. orient. in the shorter direction

Y670 1st analog com signal with + polarity
Y671 1st analog com signal output binary
Y672 2nd analog com signal with+ polarity
Y673 2nd analog com signal output binary
Y674 Piston turning
Y675 Chopping On
Y676 1st analog com signal enable
Y677 2nd analog com signal enable

Y680
Y681
Y682
Y683
Y684
Y685
Y686
Y687

I690
I691
I692
I693
I694
I695
I696
I697

I700 1st indexed message on the screen
I701 2nd indexed message on the sreen
I702 3rd indexed message on the sreen
I703 4th indexed message on the sreen
I707 5th indexed message on the sreen
I705 6th indexed message on the sreen
I706 7th indexed message on the sreen
I707 8th indexed message on the sreen

I710 1st message on the sreen
I711 2nd message on the sreen
I712 3rd message on the sreen
I713 4th message on the sreen
I714 5th message on the sreen
I715 6th message on the sreen
I716 7th message on the sreen
I717 8th message on the sreen

....

I790 65th message on the sreen
I791 66th message on the sreen
I792 67th message on the sreen
I793 68th message on the sreen
I794 69th message on the sreen
I795 70th message on the sreen
I796 71st message on the sreen
I797 72nd message on the sreen

Y690
Y691
Y692
Y693
Y694
Y695
Y696
Y697

Y700 1st indexed message request
Y701 2nd indexed message request
Y702 3rd indexed message request
Y703 4th indexed message request
Y707 5th indexed message request
Y705 6th indexed message request
Y706 7th indexed message request
Y707 8th indexed message request

Y710 1st message request
Y711 2nd message request
Y712 3rd message request
Y713 4th message request
Y714 5th message request
Y715 6th message request
Y716 7th message request
Y717 8th message request

....

Y790 65th message request
Y791 66th message request
Y792 67th message request
Y793 68th message request
Y794 69th message request
Y795 70th message request
Y796 71st message request
Y797 72nd message request

I800 73rd message on the sreen
I801 74th message on the sreen
I802 75th message on the sreen
I803 76th message on the sreen
I804 77th message on the sreen
I805 78th message on the sreen
I806 79th message on the sreen
I807 80th message on the sreen

....

I890 145th message on the sreen
I891 146th message on the sreen
I892 147th message on the sreen
I893 148th message on the sreen
I894 149th message on the sreen
I895 150th message on the sreen
I896 151st message on the sreen
I897 152nd message on the sreen

I900 1st axis interpolator stopped
I901 1st axis interpolator empty
I902
I903 1st axis reference point ready
I904
I905
I906
I907 1st axis drive ready

I910 2nd axis interpolator stopped
I911 2nd axis interpolator empty
I912
I913 2nd axis reference point ready
I914
I915
I916
I917 2nd axis drive ready

Y800 73rd message request
Y801 74th message request
Y802 75th message request
Y803 76th message request
Y804 77th message request
Y805 78th message request
Y806 79th message request
Y807 80th message request

....

Y890 145th message request
Y891 146th message request
Y892 147th message request
Y893 148th message request
Y894 149th message request
Y895 150th message request
Y896 151st message request
Y897 152nd message request

Y900 1st axis interpolator START
Y901 1st axis interpolator strobe signal
Y902 1st axis movement with feed
Y903 1st axis incremental movement
Y904 1st axis go to reference point
Y905 1st axis interpolator RESET
Y906
Y907

Y910 2nd axis interpolator START
Y911 2nd axis interpolator strobe signal
Y912 2nd axis movement with feed
Y913 2nd axis incremental movement
Y914 2nd axis go to reference point
Y915 2nd axis interpolator RESET
Y916
Y917

I920 3rd axis interpolator stopped
I921 3rd axis interpolator empty
I922
I923 3rd axis reference point ready
I924
I925
I926
I927 3rd axis drive ready

I930 4th axis interpolator stopped
I931 4th axis interpolator empty
I932
I933 4th axis reference point ready
I934
I935
I936
I937 4th axis drive ready

I940 5th axis interpolator stop
I941 5th axis interpolator empty
I942
I943 5th axis reference point ready
I944
I945
I946
I947 5th axis drive ready

I950 6th axis interpolator stopped
I951 6th axis interpolator empty
I952
I953 6th axis reference point ready
I954
I955
I956
I957 6th axis drive ready

I960 7th axis interpolator stopped
I961 7th axis interpolator empty
I962
I963 7th axis reference point ready
I964
I965
I966
I967 7th axis drive ready

Y920 3rd axis interpolator START
Y921 3rd axis interpolator strobe signal
Y922 3rd axis movement with feed
Y923 3rd axis incremental movement
Y924 3rd axis go to reference point
Y925 3rd axis interpolator RESET
Y926
Y927

Y930 4th axis interpolator START
Y931 4th axis interpolator strobe signal
Y932 4th axis movement with feed
Y933 4th axis incremental movement
Y934 4th axis go to reference point
Y935 4th axis interpolator RESET
Y936
Y937

Y940 5th axis interpolator START
Y941 5th axis interpolator strobe signal
Y942 5th axis movement with feed
Y943 5th axis incremental movement
Y944 5th axis go to reference point
Y945 5th axis interpolator RESET
Y946
Y947

Y950 6th axis interpolator START
Y951 6th axis interpolator strobe signal
Y952 6th axis movement with feed
Y953 6th axis incremental movement
Y954 6th axis go to reference point
Y955 6th axis interpolator RESET
Y956
Y957

Y960 7th axis interpolator START
Y961 7th axis interpolator strobe signal
Y962 7th axis movement with feed
Y963 7th axis incremental movement
Y964 7th axis go to reference point
Y965 7th axis interpolator RESET
Y966
Y967

I970 8th axis interpolator stopped
I971 8th axis interpolator empty
I972
I973 8th axis reference point ready
I974
I975
I976
I977 8th axis drive ready

I980
I981
I982
I983
I984
I985
I986
I987 1st main drive ready

I990
I991
I992
I993
I994
I995
I996
I997 2nd main drive ready

Y970 8th axis interpolator START
Y971 8th axis interpolator strobe signal
Y972 8th axis movement with feed
Y973 8th axis icremental movement
Y974 8th axis go to reference point
Y975 8th axis interpolator RESET
Y976
Y977

Y980
Y981
Y982
Y983
Y984
Y985
Y986
Y987

Y990
Y991
Y992
Y993
Y994
Y995
Y996
Y997

RH000 1st M function code
RH001 2nd M function code
RH002 3rd M function code
RH003 4th M function code
RH004 5th M function code
RH005 S function code
RH006 T function code
RH007 A function code
RH008 B function code
RH009 C function code

RH050 Number of prg to be executed
RH051 Start address of data to be sent
RH052 Number of bytes to be sent
RH053 Transmitter periphery code
RH054 Start address of received data
RH055 Number of received bytes
RH056 Receiver periphery code
RH057 A function current value
RH058 B function current value
RH059 C function current value

RH010 1st spindle current revolution
RH011 1st spindle modified prg rev
RH012 G96 revol. on the active spindle
RH013 Progrd max. rev. on active spindle
RH014
RH015 2nd spindle current revolution
RH016 2nd spindle modified prg rev
RH017
RH018
RH019

RH020 active message code
RH021 Year
RH022 Month, Day
RH023 Hour, Minute
RH024 Second
RH025
RH026 Meanings of softkeys
RH027 Screen codes
RH028 F%
RH029 S%

RH030 Number of prg under execution
RH031 Number of prg selected for auto
RH032 Number of prg selected for MDI
RH033
RH034
RH035 1st analog input on 1st INT board
RH036 2nd analog input on 1st INT board
RH037 3rd analog input on 1st INT board
RH038 4th analog input on 1st INT board
RH039 R%

RH040 G51.2 polyg. turn. data P
RH041 G51.2 polyg. turn. data Q
RH042 Actual feed lower word
RH043 Actual feed higher word
RH044
RH045
RH046
RH047
RH048
RH049 Code of valid push-button

RH060 1st spindle programmed S register
RH061 1st spindle binary command register
RH062 1st spindle rotation code
RH063 1st spindle range code
RH064 Active tool code (T)
RH065 2nd spindle programmed S register
RH066 2nd spindle binary command register
RH067 2nd spindle rotation code
RH068 2nd spindle range code
RH069

RH070 1st M group display
RH071 2nd M group display
RH072 3rd M group display
RH073 4th M group display
RH074 5th M group display
RH075 6th M group display
RH076 7th M group display
RH077 8th M group display
RH078 F%
RH079 S%

RH080 1st analog scaled com signal
RH081 1st analog binary com signal
RH082 1st analog.%
RH083
RH084
RH085 2nd analog scaled com signal
RH086 2nd analog binary com signal
RH087 2nd analog %
RH088 Chopping Override Register
RH089 R%

RH090 1st Y700 message variable
RH091 2nd Y701 message variable
RH092 3rd Y702 message variable
RH093 4th Y703 message variable
RH094 5th Y704 message variable
RH095 6th Y705 message variable
RH096 7th Y706 message variable
RH097 8thY707 message variable
RH098
RH099 Push-button code form PLC

RH100 1st axis current position lower word
RH101 1st axis current position upper word
RH102 1st axis lag lower word
RH103 1st axis lag upper word
RH104 1st axis drive current
RH105 2nd axis current position lower word
RH106 2nd axis current position upper word
RH107 2nd axis lag lower word
RH108 2nd axis lag upper word
RH109 2nd axis drive current

RH110 3rd axis current position lower word
RH111 3rd axis current position upper word
RH112 3rd axis lag lower word
RH113 3rd axis lag upper word
RH114 3rd axis drive current
RH115 4th axis current position lower word
RH116 4th axis current position upper word
RH117 4th axis lag lower word
RH118 4th axis lag upper word
RH119 4th axis drive current

RH120 5th axis current position lower word
RH121 5th axis current position upper word
RH122 5th axis lag lower word
RH123 5th axis lag upper word
RH124 5th axis drive current
RH125 6th axis current position lower word
RH126 6th axis current position upper word
RH127 6th axis lag lower word
RH128 6th axis lag upper word
RH129 6th axis drive current

RH130 7th axis current position lower word
RH131 7th axis current position upper word
RH132 7th axis lag lower word
RH133 7th axis lag upper word
RH134 7th axis drive current
RH135 8th axis current position lower word
RH136 8th axis current position upper word
RH137 8th axis lag lower word
RH138 8th axis lag upper word
RH139 8th axis drive current

RH150 1st axis position com lower word
RH151 1st axis position com upper word
RH152 1st axis feedrate com lower word
RH153 1st axis feedrate com upper word
RH154
RH155 2nd axis position com lower word
RH156 2nd axis position com upper word
RH157 2nd axis feedrate com lower word
RH158 2nd axis feedrate com upper word
RH159

RH160 3rd axis position com lower word
RH161 3rd axis position com upper word
RH162 3rd axis feedrate com lower word
RH163 3rd axis feedrate com upper word
RH164
RH165 4th axis position com lower word
RH166 4th axis position com upper word
RH167 4th axis feedrate com lower word
RH168 4th axis feedrate com upper word
RH169

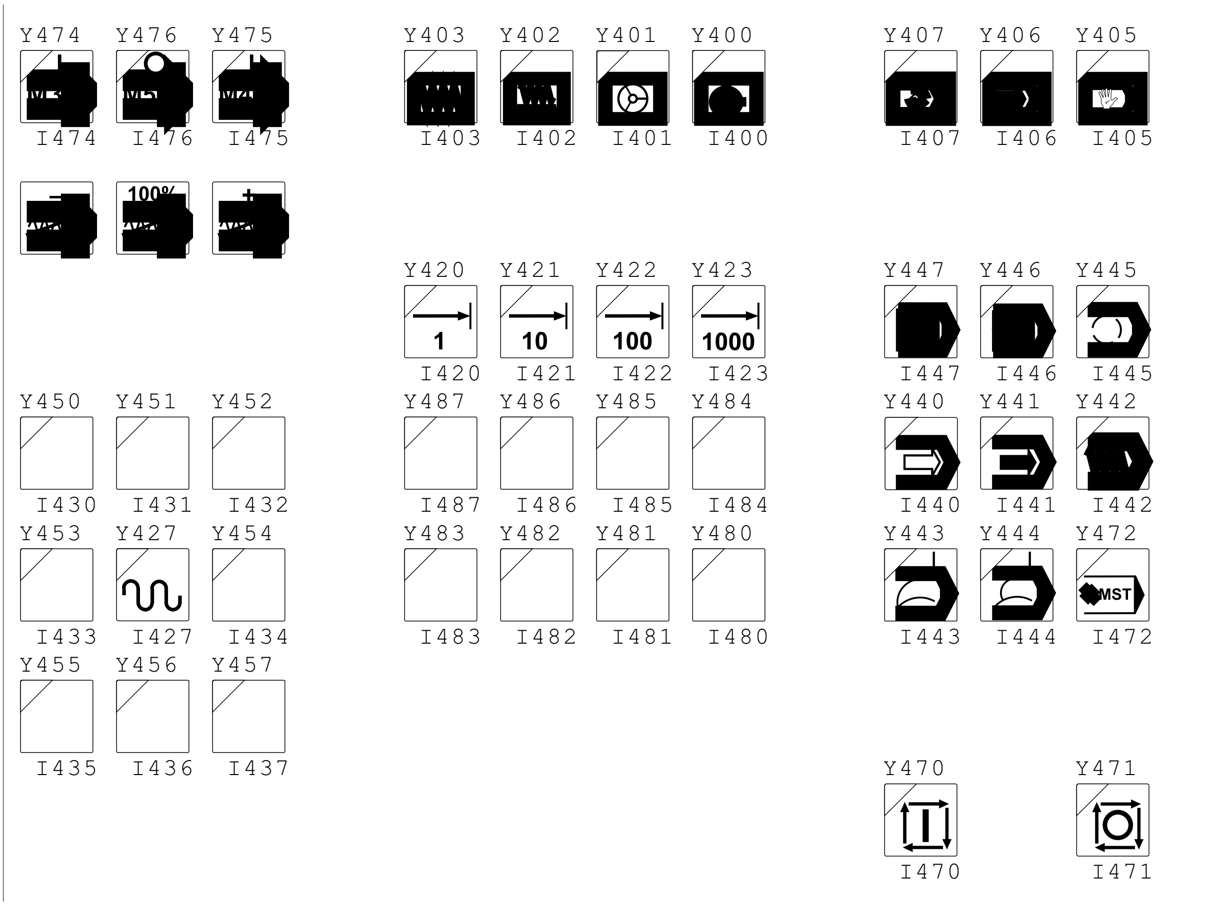
RH170 5th axis position com lower word
RH171 5th axis position com upper word
RH172 5th axis feedrate com lower word
RH173 5th axis feedrate com upper word
RH174
RH175 6th axis position com lower word
RH176 6th axis position com upper word
RH177 6th axis feedrate com lower word
RH178 6th axis feedrate com upper word
RH179

RH180 7th axis position com lower word
RH181 7th axis position com upper word
RH182 7th axis feedrate com lower word
RH183 7th axis feedrate com upper word
RH184
RH185 8th axis position com lower word
RH186 8th axis position com upper word
RH187 8th axis feedrate com lower word
RH188 8th axis feedrate com upper word
RH189

RH140
RH141
RH142
RH143
RH144 1st main drive current
RH145
RH146
RH147
RH148
RH149 2nd main drive current

RH190 Number of axis doing ovality
RH191 Position of longer diameter
RH192 Ovality lower word
RH193 Ovality higher word
RH194 Barrellity lower word
RH195 Barrellity higher word
RH196
RH197
RH198
RH199

6.2 The Bit Map of Machine Control Board 2



6.3 Error Messages of the PLC Compiler

01 identity number of module exceeds 200
02 unnecessary “Z” in program
03 too long PLC object code (compiled PLC program)
04 full address table (too many statements)
05 no module :000
06 no module :001
07 statement not interpretable
08 no module
09 not decimal or octal number
10 not hexadecimal number
11 no closing parenthesis ')' or ']' found
12 number of levels > 8
13 illegal character after 'N'
14 illegal character after 'NL'
15 illegal character after 'NS'
16 value of number exceeds 2 bytes
17 condition test not closed
18 no condition test after opening parenthesis "("
19 not decimal number
20 no statement name “L” before name of variable when referred to within brackets “[...]”
21 illegal statement within parentheses
22 illegal statement SRPnnn
23 illegal character after 'SR' or 'LR'
24 shift count >15 when shifting OP left (statement <<nn)
25 shift count >15 when shifting OP right (statement >>nn)
26 illegal character after "B"
27 illegal character after "BI"
28 illegal character after "BC"
29 too long PLC source program
30 illegal character after "S"
31 illegal character after "<"
32 illegal character after "<N"
33 illegal character after "="
34 illegal character after "=N"
35 illegal character after ">"
36 illegal character after ">N"
37 illegal character after "<="
38 illegal character after "<=N"
39 illegal character after ">="
40 illegal character after ">=N"
41 illegal reference (:198 - :200)
42 identity number of counter > 31 in statement Q
43 identity number of timer > 49, 99, 9 in statements T, H, M
44 character not interpretable
45 illegal character after multiplication "*" or division " /"

46	invalid address nnn in statements HF, PF, MR, MW, ADD, SUB, MUL, DIV, CMP
47	illegal character after "AD" (ADD)
48	illegal character after "SU" (SUB)
49	invalid PARAMETER index
50	illegal character after P
51	illegal character after "L" (in statement loading)
52	illegal character after "MU" (MUL)
53	reference to non-existing module
54	existing identity number of module
55	message module filled out incorrectly
56	illegal character after "DI" (DIV)
57	false index after statement "J"
58	writing at odd I/O address
59	illegal character after "CM" (CMP)
60	reference to non-existing I/O port (number of port>7)
61	no J0 or J1 in PLC program
62	false or useless statement name within parentheses
63	invalid condition connection (false: ,5 AI002; correct: ,5 ALI002)
64	index of statement RH is greater than 199
65	length of one of the messages is greater than 25 characters in module :199
66	index in statement SRH is not in the following ranges: $050 \leq index \leq 099$, or $150 \leq index \leq 199$
67	illegal reference in statement G (G001, G002)
68	illegal reference in statement C (C000, C001, C002)
69	length of one of the indexed messages is greater than 20 characters in module :198
70	no comma before $
71	instruction R befor J0, J1, J2
72	length of message > 16 characters
73	“E” without “Z”
74	before text modul not instruction Gnnn, R, Jn or $
75	J0, J1 instructio in condition expression
76	no comment character
77	
78	
79	
80	
81	
82	
83	
84	
85	
86	
87	
88	
89	
90	
91	

92
93
94
95
96
97
98
99

6.4 Listing of Global Messages

Below the code of each global message is listed and the message written by the control in message field is given. For detailed description of messages, reason of error as well as trouble shooting see “Operator’s Manual”.

0	SERVO 1
1	SERVO 2
2	SERVO 3
3	SERVO 4
4	SERVO 5
5	SERVO 6
6	SERVO 7
7	SERVO 8
8	SERVO 9
20	ENCODER 1
21	ENCODER 2
22	ENCODER 3
23	ENCODER 4
24	ENCODER 5
25	ENCODER 6
26	ENCODER 7
27	ENCODER 8
28	ENCODER 9
40	FEEDBACK 1
41	FEEDBACK 2
42	FEEDBACK 3
43	FEEDBACK 4
44	FEEDBACK 5
45	FEEDBACK 6
46	FEEDBACK 7
47	FEEDBACK 8
48	FEEDBACK 9
60	PLC TIMEOUT 1
61	PLC TIMEOUT 2
70	DPG TIMEOUT
80	15V FAILER
90	SYNC. FAILER 1
91	SYNC. FAILER 2
92	SYNC. FAILER 3
93	SYNC. FAILER 4
94	SYNC. FAILER 5
95	SYNC. FAILER 6
96	SYNC. FAILER 7
97	SYNC. FAILER 8
100	SHORT 000
120	SHORT 020
200	SHORT 100
220	SHORT 120
300	SHORT 200
320	SHORT 220
400	SHORT 300
420	SHORT 320
999	SHORT MON
1020	POSITION ERROR
1100	REFERENCE POINT t1
1110	
1120	
1130	
1140	
1150	
1160	
1170	
1101	REFERENCE POINT t2
1111	
1121	
1131	
1141	
1151	
1171	
1102	REFERENCE POINT t3
1112	
1122	
1132	
1142	
1152	
1162	
1172	
1103	REFERENCE POINT t4
1113	
1123	
1133	
1143	
1153	
1163	
1173	
1104	REFERENCE POINT t5
1114	
1124	
1134	
1144	
1154	
1164	
1174	
1105	REFERENCE POINT t6
1115	
1125	
1135	
1145	
1155	
1165	
1175	
1300	FORBIDDEN AREA t+
1301	

1302		3010	PLANE SELECT. IN G41, G42
1303		3011	RADIUS DIFFERENCE
1304		3012	ERRONEOUS CIRCLE DEF. R
1305		3013	MULTITURN CIRCLE FAILER
1306		3014	ERRONEOUS CIRCLE DEF.
1307		3015	
1320	FORBIDDEN AREA t–	3016	
1321		3017	,C AND ,R IN ONE BLOCK
1322		3018	,A IN G2, G3
1323		3019	DOMINATOR CONSTANT=0
1324		3020	DATA DEFINITION ERROR G33
1325		3021	G51 IN G33
1326		3022	DIVIDE BY 0 IN G33
1327		3023	DATA DEFINITION ERROR G26
1340	LIMIT t+	3024	ERRONEOUS P VALUE IN G96
1341		3025	DEFINITION ERROR S
1342		3026	DEFINITION ERROR G10 L3
1343		3027	DEFINIT. ERROR T IN G10 L3
1344		3028	MORE TOOLS IN G10 L3
1345		3029	MORE GROUPS IN G10 L3
1346		3030	DEFINITION ERROR T
1347		3031	ALL TOOL LIVES ARE OVER
1360	LIMIT t–	3032	CONFLICTING M CODES
1361		3033	DEFINITION ERROR M
1362		3034	DEFINITION ERROR A,B,C
1363		3035	DEFINITION ERROR P
1364		3036	G39 CODE IN G40
1365		3037	BEFORE G39 NOT G1, G2, G3
1366		3038	G38 NOT IN G0, G1STATE
1367		3039	G38 CODE IN G40
1380	SPINDLE LOOP OPEN	3040	G38 NOT IN G0, G1
1400	INTERNALLY FORBIDDEN AREA	3041	AFTER G2, G3 ILLEG. BLOCK
2000	PLC ERROR 001	3042	G40 IN G2, G3
2001	PLC ERROR 002	3043	G41, G42 IN G2, G3
2002	PLC ERROR 003	3044	G41, G42 DEFINITION ERROR
...		3045	
...		3046	NO INTERSECTION G41, G42
2150	PLC ERROR 151	3047	CHANGE NOT POSSIBLE
2151	PLC ERROR 152	3048	INTERFERENCE ALARM
2500	PLC MESSAGE 1	3049	CIRCLE ARC TOO LONG
2501	PLC MESSAGE 2	3050	NO REFRNC POINT G29, G30
2502	PLC MESSAGE 3	3051	G22, G28, ... G31, G37
2503	PLC MESSAGE 4	3052	ERROR IN G76, G87
2504	PLC MESSAGE 5	3053	NO BOTTOM OR R POINT
2505	PLC MESSAGE 6	3054	G31 IN INCORRECT STATE
2506	PLC MESSAGE 7	3055	G37 IN INCORRECT STATE
2507	PLC MESSAGE 8	3056	LIMIT
3000	MIRROR IMAGE IN G51, G68	3057	FORBIDDEN AREA
3001	VALUE EXCESS X,Y,...F	3058	NOT IN DNC
3002	PLANE SELECTION IN G68	3059	
3003	COORDINATE ADDRESS G68	3060	
3004	MISSING REFERENCE POINT	3061	
3005	ILLEGAL G CODE	3062	
3006	VALUE EXCESS H, D, P	3063	
3007	G43, G44, H IN G2, G3	3064	BAD MACRO STATEMENT
3008	ERRONEOUS G45...G48	3065	TOO LONG BLOCK
3009	G45...G48 IN G41, G42	3066	NO INTERSECTION POINT

3067	FAULTY ,A IN G16
3068	FAULTY READ
3069	LEVEL EXCESS
3070	NOT EXISTING BLOCK NO. P
3071	MISSING OR FAULTY P
3072	DEFINITION ERROR L
3073	NOT EXISTING PROGRAM NO.
3074	ODD G67
3075	DEFINITION ERROR N
3076	NO END OF PROGRAM
3077	
3078	
3079	
3080	ERRONEOUS USE OF #
3081	DEFINITION ERROR ,C ,R
3082	NO RETURN M99
3083	R=0
3084	,C ,R TOO HIGH
3085	CIRCLE ERROR G51
3086	DEFINITION ERROR G51
3087	
3088	
3089	BUFFER OVERRUN G41, G42
3090	# DEFINITION PROHIBITED
3091	ERRONEOUS OPERATION WITH #
3092	DIVISION BY 0 #
3093	BUFFER OVERRUN #
3094	
3095	
3096	
3097	
3098	ERRONEOUS ARGUMENT
3099	
3100	
3101	BLOCK NOT FOUND
3102	INCORRECT POSITION G12.1
3103	OUT OF RANGE
3104	
3105	
3106	
3107	
3108	
3109	
3110	
3111	
3112	
3113	
3114	
3115	
3116	
3116	
3118	
3119	
3120	
3121	
3122	
3123	
3124	
3125	
3126	
3127	
3500	PROGRAM EDITED
3502	BAD BAUDRATE VALUE
3503	SERIAL BUFFER FULL
3504	TOOL PLACE TABLE BAD
3505	NOT EXISTING PROGRAM
3507	OVERWRITE (Y/N)
3508	NC STATUS TABLE BAD
3509	LIFE TIME TABLE BAD
3510	TOOL OFFSET TABLE BAD
3511	WORK OFFSET TABLE BAD
3512	MEMORY LOCKED
3513	PLC PROGRAM BAD
3514	OVERRUN ERROR
3515	PARITY ERROR
3516	FRAMING ERROR
3518	DIRECTORY FULL
3519	MEMORY FULL
3520	FILE NOT EXISTS
3521	FILE READ ONLY
3522	BCC ERROR
3523	OVERRREAD ERROR
3524	FILE NOT OPEN
3525	FILE EXIST
3527	INVALID PASSWORD
3528	INVALID ERROR CODE
3530	SYSTEM ERROR
3545	MACRO TABLE BAD
3547	RAMDISK ERROR
3549	RESTORE MODAL FUNCTIONS? Y
3550	RESTORE MODAL FUNCTIONS? N
4000	MACRO ERROR 000
4001	MACRO ERROR 001
4002	MACRO ERROR 002
...	...
4999	MACRO ERROR 999
5000	MACRO MESSAGE 000
5001	MACRO MESSAGE 001
5002	MACRO MESSAGE 002
...	...
5999	MACRO MESSAGE 999

6.5 Listing of Push-button Codes

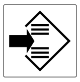
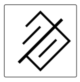
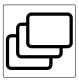
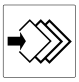
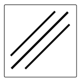
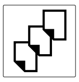
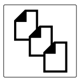
The number of buttons on NC or data input keyboard delivered together with control may differ. Codes of keyboards of different design are the same for corresponding functions or characters. The only difference is that certain characters (e.g. lower cases) can be entered on many-key keyboards but not on few-key ones. The keys or key combinations with which the appropriate functions or characters are activated are shown beside the code.

Codes of NC keyboard delivered with 15" monitor (RH049 contents I536=1)

code	button	function	code	key	function	code	key	function	code	key	function
00h	F1		01h	F2		02h	F3		03h	F4	
04h	F5		05h	F6		06h	F7		07h	F8	
08h	F9		09h	F0		0Ah	↓		0Bh	↑	
0Ch	→		0Dh	←		0Eh		INS	0Fh		DEL
10h		screen	11h		action	12h			13h		CAN-CEL
14h		PG UP	15h		PG DN	16h			17h		
18h			19h			1Ah			1Bh	+/–	sign
1Ch			1Dh			1Eh			1Fh	.	decimal point
20h		space	21h	shift ?	!	22h	“	“	23h	shift =	#
24h	shift ,	$	25h	shift :	%	26h	shift “	&	27h		
28h	shift [	(	29h	shift]	)	2Ah	shift /	*	2Bh	shift –	+
2Ch	,	,	2Dh	–	–	2Eh			2Fh	/	/
30h	0	0	31h	1	1	32h	2	2	33h	3	3
34h	4	4	35h	5	5	36h	6	6	37h	7	7
38h	8	8	39h	9	9	3Ah	:	:	3Bh		
3Ch	shift >	<	3Dh	=	=	3Eh	>	>	3Fh	?	?
40h	shift space		41h	A	A	42h	B	B	43h	C	C
44h	D	D	45h	E	E	46h	F	F	47h	G	G
48h	H	H	49h	I	I	4Ah	J	J	4Bh	K	K
4Ch	L	L	4Dh	M	M	4Eh	N	N	4Fh	O	O
50h	P	P	51h	Q	Q	52h	R	R	53h	S	S
54h	T	T	55h	U	U	56h	V	V	57h	W	W
58h	X	X	59h	Y	Y	5Ah	Z	Z	5Bh	[	[

code	button	function	code	key	function	code	key	function	code	key	function
5Ch			5Dh	]	]	5Eh			5Fh		
60h			61h	shift A	a	62h	shift B	b	63h	shift C	c
64h	shift D	d	65h	shift E	e	66h	shift F	f	67h	shift G	g
68h	shift H	h	69h	shift I	i	6Ah	shift J	j	6Bh	shift K	k
6Ch	shift L	l	6Dh	shift M	m	6Eh	shift N	n	6Fh	shift O	o
70h	shift P	p	71h	shift Q	q	72h	shift R	r	73h	shift S	s
74h	shift T	t	75h	shift U	u	76h	shift V	v	77h	shift W	w
78h	shift X	x	79h	shift Y	y	7Ah	shift Z	z	7Bh		
7Ch			7Dh			7Eh	⇧	SHIFT	7Fh		

Codes of NC keyboard delivered with 9" monitor (RH049 contents I536=1)

code	key	function	code	key	function	code	key	function	code	key	function
00h	F1		01h	F2		02h	F3		03h	F4	
04h	F5		05h			06h			07h		
08h			09h			0Ah	↓		0Bh	↑	
0Ch	→		0Dh	←		0Eh		INS	0Fh		DEL
10h		screen	11h		action	12h			13h		CAN-CEL
14h		PG UP	15h		PG DN	16h	↠		17h	↞	
18h			19h			1Ah			1Bh	+/–	sign
1Ch			1Dh			1Eh			1Fh	.	decimal point
20h		space	21h	shift .	!	22h	shift T	“	23h	shift 7	#
24h			25h	shift O	%	26h			27h		
28h	shift +/-	(	29h	shift 0	)	2Ah	shift 5	*	2Bh	shift 8	+
2Ch	shift G	,	2Dh	shift 9	–	2Eh			2Fh	shift 6	/
30h	0	0	31h	1	1	32h	2	2	33h	3	3
34h	4	4	35h	5	5	36h	6	6	37h	7	7
38h	8	8	39h	9	9	3Ah	shift N	:	3Bh		
3Ch			3Dh	shift 4	=	3Eh			3Fh	shift 1	?

code	key	function	code	key	function	code	key	function	code	key	function
40h	shift space		41h	shift I	A	42h	shift J	B	43h	shift K	C
44h	shift H	D	45h	shift F	E	46h	F	F	47h	G	G
48h	H	H	49h	I	I	4Ah	J	J	4Bh	K	K
4Ch	shift S	L	4Dh	M	M	4Eh	N	N	4Fh	O	O
50h	shift M	P	51h	shift R	Q	52h	R	R	53h	S	S
54h	T	T	55h	shift X	U	56h	shift Y	V	57h	shift Z	W
58h	X	X	59h	Y	Y	5Ah	Z	Z	5Bh	shift 2	[
5Ch			5Dh	shift 3	]	5Eh			5Fh		
60h			61h			62h			63h		
64h			65h			66h			67h		
68h			69h			6Ah			6Bh		
6Ch			6Dh			6Eh			6Fh		
70h			71h			72h			73h		
74h			75h			76h			77h		
78h			79h			7Ah			7Bh		
7Ch			7Dh			7Eh	⇧	SHIFT	7Fh		

6.6 Codes of Screen Menu and Action Menu Captions

Codes of screens in register RH027 in case of ***NCT98*** and ***NCT99***:

RH027	upper byte									
lower byte	**01h**	**02h**	**03h**	**04h**	**05h**	**06h**	**07h**	**08h**	**09h**	**0Ah**
01h	Oprtr's Panel									
02h	Absolt	Relatv	Machin	End	Overll					
03h	Text	Functn	Last	Active	Messag					
04h	Direc-tory	View	Edit	Block input						
05h	Work offsts	Tool offsts	W. offs measur	T. leng measur	Rel. ps offsts					
06h	Grphcs setting	Draw								
07h	#1-#33	#100-#199	#500-#599	Timer / countr	Tool pot	PLC table	User's params	Secrty		
08h	Params	PLC	Test I/O	Logic anal	Test mes	Scope	Errors	Monitor	Version	
09h										
0Ah										

Codes of screens in register RH027 in case of ***NCT2000, 990, 100, 101***, ***104*** and ***115***:

RH027	upper byte									
lower byte	**01h**	**02h**	**03h**	**04h**	**05h**	**06h**	**07h**	**08h**	**09h**	**0Ah**
01h	Absolt	Relatv	Machin	End	Overll	Cartsn				
02h	Text	Functn	Last	Active	Cntrl Pn	Message				
03h	Direc-tory	View	Edit	Block input						
04h	Work offsts	Tool offsts	W. offs measur	T. leng measur	Rel. ps offsts					
05h	Grphcs setting	Draw								
06h	#1-#33	#100-#199	#500-#599	Timer / countr	Tool pot	PLC table	User's params	Secrty		
07h	Params	PLC	Test I/O	Logic anal	Test mes	Scope	Errors	Monitor	Version	
08h										
09h										
0Ah										

That is if the contents of register RH027: RH027=0104h, then sceen DIRECTORY is displayed in case of NCT99 controls while Work offsts in NCT2000.
If the PLC needs to transmit data input key codes to NC and sets flag Y537 to 1 screen ABSOLUTE POSITION is displayed and register RH027 acknowledges this screen code:

RH027=0102h (NCT99)
RH027=0101h (NCT2000)

Softkey codes can be found in register RH026. If the upper byte of the register is 0 the screen menu is seen on softkeys, if the upper byte is 1 the action menu is apparent.

RH026=00xxh: screen menu

RH026=01xxh: action menu

Independent of the upper byte (screen menu or action menu) state the lower byte of register always shows the code of the previously selected action menu belonging to the screen.

If the PLC needs to transmit data input key codes to NC and sets flag Y537 to 1 softkeys and register RH026 are set to default state:

RH026=0000h

RH026		**lower byte**										
upper byte		action menu	sub-menus of action menu									
			1	2	3	4	5	6	7	8	9	0
		00h	01h	02h	03h	04h	05h	06h	07h	08h	09h	0Ah
01h	**F1**	1	1.1	2.1	3.1	4.1	5.1	6.1	7.1	8.1	9.1	0.1
	F2	2	1.2	2.2	3.2	4.2	5.2	6.2	7.2	8.2	9.2	0.2
	F3	3	1.3	2.3	3.3	4.3	5.3	6.3	7.3	8.3	9.3	0.3
	F4	4	1.4	2.4	3.4	4.4	5.4	6.4	7.4	8.4	9.4	0.4
	F5	5	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5	0.5
	F6	6	1.6	2.6	3.6	4.6	5.6	6.6	7.6	8.6	9.6	0.6
	F7	7	1.7	2.7	3.7	4.7	5.7	6.7	7.7	8.7	9.7	0.7
	F8	8	1.8	2.8	3.8	4.8	5.8	6.8	7.8	8.8	9.8	0.8
	F9	9	1.9	2.9	3.9	4.9	5.9	6.9	7.9	8.9	9.9	0.9
	F0	0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	0.0

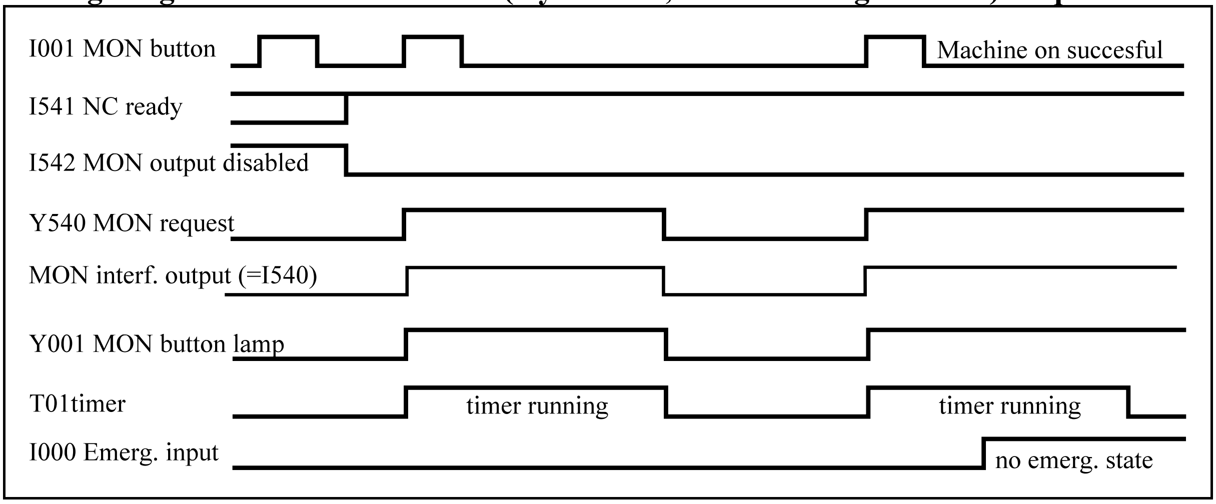
On the basis of the above table the lower byte of the register can accept values 01h, 02h, ... if the action menus belonging to the screen have sub-menus.

For example let us examine the codes of actions belonging to DIRECTORY screen. The upper byte of the register is 01h, thus action menu is on softkeys. If the lower byte is 00h action menu captions (New, Search, ...) can be found on softkeys. The lower byte cannot be 01 since softkey New F1 is action key, thus it implements data input. Softkey Load F4 is action menu key, i.e. it covers further actions. Therefore when it is pressed the value of the lower byte changes to 04h showing, that actions of action menu Load (Serial, Ramdisc, ...) can be found on softkeys.

RH026		**lower byte**										
upper byte		action menu	sub-menus of action menu									
					Delete	**Load**	**Save**	**Run**	**Reset**	**Arran-ge**		
		00h	01h	02h	03h	04h	05h	06h	07h	08h	09h	0Ah
01h	**F1**	**New**			Ram-disc	Serial	Serial	Auto	OK	Increa-sing		
	F2	**Search**			OK	Ram-disc	Ram-disc	MDI.	Cancel	Decre-asing		
	F3	**Delete**			Cancel	Prom	OK	DNC		Select-ed		
	F4	**Load**				OK	Cancel	DNC NCT		Type		
	F5	**Save**				Cancel		Table		Size (byte)		
	F6	**Run**								No		
	F7	**Reset**								OK		
	F8	**Arran-ge**										
	F9	**Protec-ted**										
	F0											

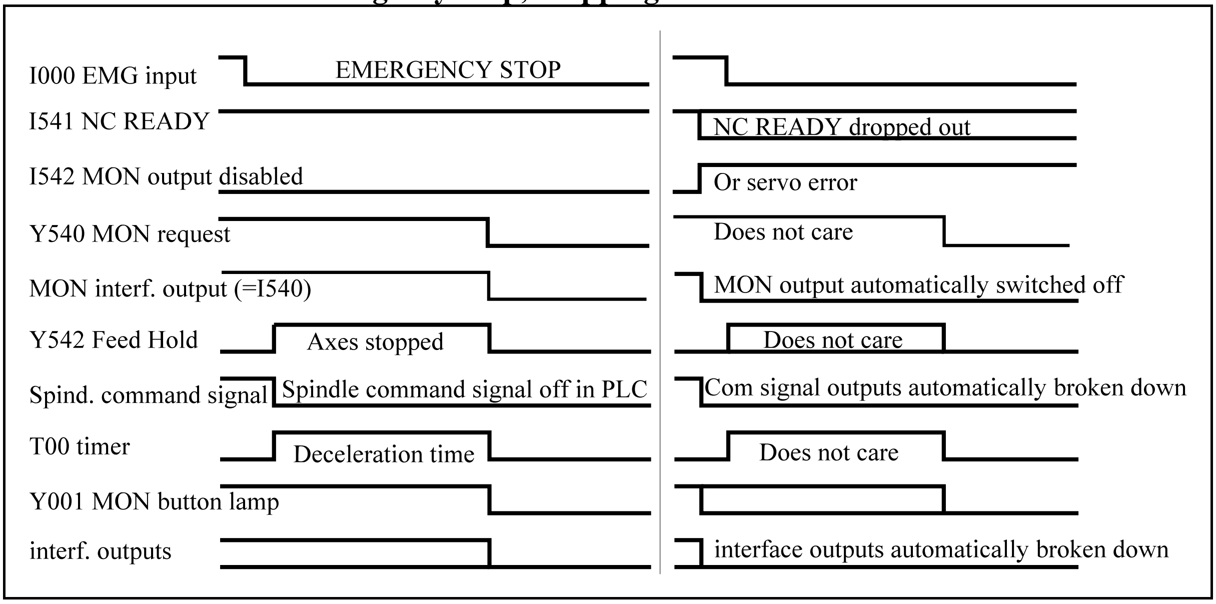
6.7 Timing Diagrams of PLC Variables

Timing Diagram of the Machine On (Hydraulics, Machine Magnetics on) Request



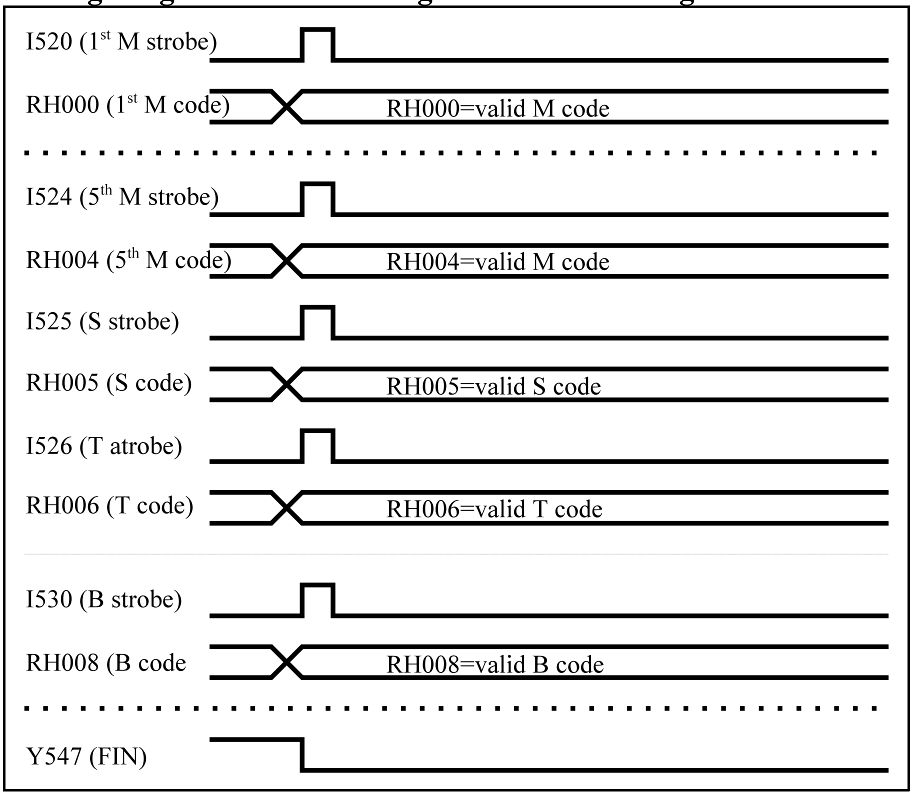
The machine can be turned on when the MON output is not disabled (I542=0). As the effect of button MACHINE ON timer T01 is started. If the machine is not in emergency state input line EMG is set to 1. If this signal has arrived before the termination of the timer has run off MON output is left switched on, otherwise it is switched off.

Procedure in Case of Emergency Stop, Stopping of NC READY or Servo Error.



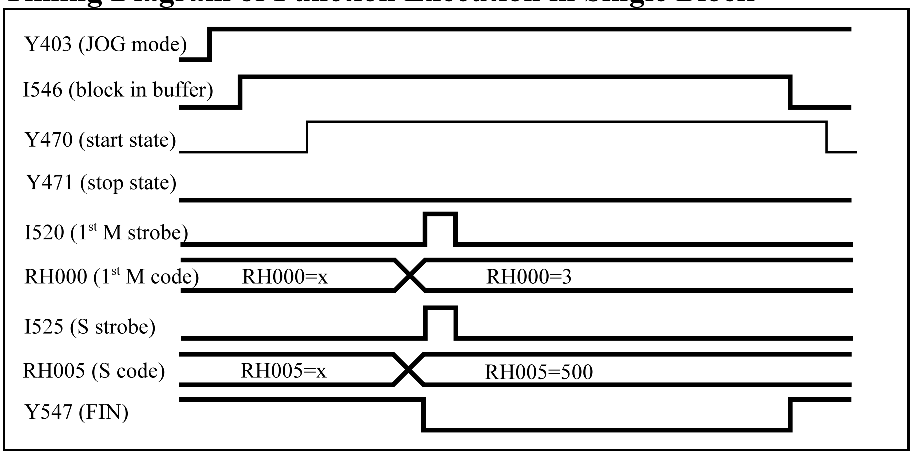
In case of emergency stop, if emergency state is activated with a lag regarding the drive permissions a deceleration process can be started by zeroing the spindle command signal and switching FEED HOLD flag on. The time period of deceleration is initialized at timer T00, than after the termination of the timer the MON output line is switched off by the PLC.
If the NC READY is stopped or the control detects servo error the switching MON output on disabled flag is immediately set to 1, the MON output line, the command signal transfer lines and all the interface outputs are instantly switched off by the control, independent of the PLC. The machine can be started again only after turning the control off.

Timing Diagram of Strobe Flags and Transfer Registers of Functions



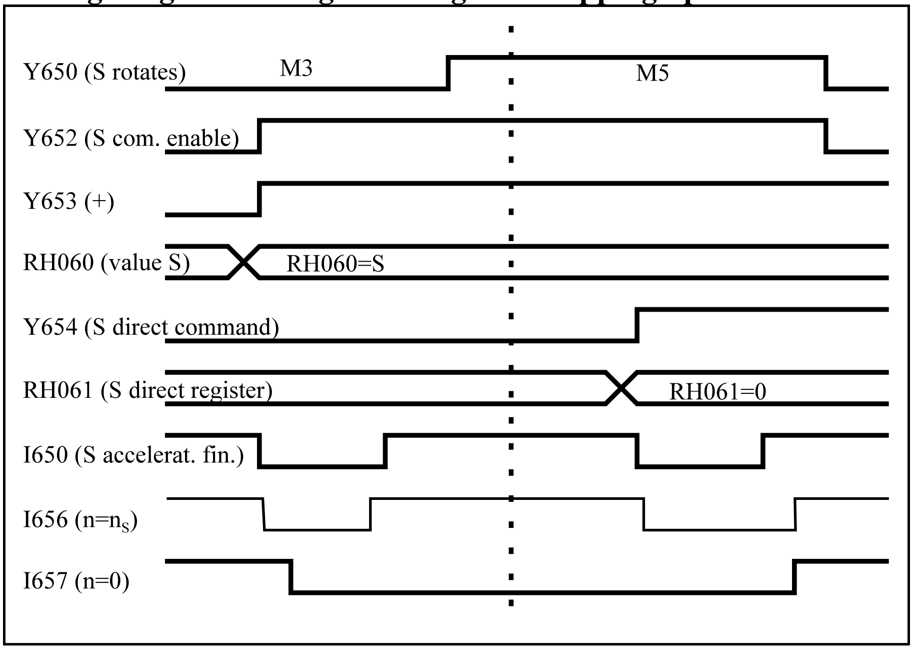
All functions entered into the program block are transferred to the PLC in the same period. The strobe flag, in the transfer register of which valid code is transferred, is set to 1 till the end of the PLC period, than it is set back to 0. When receiving the appropriate code decoding the command and setting FIN (functions executed) flag to 0 is the task of the PLC. The FIN flag is set back to 1 by the PLC after every function has been executed. This informs the NC that the function part of the block has been executed.

Timing Diagram of Function Execution in Single Block



In the above example the execution of single block M3 S500 is shown in JOG mode. If executable block in buffer flag I546 is set to 1 the execution can be started with the help of START button. After the block has been decoded by the preprocessor module through strobe signals I520, I525 and transfer registers RH000 and RH005 the block is sent to the PLC for execution. The PLC sets FIN flag Y547 to 0 until the command is under execution. After execution FIN flag is set to 1 the NC cancels executable block in buffer flag I546, than the PLC cancels start state Y470.

Timing Diagram of Flags Starting and Stopping Spindle Rotation



The above diagram shows the case when the stopped spindle is rotated in direction M3, than stopped by means of command M5.

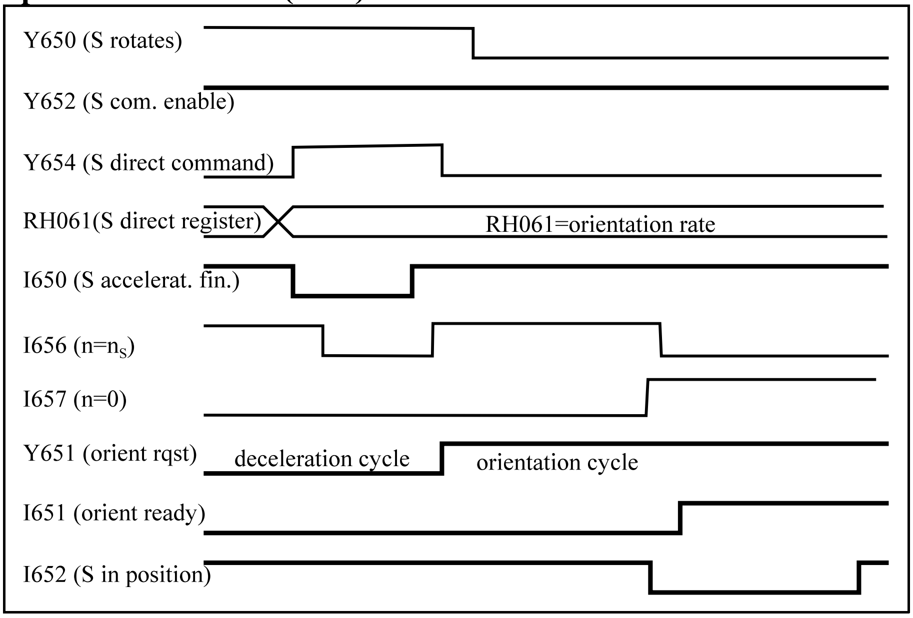
In case of command M3 before setting command signal transfer enabled flag Y652 the direction (Y653) must be specified, Y654=0, i.e. the command signal is taken from register RH060 and programmed code S is written into register RH060.

Flag I650 is set to 1 if the command signal integrator in NC has reached the value corresponding to the programmed revolution, and flag I656 is set to 1 if the spindle reached the programmed revolution. Afterwards spindle rotation flag (Y650) can be switched on.

In case of command M5 RH061 must be set to 0, and flag Y654 to 1, i.e. the command signal is taken from register RH061.

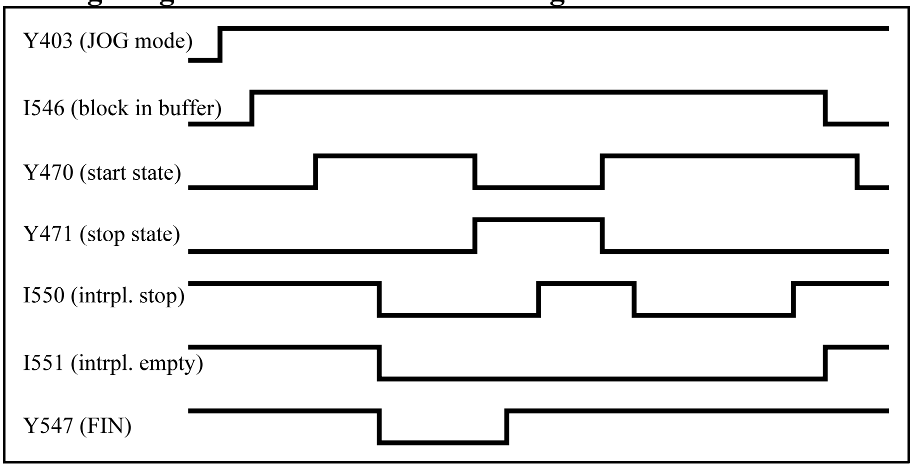
After the command signal integrator has reached level No. 0 (I650=1)and 0 rotation signal has been received (I657=1), i.e. the spindle has stopped, command signal transfer enabled flag Y652 and spindle rotation flag Y650 must be switched off.

Spindle Orientation (M19) Started from Rotation State



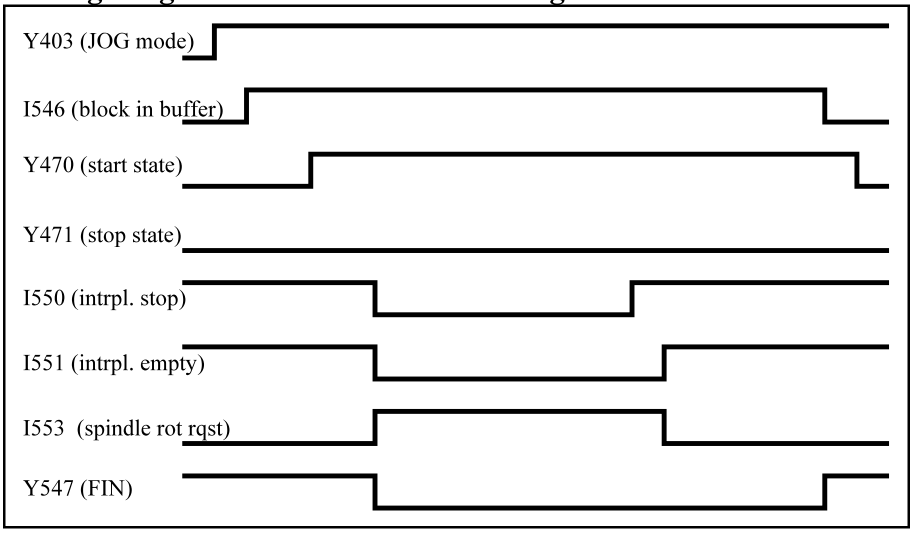
First the spindle has to be decelerated by setting register RH061 and flagY654 to 1 (command signal transfer from register RH061).
After the spindle has decelerated (I650=1 and I656=1) orientation request flag Y651 must be set.
The orientation is finished when orientation ready flag I651, as well as spindle in position flag I652 returns. During and after the process the spindle command signal transfer enabled flag Y652 must be switched on.

Timing Diagram of the Execution of Single Block G0 X150 M3 S500



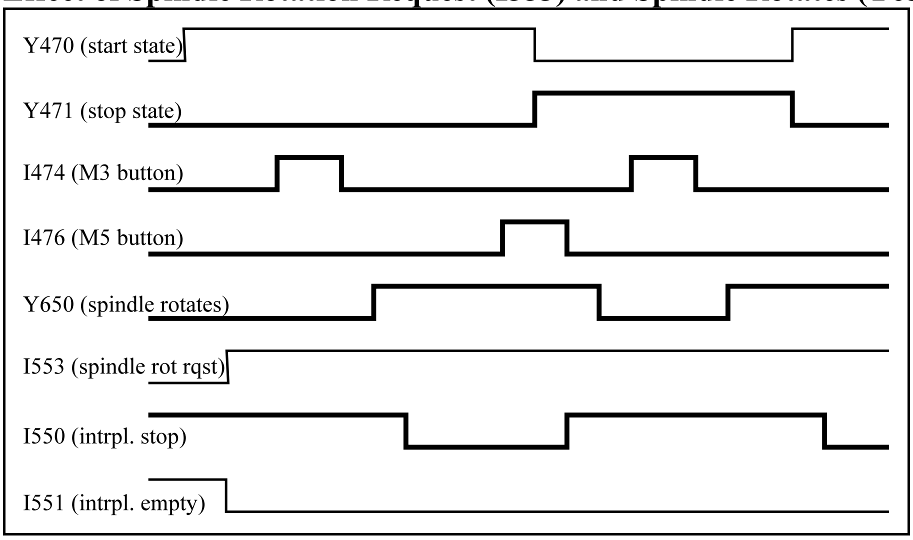
If special block G0 X150 M3 S500 is entered in JOG mode following the block input executable block in buffer flag I546 is set to 1. In this situation the execution may be started (Y470).
After the preprocessor had processed the block it sends its commands to the interpolator and the PLC for execution. At this point flags I550, I551 are set to 0 by the interpolator and FIN flag Y547 is set to 0 by the PLC.
The interpolation and spindle rotation occur simultaneously and the PLC finishes block execution earlier. On this the PLC informs the NC by setting FIN signal to 1.
STOP can be issued during movement: Y470=0, Y471=1. In this case the interpolator stops following a deceleration process, which is indicated by state I550=1.
After restart (Y470=1, Y471=0) the interpolator moves the rest path to be done and sets flags I550 and I551 to 1. If both flag Y547 (FIN) and I551 (empty interpolator) are set to 1 the block execution is finished and flag I546 is set to 0 by the NC. Afterwards start and stop states can be canceled.

Timing Diagram of the Execution of Single Block G1 X0 M5



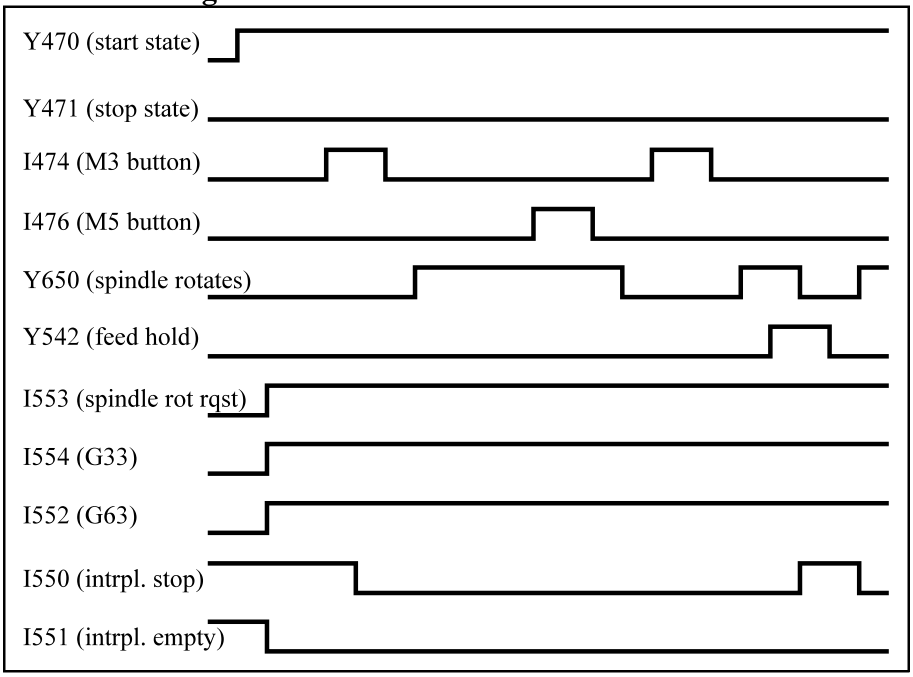
If single block G1 X0 M5 is entered in JOG mode following the block termination executable block in buffer flag I546 is set to 1. In this situation the execution may be started (Y470).
After the program module preprocessor had prepared the block it sends the commands of the block to the interpolator and the PLC for execution. At this point flags I550, I551 are set to 0 by the interpolator and FIN flag Y547 is set to 1 by the PLC.
In block G1 (spindle rotation request flag I553 set to 1) the PLC must wait until the interpolation is finished, which is indicated by the TRUE state of flag I551 (empty interpolator).
Afterwards the execution of command M5 can be started the end of which is indicated by Y547=1. If both flag Y547 (FIN) and I551 (interpolator empty) are set to10 the block execution is finished and flag I546 is set to 0 by the NC. Afterwards start and stop states can be canceled.

Effect of Spindle Rotation Request (I553) and Spindle Rotates (Y650) Flags



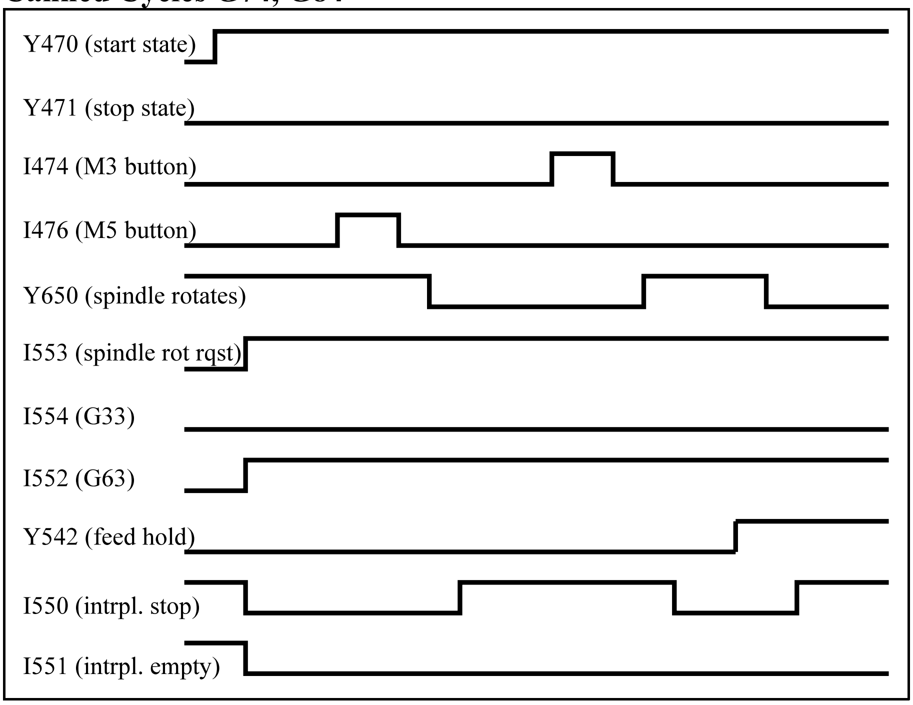
In blocks G1, G2, G3 the interpolator asks for spindle rotation through flag I553. The movement of interpolator is started after the PLC switched spindle rotates flag Y650 on. On the diagram the spindle rotation starts as the effect of button M3 (flag I474) .
If the rotation is stopped (as the effect of button M5 flag I476) the PLC must wait until the interpolator is finished, only than can the spindle be stopped. In case of restart the spindle rotation must be started before pressing START.

Thread Cutting Block G33



In case of thread cutting G33 the interpolator asks for spindle rotation through flag I553. Flag I552 of override disabled command G63 and flag I554 of thread cutting command G33 are switched on.
If pulses are started from the spindle encoder the thread cutting can be started. The thread cutting cannot be stopped with STOP button. The feed is stopped only if the spindle rotation has been already stopped, because this way pulses are not coming from the encoder any longer. However interpolator stop signal is not set to 1, for the interpolator keeps on waiting for the encoder pulses of spindle. The thread cutting can be restarted by means of button M3.
Be aware of stopping spindle from PLC when switching FEED HOLD signal (Y542) on, for all movements are instantly stopped due to the FEED HOLD signal.

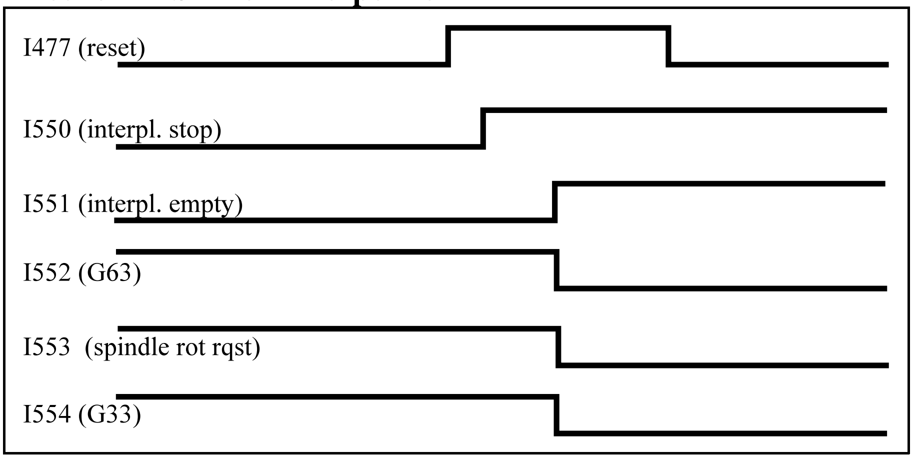
Canned Cycles G74, G84



In case of tapping G74, G84 the interpolator asks for spindle rotation through flag I553. Flag I552 of override disabled command G63 and flag I554 of thread cutting command G33 are switched on.
If spindle rotation flag Y650 is returned the milling is started. The milling cannot be stopped with STOP button.
The feed can be stopped only if the

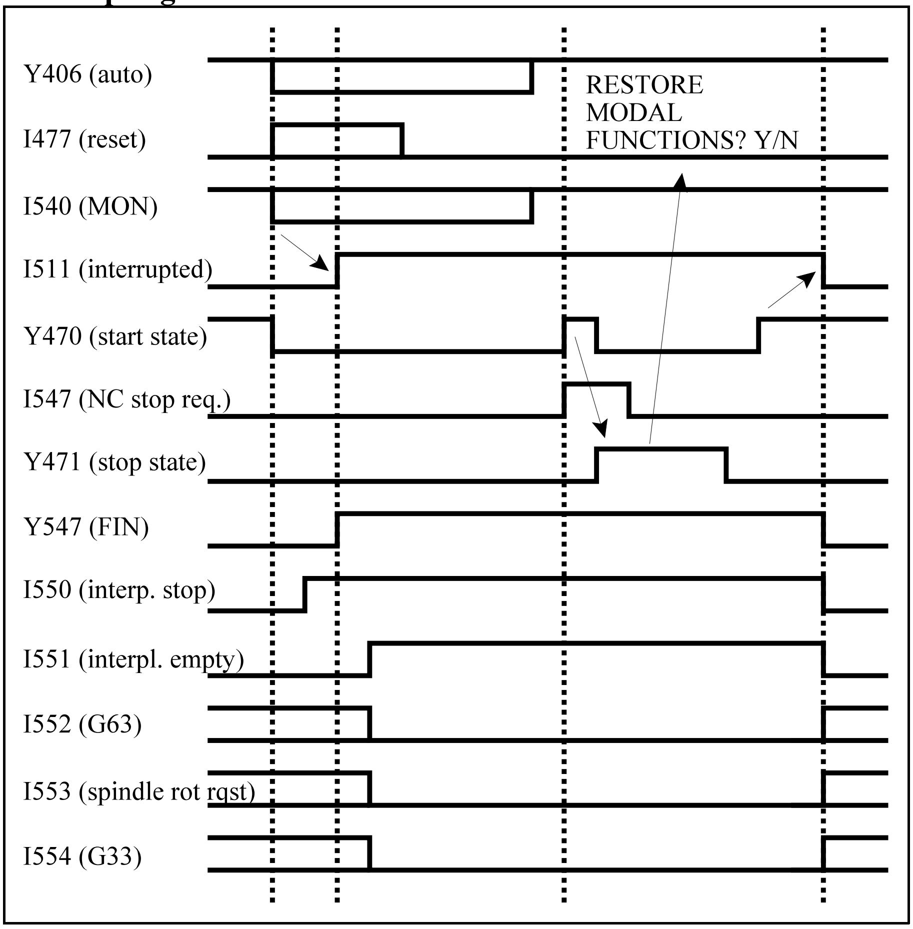
spindle rotation has been already stopped, because in 0 state of spindle rotation flag there is no feed.
Spindle rotation flag Y650 can be switched off as the effect of button M5. The command can be restarted by means of button M3.
The feed may be stopped by FEED HOLD (Y542=1) in this case however the PLC programmer must take care of stopping the spindle rotation.

Effect of RESET on Interpolator



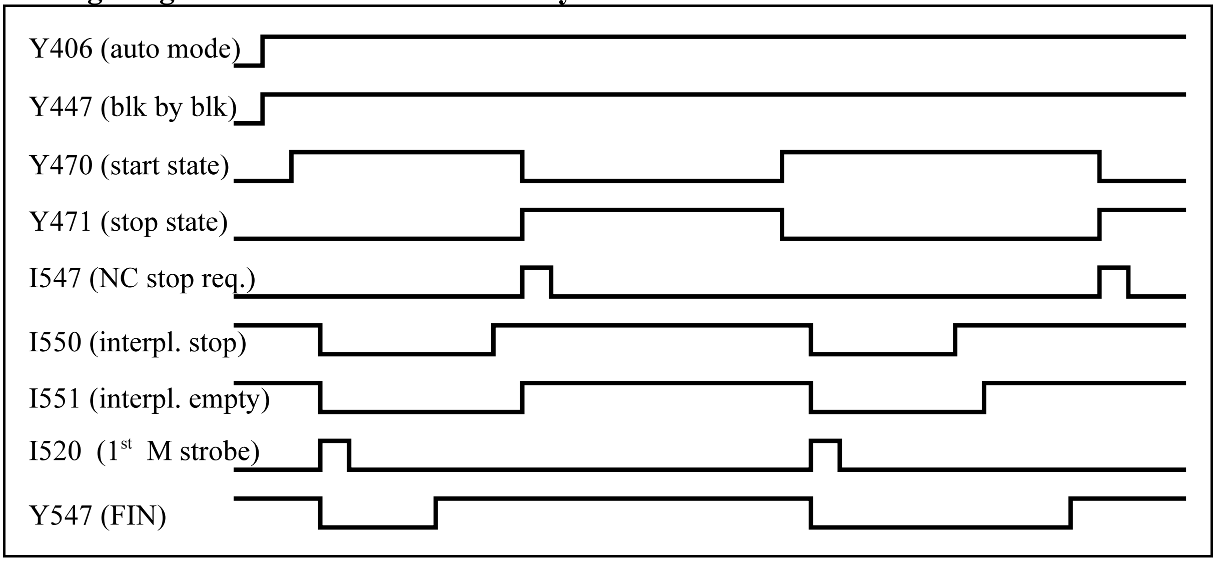
By pressing RESET button (I477=1) the interpolator gets to standard state, i.e. it stops after decelerating (I550=0) switches interpolator empty flag I551 on and flag I552 of override disabled command (G63) and flag I554 of thread cutting (G33) are switched off.
After pressing RESET handling the machine tool is the PLC programmer's task.

Interrupting Automatic Mode



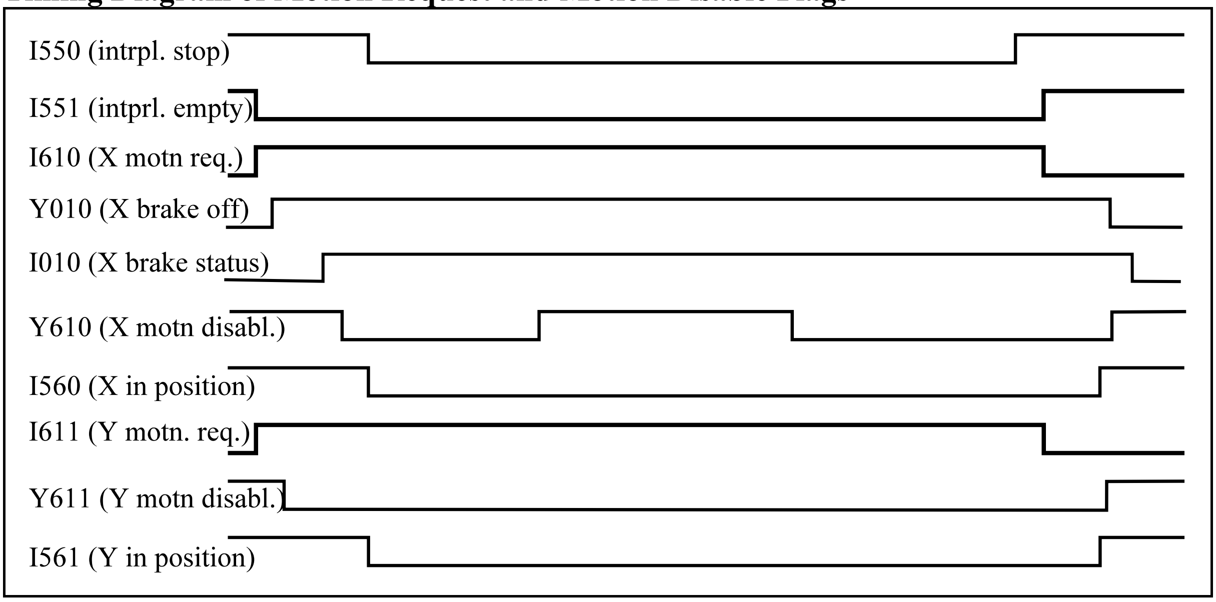
The automatic mode can be interrupted by exiting from the mode, pressing RESET button or turning off the machine, e.g. as the effect of emergency stop (switching MON off). The NC stops the interpolator, than switches flag I511 (HOLD state) on, PLC saves the functions not executed yet in HOLD state, and sets FIN flag to 1.
In case of HOLD state, if START has been pressed in automatic mode the NC asks for stop through flag I547. In STOP state (Y471=1) message RESTORE MODAL FUNCTIONS? Y, or (after pressing <shift> button) RESTORE MODAL FUNCTIONS? N is displayed.
After selecting Y(es) or N(o) HOLD state can be canceled (I511=0) with the help of START button. The NC starts the interpolator, the PLC restores the saved functions not executed before suspension and switches FIN signal off (Y547=0).

Timing Diagram of Execution in Block by Block Mode



In case of execution in block by block mode (Y447=1) at the end of block (Y547=1 and I551=1) the NC informs on registering STOP state through flag I547. At this point the start state must be switched off and the stop state switched on in the PLC.

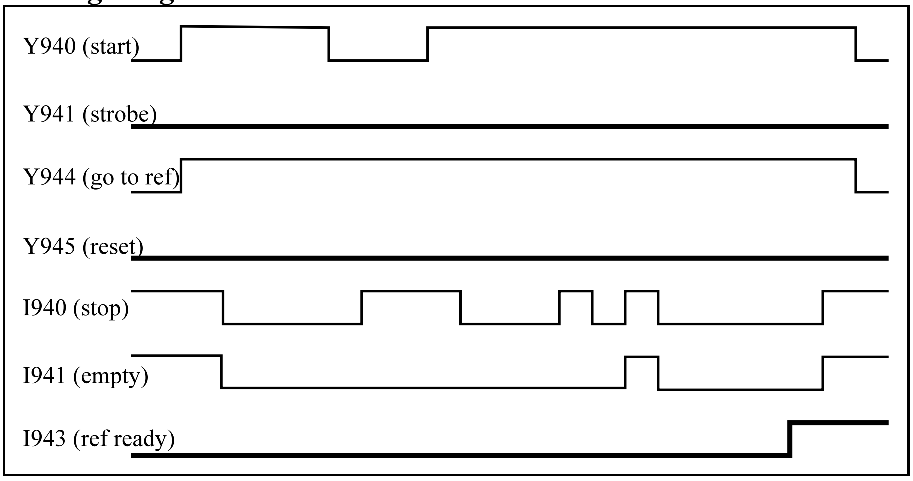
Timing Diagram of Motion Request and Motion Disable Flags



The movement is not started in the appropriate axis till the movement disabled flag is on. Movement request flag ceases only if the interpolator has stopped on the given axis. If two or more axes are involved in the interpolation, the interpolation does not start unless there is movement enable on each axis taking part in the interpolation.
After movement request (I610=1) brake unclamp output is switched on (Y010=1), feedback is awaited (I010=1), than the movement is enabled (Y610=0).
After the movement is finished (I610=0) in position signal is awaited (I560=1), than movement is disabled (Y610=1), and the brake unclamp is switched off (Y010=0). The process ends if

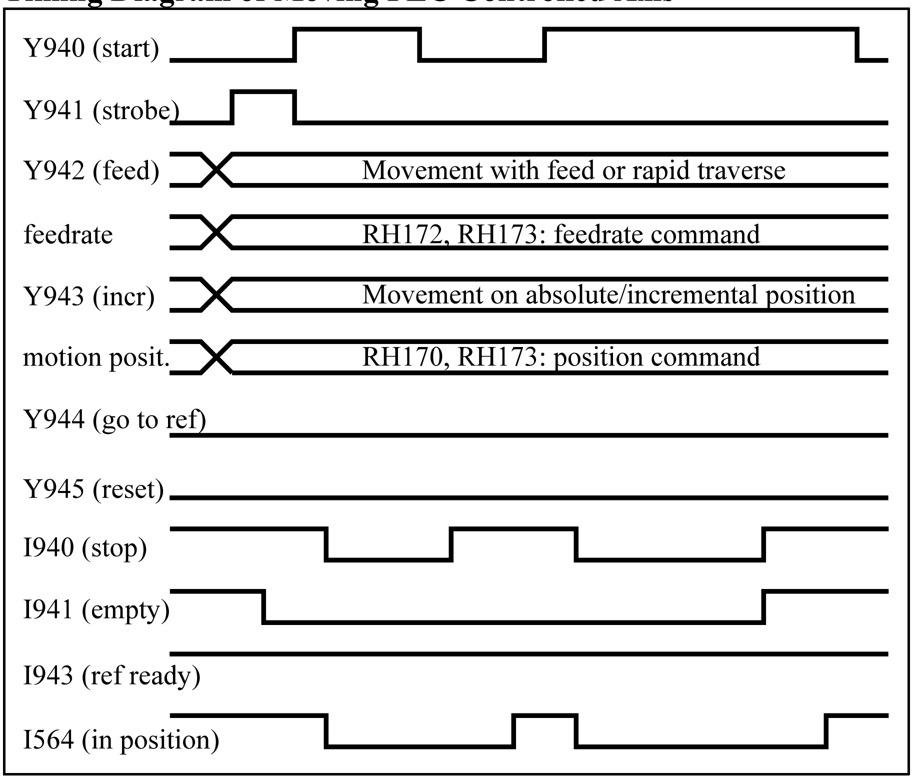
feedback of the brake has arived (I010=0).

Timing Diagram of Reference Point Return of PLC Controlled Axis



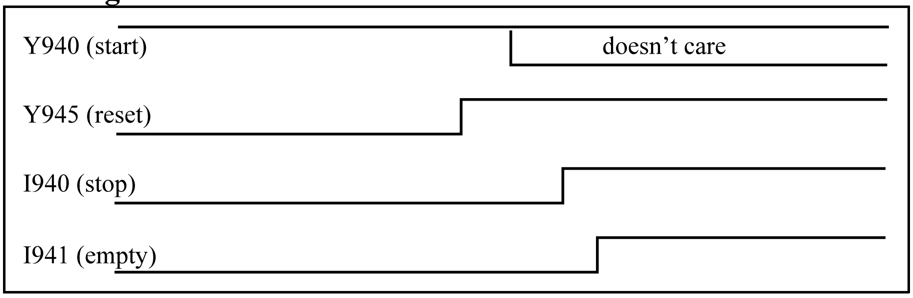
Reference point return on PLC controlled axis can be initiated by switching axis go to reference position flag (Y944 on the diagram) to 1 and switching start bit (Y940) on. The cycle has ended if the interpolator is stopped and empty on the given axis (I940=1, I941=1) and axis reference position ready signal has arrived (I943=1).

Timing Diagram of Moving PLC Controlled Axis



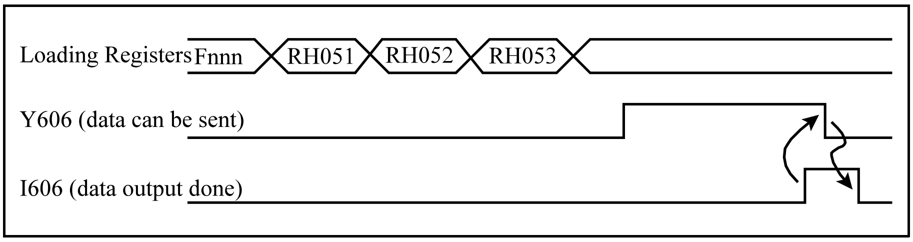
Before movement is started on PLC controlled axis the appropriate flags and registers must be set. In case of feed movement (Y942=1) the desired rate must be entered into registers RH172, RH173. It must be specified, whether the movement is to be done incrementally or absolutely (Y943) and the position registers (RH170, RH171) must be loaded according to this. Afterwards the strobe flag (Y941) must be switched on and the signaling of interpolator by means of setting empty interpolator flag (Y941) to 0, that the command has been transferred is awaited. Than the movement can be started by switching start flag (Y940=1). The movement can be stopped and restarted by switching start flag off and on. If stop and empty flags (I940=1, I941=1) are returned by the interpolator the start bit (Y940) can be switched off. The movement stops if axis in position flag I564 has arrived.

Reseting the Movement of PLC Controlled Axis



The pressing of RESET button on control has no effect on the PLC controlled axes. If the movement of PLC controlled axis is to be suspended reset flag (Y945 on the diagram) needs to be set. This way the interpolator stops after deceleration (I940=1) and switches interpolator empty flag (I941) on.

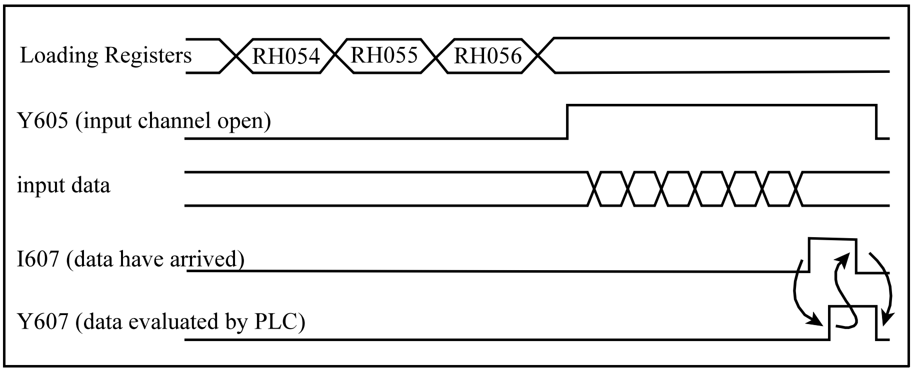
Timing Diagram of Data Output



After specifying data (F010 ... F499) and registers RH051, ..., RH053 flag Y606 is set to 1. After on input flag I606 feedback was detected flag Y606 is set to 0.

New output can be initiated after the NC has reset flag I606.

Timing Diagram of Data Input

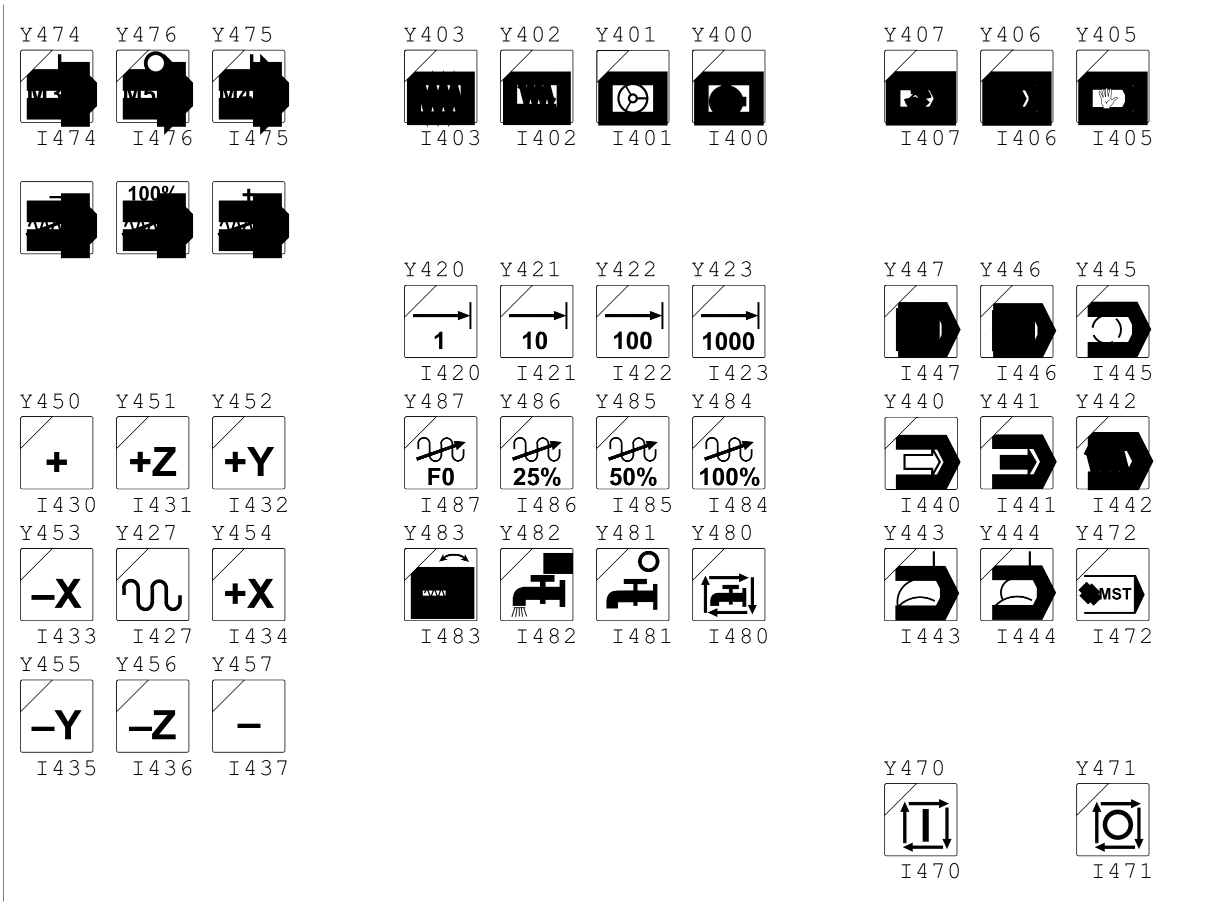


After specifying registers RH054, ..., RH056 input channel is enabled by the instruction U605. After input data have arrived the NC sets flag I607 to 1. After the PLC has evaluated data it gives out U607 instruction.

After it the NC resets flag I607 then the PLC resets Y607.

6.8 The Sample. plc Program

Below a PLC sample program is shown.
This PLC program covers a standard program capable of being the basic program of the PLC program of any machine.
Pushbuttons of machine control board 2 are applied in the sample program.
JOG direction and rapid traverse buttons are held down by START button, which is ceased by STOP button.
If in automatic mode handwheel is to be used the automatic mode button must be pressed and held down, menawhile manual handle mode button must be also pressed. In this case automatic and manual handle modes are simultaneously selected.
The sample program interprets tool replacement (T), spindle gear range change (M11-M18), S, spindle rotation (M3, M4, M5, M19), coolant (M8, M9), and program control code (M0, M1, M2, M30) functions.
Tool replacement and spindle gear range change need manual operation. The code of the tool or spindle range to be activated is displayed by the control than goes on when START is pressed.
Tool replacement can be initiated by programming address T.
In case of test, machine lock and function lock conditions the tool number taken from program is written into register RH064 without the tool replacement being initiated by the PLC for the sake of comfortable part program test. As test, machine lock or function lock condition is switched off the code of the current tool being in spindle appears in register RH064.
The sample program generates spindle stop and revolution signals from spindle encoder in PLC.
Spindle orientation (M19) is realised by closing position control loop.
No slide lubrication request is programmed in PLC.
Push-button arrangement of machine control board 2 applied by the PLC program is as follows:




```
/* SAMPLE.PLC PLC program with machine control board 2 */

/ *
input lines:

I000  -     no emergency stop

I002  -     machine power on line


I005  -     FEED - HOLD switch

I020  -     X ref position line
I021  -     Y ref position line
I022  -     Z ref position line
I023  -     4th ref position line


user's push-buttons in case of external handwheel

I450  -     X axis push-button
I451  -     Y axis push-button
I452  -     Z axis push-button
I453  -     4th axis push-button
I454  -     5th axis push-button
I455  -     6th axis push-button
I456  -
I457  -
```

```
I460  -     1 increment push-button
I461  -     10 increment push-button
I462  -     100 increment push-button
I463  -
I464  -     from NC
I465  -     external handwheel operates
I466  -
I467  -



JOG push-buttons in case of machine control board 2:

      jog (in case of vertical machine)

I430  -     +4th axis push-button
I431  -     +Z axis push-button
I432  -     +Y axis push-button
I433  -     -X axis push-button
I434  -     +X axis push-button
I435  -     -Y axis push-button
I436  -     -Z axis push-button
I437  -     -4th axis push-button


      optional push-buttons

I480  -     M8 auto push-button
I481  -     M9 push-button
I482  -     M8 push-button
I483  -     S jog push-button
I484  -     R100% push-button
I485  -     R50% push-button
I486  -     R25% push-button
I487  -     RF0% push-button



output lines

Y001  -     drive enabled
Y002  -     coolant on




output flags in case of machine control board 2:

      jog push-buttons (in case of vertical machine)

Y450  -     +4th axis active
Y451  -     +Z axis active
Y452  -     +Y axis active
Y453  -     -X axis active
Y454  -     +X axis active
Y455  -     -Y axis active
Y456  -     -Z axis active
Y457  -     -4th axis active

      optional push-buttons

Y480  -     M8 auto active
Y481  -     M9 active
Y482  -     M8 active
```

```
Y483   -      S jog active
Y484   -      R100% active
Y485   -      R50% active
Y486   -      R25% active
Y487   -      RFO% active
```

modules, labels:

```
:000   -
:001   -      20 msec rapid module
:002   -
:003   -      M code classification
:004   -      goto label in M code selection module
:005   -      preparing spindle stop
:006   -      resetting spindle rotation code
:007   -
:008   -
:009   -      operations before interruption of AUTO
:010   -      operations after return to AUTO
:011   -      function RESET
:012   -      start push-buttons RESET
:013   -      interface board RESET
:014   -      output flags RESET
:015   -      auxiliary module: if OP>0 then OP=1
:016   -      spindle rotation from push-buttons

:196   -      skip module of module :000
```

messageing M codes:

```
RH070 -      M8, M9 coolant state register
```

local flags:

```
F0100 -      mode change
F0101 -      JOG push-buttons enabled
F0102 -      interruption enabled
F0103 -      interruption enabling reset disabled
F0104 -      test emergency stop timer
F0105 -      evaluate MON on timer
F0106 -      previous state of AUTO mode (Y406)
F0107 -      external handwheel mode

F0110 -      test JOG push-buttons on START
F0111 -      initiate START state
F0112 -      initiate STOP state
F0113 -      initiate EMERGENCY STOP state
F0114 -      spindle started flag
F0115 -      spindle rotates
F0116 -      PLC suspended state
F0117 -      press M5 when suspending PLC

F0120 -      executable M code found
F0121 -      M3, M4 push-buttons:1, programmed:0
F0122 -      M5 push-button:1, programmed:0
F0123 -      saving coolant pump state
F0124 -
F0125 -      initiate M3 state
F0126 -      initiate M4 state
F0127 -      initiate M5 state
```

```
F0130 -     function stop
F0131 -     tool replacement execution enabled
F0132 -     tool preparation execution enabled
F0133 -     gear range change execution enabled
F0134 -     spindle revolution execution enabled
F0135 -     spindle rotation execution enabled
F0136 -
F0137 -


F0147 -     program controlling code execution enabled


F016  -     range code shadow register
            (Its value: 10, 11, ..., 18)
F018  -     rotation code shadow register
            (Its value: 3, 4, 5, 19)

F024  -     T code shadow register
F026  -     S code shadow register
F028  -     program controlling code shadow register
            (Its value: 0, 1, 2, 30)


F030  -     rotation code register saving area
F032  -     Q05 spindle rotation (M3, M4, M5, M19)
            phase counter saving area
F034  -
F036  -

F050  -     FIN counter saving register
F052  -     Q01 tool replacement (M06) phase counter saving register
F054  -     Q02 tool preparation (T) phase counter
            saving register
F056  -     Q03 gear range change (M10, M11, ..., M18)
            phase counter saving register
F058  -     Q04 spindle revolution (S) phase counter
            saving register
F060  -     Q05 spindle rotation (M3, M4, M5, M19) phase counter
            saving register
F062  -     Q06 coolant (M8, M9) phase counter saving register


F078  -     Q19 program controlling codes (M00, M01, M02, M30)
            phase counter saving register

F080  -     active tool number
F082  -     gained T code in case of test, machine lock, function lock

counters:

Q00   -     FIN counter
            =0 FIN signal transferable
            >0 its content is the number of functions to be executed

Q01   -     tool replacement (M06) phase counter
Q02   -     tool preparation (T) phase counter
Q03   -     gear range change (M10, M11, ..., M18) phase counter
Q04   -     spindle revolution (S) phase counter
Q05   -     spindle rotation (M3, M4, M5, M19) phase counter
Q06   -     coolant(M8, M9) phase counter
```

```
Q19    -      phase counter of program controlling codes (M00, M01, M02, M30)


              Interpretation of the content of the counter:
                    =0 function executed
                    =1,2,... execution times of functions


20 msec timers:

T00    -      emergency stop timer
T01    -      MON timer
T02    -      spindle revolution check timer


1 sec timers

H00    -      spindle revolution ready


PLC constants:


CONST39      -      rapid traverse override selection
                    if 0: from softkeys
                    if 1: from F% rotary switch 4 steps
                    if 2: from Machine control board 2 push-buttons
                    if 3: from F% rotary switch 13 steps, 1204 RAPOVER=0
                    if 4: from F% rotary switch 9 steps, 1204 RAPOVER=0


*/



/*SAMPLE.PLC */

/* :001 module start */


:001                ;20 msec cyclical PLC module


      /* INITIALIZATION */

I510                ;if first execution of module :001 after turn-on

      U521          ;axis selected
                    ;from NC
      U524          ;PLC push-buttons enabled from softkeys
      U532          ;selecting machine control board 2

      U407          ;start mode=EDIT
      U420          ;start increment=1
      U480          ;start spindle push-button=M8 auto
      LRP039        ;loading CONST39
 =2                 ;rapid traverse override from machine control board 2
      U484          ;start rapid override=100%
 Z

      UF0102        ;interruption enabled
      ,0            ;0 to OP
      SRH060        ;start spindle revolution=0
      SF080         ;start tool code=0
```

```
      ,5           ;5 to OP
      SRH062       ;start spindle rotation state: stopped
      ,11          ;11 to OP
      SRH063       ;start spindle range=11
      ,9           ;9 to OP
      SRH070       ;start coolant state: off

Z                  ;end of condition
                   ;first execution of module :001 after turn-on


      /* EMERGENCY STOP */

(V000ANI000)       ;if activating emergency stop
      UF0113       ;initiate EMERGENCY STOP state
Z                  ;end of condition;
                   ;activating emergency stop

(V540ANI540)       ;if MON output line is off
      UF0113       ;initiate EMERGENCY STOP state
Z                  ;MON output line is off


F0113              ;if initiate EMERGENCY STOP state

 Y001              ;if spindle enabled
      D651         ;orientation request off
      U654         ;1st spindle command signal direct output
      ,0           ;0 to OP
      SRH061       ;storing into spindle JOG command signal register
 Z                 ;spindle enabled

 (Y406             ;if AUTO mode active
 ANF0116)          ;and PLC not suspended
      C009         ;operations before interruption of AUTO
 E                 ;else
      C011         ;function RESET
 Z                 ;end of condition AUTO operation ...

      C012         ;start RESET
      ,50          ;50 to OP (1 sec lag)
      ST00         ;storing into emergency stop timer
      UF0104       ;test emergency stop timer
      DF0113       ;clearing initiate EMERGENCY STOP state

Z                  ;end of condition
                   ;initiate EMERGENCY STOP state


F0104              ;if initiate emergency stop timer
 T00               ;emergency stop timer testing
 E                 ;else, if terminated

      C013         ;interface board RESET
      C014         ;output flags RESET
      LY40         ;loading line Y40
      A.FF00       ;clearing bits Y400...Y407
      SY40         ;storing
      U407         ;activating EDIT mode
      DF0107       ;external handwheel mode off
      DF0104       ;evaluate emergency stop timer

 Z                 ;end of condition
                   ;inactivating lagged
```

```
Z                  ;test emergency stop timer


      /* handling MON output line */

(V002AI002) ;if MON input signal

 (NI542            ;if MON output line enabled
 ANY540            ;and MON off
 ANF0802)          ;and no erroneous parameter writing

      U540         ;activating MON output line
      UF0105       ;evaluate MON timer
      ,126         ;126 to OP (2.5 sec lag)
      ST01         ;initializing MON timer

 Z                 ;end of condition MON output line ...

Z                  ;end of condition MON input signal


F0105              ;if test MON timer

 T01               ;MON timer running

  I000             ;if no emergency stop
      DF0105       ;clearing evaluate MON timer
  Z                ;no emergency stop

 E                 ;else terminated
      D540         ;activating MON output line off
      DF0105       ;clearing test MON timer
 Z                 ;end of condition timer running

Z                  ;end of condition test MON timer


      /* handling RESET push-button */

(V477AI477) ;if RESET push-button selected

 (Y406             ;if AUTO mode active
 ANF0116           ;and PLC not suspended
 A(Y470            ;and or START state
 OY471))           ;or STOP state
      UF0117       ;press M5 when suspending PLC
      C009         ;operations before interruption of AUTO
      C012         ;start RESET
 E                 ;else
      C011         ;function RESET
      C012         ;start RESET
      UF0127       ;initiate M5 state
 Z                 ;end of condition AUTO mode active

      LI70         ;loading message word I70
 >0                ;if there is message on screen
      ONLY70       ;
      NSY70        ;clearing
                   ;message on screen (I700 - I717)
 Z                 ;end of condition there is message on screen

      LI72         ;loading message word I72
 >0                ;if there is message on screen
```

```
      ONLY72       ;
      NSY72        ;clearing
                   ;message on screen (I720 - I737)
 Z                 ;end of condition there is message on screen

      LI74         ;loading message word I74
 >0                ;if there is message on screen
      ONLY74       ;
      NSY74        ;clearing
                   ;message on screen (I740 - I757)
 Z                 ;end of condition there is message on screen


      LI76         ;loading message word I76
 >0                ;if there is message on screen
      ONLY76       ;
      NSY76        ;clearing
                   ;message on screen (I760 - I777)
 Z                 ;end of condition there is message on screen

Z                  ;end of condition RESET push-button selected


      /* handling USER'S push-buttons */


      /* MODE switches */

Y406               ;if AUTO mode active
      UF0106       ;previous state of AUTO mode (Y406) on
E                  ;else, if not on
      DF0106       ;previous state of AUTO mode (Y406) off
Z                  ;end of condition AUTO mode active


      /* MODE push-buttons */

(F0102             ;if interruption enabled
ANI552             ;and override is enabled
ANF0107)           ;and no external handwheel mode


 (V400AI400)       ;if REF mode selected
      LY40         ;loading line Y40
      A.FF00       ;clearing bits Y400...Y407
      SY40         ;storing
      U400         ;activating REF mode
      UF0100       ;mode change on
 Z                 ;end of condition REF mode selected

 (V401AI401)       ;if HNDL mode selected

  Y406             ;if AUTO mode active
   Y401            ;if HNDL mode active
      D401         ;inactivating HNDL mode in auto
   E               ;if HNDL mode inactive
    I406           ;if AUTO mode also selected
      U401         ;activating HNDL mode in auto
      D423         ;clearing 1000 increment
    E              ;else if AUTO not selected
      LY40         ;loading line Y40
      A.FF00       ;clearing bits Y400...Y407
      SY40         ;storing
      U401         ;activating HNDL mode
```

```
     D423         ;clearing 1000 increment
     UF0100       ;mode switch

   Z              ;end of condition AUTO mode also selected
  Z               ;end of condition HNDL mode active
 E                ;else, if not on
     LY40         ;loading line Y40
     A.FF00       ;clearing bits Y400...Y407
     SY40         ;storing
     U401         ;activating HNDL mode
     D423         ;clearing 1000 increment
     UF0100       ;mode change
 Z                ;end of condition AUTO mode active

Z                 ;end of condition HNDL mode selected

(V402AI402)       ;if INCR mode selected
     LY40         ;loading line Y40
     A.FF00       ;clearing bits Y400...Y407
     SY40         ;storing
     U402         ;activating INCR mode
     UF0100       ;mode change on
Z                 ;end of condition INCR mode selected

(V403AI403)       ;if JOG mode selected
     LY40         ;loading line Y40
     A.FF00       ;clearing bits Y400...Y407
     SY40         ;storing
     U403         ;activating JOG mode
     UF0100       ;mode change on
Z                 ;end of condition JOG mode selected

(V405AI405)       ;if MDI mode selected
     LY40         ;loading line Y40
     A.FF00       ;clearing bits Y400...Y407
     SY40         ;storing
     U405         ;activating MDI mode
     UF0100       ;mode change on
Z                 ;end of condition MDI mode selected

(V406AI406)       ;if AUTO mode selected
 NY406            ;if no auto operation
     LY40         ;loading line Y40
     A.FF00       ;clearing bits Y400...Y407
     SY40         ;storing
     U406         ;activating AUTO mode
     UF0100       ;mode change on
 Z
Z                 ;end of condition AUTO mode selected

(V407AI407)       ;if EDIT mode selected
     LY40         ;loading line Y40
     A.FF00       ;clearing bits Y400...Y407
     SY40         ;storing
     U407         ;activating EDIT mode
     UF0100       ;mode change on
Z                 ;end of condition EDIT mode selected

(Y403             ;if JOG operation
OY402             ;or INCR operation
OY401)            ;or HNDL operation

 (V483AI483)      ;if SPINDLE JOG selected
  NY483           ;if SPINDLE JOG inactive
```

```
      U483         ;SPINDLE JOG mode
      UF0127       ;initiate M5 state
   E               ;else
      D483         ;inactivating SPINDLE JOG
   Z               ;end of condition
                   ;SPINDLE JOG inactive
  Z                ;end of condition
                   ;SPINDLE JOG mode selected
 E                 ;if SPINDLE JOG not selected
      D483         ;inactivating SPINDLE JOG

 Z                 ;end of condition SPINDLE JOG selected


Z                  ;end of condition
                   ;interruption enabled and ...


      /* Operations after mode change */

F0100              ;if mode change on
      D470         ;inactivating START state and
      D471         ;STOP state
      DF0101       ;JOG push-buttons disabled
      LY42         ;loading line Y42
      A.007F       ;clearing JOG bits Y427,Y430,...,Y437
      SY42         ;storing
      D713         ;SPINDLE ROTATION REQUEST off

      LY40         ;loading line Y40
      A.00FF       ;clearing axis bits Y410...Y417
      SY40         ;storing
      LY44         ;loading line Y44
      A.00FF       ;clearing jog drive bits Y450...Y457
      SY44         ;storing


 (F0106            ;if OTHER mode selected
 ANY406)           ;from AUTO mode
  NF0116           ;if PLC not suspended
      C009         ;operations before interruption of AUTO
  Z                ;PLC not suspended
 Z                 ;end of condition
                   ;OTHER mode switched from AUTO mode

 (NF0106AY406)     ;if AUTO mode switched
                   ;from OTHER mode
      C011         ;function RESET
 Z                 ;end of condition
                   ;AUTO mode switched from OTHER mode

      DF0100       ;mode change off

Z                  ;end of condition mode change on


      /* External handwheel */

Y401               ;if manual handle mode selected

 NI465             ;if no external handwheel

      DF0107       ;no external handwheel mode
```

```
 (I433OI434)      ;if JOG-X, or JOG+X axis selected
     LY40         ;loading line Y40
     A.00FF       ;clearing axis bits Y410...Y417
     SY40         ;storing
     LY44         ;loading line Y44
     A.00FF       ;clearing bits Y450...Y457
     SY44         ;storing
     U410         ;activating
                  ;1st axis
     U453         ;activating -X on control board 2
     U454         ;activating +X on control board 2
 Z                ;

 (I435OI432)      ;if JOG-Y, or JOG+Y axis selected
     LY40         ;loading line Y40
     A.00FF       ;clearing axis bits Y410...Y417
     SY40         ;storing
     LY44         ;loading line Y44
     A.00FF       ;clearing bits Y450...Y457
     SY44         ;storing
     U410         ;activating
                  ;2nd axis
     U452         ;activating -Y on control board 2
     U455         ;activating +Y on control board 2
 Z                ;

 (I436OI431)      ;if JOG-Z, or JOG+Z axis selected
     LY40         ;loading line Y40
      A.00FF      ;clearing axis bits Y410...Y417
     SY40         ;storing
     LY44         ;loading line Y44
     A.00FF       ;clearing bits Y450...Y457
     SY44         ;storing
     U410         ;activating
                  ;3rd axis
     U451         ;activating -Z on control board 2
     U456         ;activating +Z on control board 2
 Z                ;

 (I430OI437)      ;if JOG-4, or JOG+4 axis selected
     LY40         ;loading line Y40
      A.00FF      ;clearing axis bits Y410...Y417
     SY40         ;storing
     LY44         ;loading line Y44
     A.00FF       ;clearing bits Y450...Y457
     SY44         ;storing
     U410         ;activating
                  ;4th axis
     U450         ;activating -4 on control board 2
     U457         ;activating +4 on control board 2
 Z

E                 ;else, if external handwheel
     LI46         ;loading word I46 I47
     A.00FF       ;clearing byte I470
 >32              ;if increment push-button byte
                  ;not in transitional state
  I464            ;if push-button state from NC
     DF0107       ;no external handwheel mode
     ,0           ;0 to OP
     SY41         ;inactivating increments and axes
                  ;in NC state of push-button, in order
                  ;not to move, for there is already
                  ;manual handle mode for the NC
```

```
   E             ;else manual handle
      UF0107     ;activating external handwheel mode
      LI45       ;loading user's push-buttons
      A.07FF     ;inactivating increments and axes
      SY41       ;storing
                 ;axis and increment lamp
   Z             ;end of condition push-button state from NC
  Z              ;end of condition increment push-button
                 ;is not in transitional state
 Z               ;end of condition no external handwheel

Z                ;end of condition
                 ;manual handle mode selected


      /* handling AXIS push-buttons */

NF0107           ;if no external handwheel mode

 (V410AI410)     ;if 1st axis
                 ;selected
      LY40       ;loading line Y40
      A.00FF     ;clearing bits Y410...Y417
      SY40       ;storing
      U410       ;activating
                 ;1st axis
 Z               ;end of condition
                 ;1st axis selected

 (V411AI411)     ;if 2nd axis
                 ;selected
      LY40       ;loading line Y40
      A.00FF     ;clearing bits Y410...Y417
      SY40       ;storing
      U411       ;activating
                 ;2nd axis
 Z               ;end of condition
                 ;2nd axis selected

 (V412AI412)     ;if 3rd axis
                 ;selected
      LY40       ;loading line Y40
      A.00FF     ;clearing bits Y410...Y417
      SY40       ;storing
      U412       ;activating
                 ;3rd axis
 Z               ;end of condition
                 ;3rd axis selected

 (V413AI413)     ;if 4th axis
                 ;selected
      LY40       ;loading line Y40
      A.00FF     ;clearing bits Y410...Y417
      SY40       ;storing
      U413       ;activating
                 ;4th axis selected
 Z               ;end of condition
                 ;4th axis mode selected

 (V414AI414)     ;if 5th axis
                 ;selected
      LY40       ;loading line Y40
      A.00FF     ;clearing bits Y410...Y417
      SY40       ;storing
```

```
     U414         ;activating
                  ;5th axis
Z                 ;end of condition
                  ;5th axis selected

(V415AI415)       ;if 6th axis
                  ;selected
     LY40         ;loading line Y40
     A.00FF       ;clearing bits Y410...Y417
     SY40         ;storing
     U415         ;activating
                  ;6th axis
Z                 ;end of condition
                  ;6th axis selected

(V416AI416)       ;if 7th axis
                  ;selected
     LY40         ;loading line Y40
     A.00FF       ;clearing bits Y410...Y417
     SY40         ;storing
     U416         ;activating
                  ;7th axis
Z                 ;end of condition
                  ;7th axis selected

(V417AI417)       ;if 8th axis
                  ;selected
     LY40         ;loading line Y40
     A.00FF       ;clearing bits Y410...Y417
     SY40         ;storing
     U417         ;activating
                  ;8th axis
Z                 ;end of condition
                  ;8th axis selected


     /* handling INCREMENT push-buttons */

(V420AI420)       ;if 1 increment selected
     LY42         ;loading line Y42
     A.FF00       ;clearing bits Y420...Y427
     SY42         ;storing
     U420         ;activating 1 increment
Z                 ;end of condition
                  ;1 increment selected

(V421AI421)       ;if 10 increment selected
     LY42         ;loading line Y42
     A.FF00       ;clearing bits Y420...Y427
     SY42         ;storing
     U421         ;activating 10 increment
Z                 ;end of condition
                  ;10 increment selected

(V422AI422)       ;if 100 increment selected
     LY42         ;loading line Y42
     A.FF00       ;clearing bits Y420...Y427
     SY42         ;storing
     U422         ;activating 100 increment
Z                 ;end of condition
                  ;100 increment selected

NY401             ;if no manual handle mode
 (V423AI423)      ;if 1000 increment selected
```

```
      LY42         ;loading line Y42
      A.FF00       ;clearing bits Y420...Y427
      SY42         ;storing
      U423         ;activating 1000 increment
 Z                 ;end of condition
                   ;1000 increment selected
 Z                 ;end of condition no manual handle mode


Z                  ;end of condition
                   ;no external handwheel mode


      /* handling push-buttons of CONDITIONS */

(NI546O            ;if no executable block
                   ;in buffer or
(Y447A             ;special block and
Y547A              ;FIN and
I551A              ;interpolator empty and
NI552))            ;override enabled

 (V440AI440)       ;if TEST selected
  NY440            ;if TEST state inactive
      U440         ;activating TEST state
  E                ;else
      D440         ;inactivating TEST state
  Z                ;end of condition TEST state inactive
 Z                 ;end of condition TEST selected

 (V441AI441)       ;if MCH.LK selected
  NY441            ;if MCH.LK state inactive
      U441         ;activating MCH.LK state
  E                ;else
      D441         ;inactivating MCH.LK state
  Z                ;end of condition MCH.LK state inactive
 Z                 ;end of condition MCH.LK selected

 (V472AI472)       ;if FUNCT LK selected
      NLY472       ;inverse load of FUNCT LK active
      SY472        ;enter FUNKC ZAR active
 Z                 ;end of condition FUNCT LK selected

Z                  ;end of condition
                   ;no executable block...

(V442AI442)        ;if DRY RN selected
 NY442             ;if DRY RN state inactive
      U442         ;activating DRY RN state
 E                 ;else
      D442         ;inactivating DRY RN state
 Z                 ;end of condition DRY RN state inactive
Z                  ;end of condition DRY RN selected

(V443AI443)        ;if BK.RST selected
 (NY443            ;if BK.RST state inactive
 AI511)            ;and HOLD state
      U443         ;activating BK.RST state
      D444         ;inactivating BK.RET state
 E                 ;else
      D443         ;inactivating BK.RST state
 Z                 ;end of condition BK.RST state inactive
Z                  ;end of condition BK.RST selected
```

```
(V444AI444)        ;if BK.RET selected
 (NY444            ;if BK.RET state inactive
 AI511)            ;and HOLD state
      U444         ;activating BK.RET state
      D443         ;inactivating BK.RST state
 E                 ;else
      D444         ;inactivating BK.RET state
 Z                 ;end of condition BK.RET state inactive
Z                  ;end of condition BK.RET selected

(V445AI445)        ;if CND.SP selected
 NY445             ;if CND.SP state inactive
      U445         ;activating CND.SP
 E                 ;else
      D445         ;inactivating CND.SP state
 Z                 ;end of condition CND.SP state inactive
Z                  ;end of condition CND.SP selected

(V446AI446)        ;if CND.BK 1 selected
 NY446             ;if CND.BK 1 state inactive
      U446         ;activating CND.BK 1
 E                 ;else
      D446         ;inactivating CND.BK 1 state
 Z                 ;end of condition CND.BK 1 state inactive
Z                  ;end of condition CND.BK 1 selected

(V447AI447)        ;if SGL.BK selected
 NY447             ;if SGL.BK state inactive
      U447         ;activating SGL.BK
 E                 ;else
      D447         ;inactivating SGL.BK state
 Z                 ;end of condition SGL.BK state inactive
Z                  ;end of condition SGL.BK selected


      /* handling JOG push-buttons */

(I000              ;if no emergency state
AI540)             ;and MON on


 I427              ;if JOG rapid traverse selected
      U427         ;activating JOG rapid traverse
 E                 ;else
  NF0101           ;JOG push-buttons are disabled
      D427         ;inactivating JOG rapid traverse
  Z                ;end of condition
                   ;JOG push-buttons disabled
 Z                 ;end of condition
                   ;JOG rapid traverse selected


 (Y400             ;if activating REF
 OY402             ;or INCR
 OY403)            ;or JOG mode

  I433             ;if JOG 4th axis selected
      U434         ;activating JOG X- on control board 2
      U453         ;activating 4th axis
      D430         ;inactivating JOG X+ on control board 2
      D454         ;inactivating 5th axis
  E                ;else
   NF0101          ;JOG push-buttons disabled
      D434         ;inactivating JOG X- on control board 2
```

```
    D453        ;inactivating 4th axis
 Z              ;end of condition
                ;JOG push-buttons disabled
Z               ;;end of condition JOG 4th axis selected

I434            ;if JOG 5th axis selected
    U430        ;activating JOG X+ on control board 2
    U454        ;activating 5th axis
    D434        ;inactivating JOG X- on control board 2
    D453        ;inactivating 4th axis
E               ;else
 NF0101         ;JOG push-buttons disabled
    D430        ;inactivating JOG X+ on control board 2
    D454        ;inactivating 5th axis
 Z              ;end of condition
                ;JOG push-buttons disabled
Z               ;end of condition JOG 5th axis selected

I435            ;if JOG 6th axis selected
    U435        ;activating JOG Y- on control board 2
    U455        ;activating 6th axis
    D431        ;inactivating JOG Y+ on control board 2
    D452        ;inactivating 3rd axis
E               ;else
 NF0101         ;JOG push-buttons disabled
    D435        ;inactivating JOG Y- on control board 2
    D455        ;inactivating 6th axis
 Z              ;end of condition
                ;JOG push-buttons disabled
Z               ;end of condition JOG 6th axis selected

I432            ;if JOG 3rd axis selected
    U431        ;activating JOG Y+ on control board 2
    U452        ;activating JOG 3rd axis
    D435        ;inactivating JOG Y- on control board 2
    D455        ;inactivating JOG 6th axis
E               ;else
 NF0101         ;JOG push-buttons disabled
    D431        ;inactivating JOG Y+ on control board 2
    D452        ;inactivating 3rd axis
 Z              ;end of condition
                ;JOG push-buttons disabled
Z               ;end of condition JOG 3rd axis selected

I436            ;if JOG 7th axis selected
    U436        ;activating JOG Z- on control board 2
    U456        ;activating 7th axis
    D432        ;inactivating JOG Z+ on control board 2
    D451        ;inactivating 2nd axis
E               ;else
 NF0101         ;JOG push-buttons disabled
    D436        ;inactivating JOG Z- on control board 2
    D456        ;inactivating 7th axis
 Z              ;end of condition
                ;JOG push-buttons disabled
Z               ;end of condition JOG 7th axis selected

I431            ;if JOG 2nd axis selected
    U432        ;activating JOG Z+ on control board 2
    U451        ;activating 2nd axis
    D436        ;inactivating JOG Z- on control board 2
    D456        ;inactivating 7th axis
E               ;else
 NF0101         ;JOG push-buttons disabled
```

```
      D432        ;inactivating JOG Z+ on control board 2
      D451        ;inactivating 2nd axis
   Z              ;end of condition
                  ;JOG push-buttons disabled
  Z               ;end of condition JOG 2nd axis selected


  I437            ;if JOG 8th axis selected
      U437        ;activating 8th axis
      U413        ;activating 4th axis
      D433        ;inactivating JOG + on control board 2
      D450        ;inactivating 1st axis
  E               ;else
   NF0101         ;JOG push-buttons disabled
      D437        ;inactivating JOG - on control board 2
      D457        ;inactivating 8th axis

   Z              ;end of condition
                  ;JOG push-buttons disabled
  Z               ;end of condition JOG 8th axis selected

  I430            ;if JOG 1st axis selected
      U433        ;activating JOG + on control board 2
      U450        ;activating 1st axis
      U413        ;activating 4th axis
      D437        ;inactivating JOG - on control board 2
      D457        ;activating 8th axis
  E               ;else
   NF0101         ;JOG push-buttons disabled
      D433        ;inactivating JOG + on control board 2
      D450        ;inactivating 1st axis

   Z              ;end of condition
                  ;JOG push-buttons disabled
  Z               ;end of condition JOG 1st axis selected

 Z                ;end of condition
                  ;activating REF or INCR or JOG mode

Z                 ;end of condition no emergency state
                  ;and MON on


      /* handling OVERRIDE push-buttons */

      LRP039      ;selecting rapid traverse override
                  ;at parameter CONST20
=0                ;if 0: from NC keyboard
      U525        ;R% from NC keyboard
      LRH039      ;loading input register R%
E                 ;else

 =1               ;from F% override push-button
      D525        ;R% not from NC keyboard
      LRH028      ;loading input register F%
  <4              ;if F%<10%
      ,0          ;R%=F0
  E                     ;else
   <7             ;if 5%<F%<40%
      ,1          ;R%=25%
   E              ;else
    <10           ;if 40%<F%<70%
      ,2          ;R%=50%
    E             ;else, if 70%<F%
```

```
      ,3           ;R%=100%
    Z              ;end of condition 40%<F%<70%
   Z               ;end of condition 5%<F%<40%
  Z                ;end of condition F%<10%
 E                 ;
  =2               ;push-buttons from machine control board 2
   (V487AI487)     ;if RF0
                   ;selected
      LY48         ;loading line Y48
      A.FF0F       ;clearing bits Y484...Y487
      SY48         ;storing
      U487         ;activating
                   ;RF0
   Z               ;end of condition
                   ;RF0 selected
   (V486AI486)     ;if R25%
                   ;selected
      LY48         ;loading line Y48
      A.FF0F       ;clearing bits Y484...Y487
      SY48         ;storing
      U486         ;activating
                   ;R25%
   Z               ;end of condition
                   ;R25% selected
   (V485AI485)     ;if R50
                   ;selected
      LY48         ;loading line Y48
      A.FF0F       ;clearing bits Y484...Y487
      SY48         ;storing
      U485         ;activating
                   ;R50%
   Z               ;end of condition
                   ;R50% selected
   (V484AI484)     ;if R100%
                   ;selected
      LY48         ;loading line Y48
      A.FF0F       ;clearing bits Y484...Y487
      SY48         ;storing
      U484         ;activating
                   ;R100%
   Z               ;end of condition
                   ;R100% selected

   Y487            ;if RF0 active
      ,0           ;R% code=0
   Z               ;end of condition RF0 active

   Y486            ;if R25% active
      ,1           ;R% code=1
   Z               ;end of condition R25% active

   Y485            ;if R50% active
      ,2           ;R% code=2
   Z               ;end of condition R50% active

   Y484            ;if R100% active
      ,3           ;R% code=3
   Z               ;end of condition R100% active
  E                ;
   =3              ;if feedrate override affects
      LRH028       ;loading input register F%
   E               ;else not affects
      LRH028       ;loading input register F%
    >8
```

```
      ,13         ;100%
     Z            ;Z of >8
     =8
      ,11         ;80%
     Z
     =7
      ,9          ;60%
     Z
     =6
      ,7          ;40%
     Z
    Z             ;Z of =3
   Z              ;Z of =2
  Z               ;Z of =1
 Z                ;Z of =0

      SRH089      ;storing into output register R%
      LRH028      ;loading input register F%
      SRH078      ;storing into output register F%
      LRH029      ;loading input register S%
      SRH079      ;storing into output register S%



      /* Handling START push-button */


(I000             ;if no emergency state
AI540)            ;and MON on

 (V470AI470)      ;if START mode selected

  NY470           ;if START state inactive

   Y400           ;if REF mode active
      UF0101      ;JOG push-buttons enabled
      UF0111      ;initiate START state
   Z              ;end of condition REF mode active

   (Y401          ;if HNDL mode active
   OY402)         ;or INCR mode active

    (I546         ;if executable block in buffer
    ONY547        ;or FIN inactive
    ONI551)       ;or interpolator not empty

      UF0111      ;initiate START state
    Z             ;end of condition
                  ;executable block ...

   Z              ;end of condition
                  ;HNDL or INCR mode active


   Y403           ;if JOG mode active

    (I546         ;if executable block in buffer
    ONY547        ;or FIN inactive
    ONI551)       ;or interpolator not empty
      UF0111      ;initiate START state
    E             ;else
      UF0110      ;test JOG push-buttons for START
    Z             ;end of condition
```

```
                  ;executable block in buffer

   Z              ;end of condition JOG mode active


   (Y405          ;if MDI mode active
   OY406)         ;or AUTO mode active

    (I546         ;if executable block in buffer
    ONY547        ;or FIN inactive
    ONI551)       ;or interpolator not empty

      UF0111      ;initiate START state
    Z             ;end of condition
                  ;executable block in buffer
   Z              ;end of condition
                  ;MDI or AUTO mode active

   I545           ;if G28 active
      UF0111      ;initiate START state
   Z              ;end of condition G28 active


   F0130          ;if initiate FUNCTION STOP
      UF0111      ;initiate START state
      DF0130      ;clearing FUNCTION STOP
   Z              ;end of condition initiate FUNCTION STOP

  Z               ;end of condition START state inactive
 Z                ;end of condition START mode selected

Z                 ;end of condition no emergency state
                  ;and MON on


      /* Enabling jog push-buttons */

F0110             ;if test JOG push-buttons
                  ;for START
      LY42        ;loading line Y42
      A.FF00      ;clearing bits Y42n
 >0               ;one of JOG push-buttons on
      UF0111      ;initiate START state
      UF0101      ;JOG push-buttons enabled
 Z                ;end of condition
                  ;one of JOG push-buttons on
      DF0110      ;clearing test JOG push-buttons
                  ;for START
Z                 ;end of condition
                  ;test JOG push-buttons


      /* Creating START state at flag */

F0111             ;if initiate START state

      U470        ;activating START state
      D471        ;inactivating STOP state
      DF0111      ;clearing initiate START state

Z                 ;end of condition
                  ;initiate START state
```

```
      /* Handling STOP push-buttons */

(V471AI471)         ;if STOP selected
      UF0112        ;initiate STOP state
Z                   ;end of condition STOP selected


      /* STOP state from  NC */

I547                ;if NC asks for STOP state
      UF0112        ;initiate STOP state
Z                   ;NC switched on in STOP state now


      /* Creating STOP state at flag */

F0112               ;if initiate STOP state

 (NI552             ;if override and STOP is disabled
 OI555)             ;or G76, G78
      D470          ;inactivating START state
      U471          ;activating STOP state
  F0101             ;if JOG push-buttons enabled
      DF0101        ;clearing JOG push-buttons enabled
      D471          ;inactivating STOP state
      LY42          ;loading line Y42
      A.007F        ;clearing JOG bits Y427,Y430,...,Y437
      SY42          ;storing
  Z                 ;end of condition JOG push-buttons enabled
 Z                  ;Z of override and STOP

      DF0112        ;clearing initiate STOP state
Z                   ;end of condition initiate STOP state


      /* INTD state after STOP */

(I555               ;if thread cutting cycle
AY471               ;and STOP state
AY406)              ;and AUTO mode

  NF0116            ;if PLC is not interrupted
      C009          ;activity after interrupting AUTO
  Z                 ;PLC not interrupted

Z                   ;Z of thread cutting cycle



      /* Handling spindle rotating push-buttons */

(I000               ;if no emergency state
AI540)              ;and MON on
 (                  ;filtering start
 (F0131             ;if tool replacement execution enabled
 ANF0102)           ;and interruption disabled (process M6)
 O                  ;or
 (F0132             ;if tool preparation execution enabled
 ANF0102)           ;and interruption disabled (process T)
 O                  ;or
 (F0133             ;if gear range change execution enabled
 ANF0102)           ;and interruption disabled (process M11, ..., M18)
 O                  ;or
 (F0147             ;if program controlling code execution enabled
```

```
 ANF0102)          ;and interruption disabled (process M0, ..., M30)
 )                 ;push-button disabled
 E                 ;else either S or M3, ... M19 under execution

  (V476AI476)      ;if M5 selected on control board 2
      UF0127       ;initiate M5 state
  Z                ;end of condition M5 selected on control board 2

  (NY483           ;if no spindle JOG
  ANY440           ;and no test
  ANY441           ;and no machine lock
  ANY472)          ;and no function lock

   (V474AI474)     ;if M3 selected on control board 2
      UF0125       ;initiate M3 state
   Z               ;end of condition M3 selected on control board 2
   (V475AI475)     ;if M4 selected on control board 2
      UF0126       ;initiate M4 state
   Z               ;end of condition M4 selected on control board 2
  Z                ;end of condition no spindle JOG, ...

 Z                 ;end of condition filtering

Z                  ;end of condition no emergency state ...



(NI000             ;if emergency state
ONI540)            ;or MON off
      DF0125       ;clearing initiate spindle start M3
      DF0126       ;clearing initiate spindle start M4
      DF0127       ;clearing initiate spindle stop M5
Z                  ;


(F0121             ;if M3, M4 from control board
OF0122)            ;or M5 from control board

      LQ04         ;loading S phase counter to OP
 =2                ;if waiting for N=Ns exit
      DQ00         ;decrementing FIN counter
      UF0102       ;interruption enabled
      ,0           ;loading 0 to OP
      SQ04         ;storing into phase counter
 Z                 ;end of condition waiting for N=Ns

      LQ05         ;M3,M4,M5,M19 phase counter to OP
 =0                ;if finished
      DF0135       ;spindle rotation execution disabled
      LF030        ;loading rotation code save
      SF018        ;storing into rotation code shadow register
      LF032        ;loading Q05 spindle rotation
                   ;(M3, M4, M5, M19)
                   ;phase counter save
   >1              ;if greater than 1
                   ;M3, M4 processing
      DQ00         ;FIN decrements
      ,0           ;resetting phase number
   Z               ;end of condition greater than 1
      SQ05         ;storing into M3,M4,M5,M19 phase counter
   F0121           ;if M3, M4 processing from control board
      DF0121       ;inactivating M3, M4 from control board
   Z               ;end of condition
                   ;M3, M4 processing from control board
```

```
   F0122          ;if M5 processing from control board
      DF0122      ;inactivating M5 from control board
   Z              ;end of condition
                  ;M5 processing from control board
 Z                ;end of condition finished

Z                 ;end of condition M3, M4, M5 from control board


      /* M3, M4 start at flag */

((F0125           ;if requesting spindle start M3,
OF0126)           ;or M4 push-buttons
ANF0122)          ;and end of M5 from push-buttons

 (NY710           ;if no SPINDLE REVOLUTION ERROR
 ANY711)          ;and no SPINDLE RISING/FALLING EDGE

  (NI546          ;if executable block in buffer
  ONY470          ;or no START state
  OF0121          ;or manual start processing
  OY713)          ;or if message SPINDLE ROTATION REQUEST

      C016        ;spindle rotation from push-buttons
      UF0121      ;activating M3, M4 from push-button

  Z               ;end of condition no spindle rotation
 Z                ;end of condition no spindle error

      DF0125      ;clearing initiate M3 state
      DF0126      ;clearing initiate M4 state

Z                 ;end of condition requesting spindle start


      /* Spindle stop M5 at flag */

(F0127            ;if spindle stop M5 request
ANF0122)          ;and end of M5 from push-button

 Y652             ;if spindle command signal output enabled
      C016        ;spindle rotation from push-buttons
      UF0122      ;setting flag from M5 push-button
 Z                ;spindle command signal enabled
      DF0127      ;clearing initiate M5 state

Z                 ;end of condition
                  ;spindle stop M5 request
```

```
      /* Handling SPINDLE JOG */

(Y483                 ;if SPINDLE JOG active
ANF0122)              ;and M5 not selected
 (I474                ;if M3
 OI475)               ;or M4 selected on control board 2
      U001            ;drive enabled
      U652            ;1st spindle command signal output enabled
      U654            ;1st spindle command signal direct output
  I475                ;if M4 selected (CCW)
      D474            ;inactivating M3 on control board 2
      U475            ;activating M4 on control board 2
      D476            ;inactivating M5 on control board 2
      .007F           ;positive number to OP
  E                   ;else M3 selected
      U474            ;activating M3 on control board 2
      D475            ;inactivating M4 on control board 2
      D476            ;inactivating M5 on control board 2
      .FF80           ;negative number to OP
  Z                   ;end of condition M4 direction
      SRH061          ;storing into spindle JOG command signal register
 E                    ;else if M3 or M4
                      ;not selected on control board 2
      D474            ;inactivating M3 on control board 2
      D475            ;inactivating M4 on control board 2
      U476            ;activating M5 on control board 2
      D001            ;inactivating spindle drive
      D652            ;inactivating 1st spindle command signal output
      U654            ;activating 1st spindle command signal direct output
      ,0              ;0 to OP
      SRH061          ;storing into SPINDLE JOG command signal register
 Z                    ;end of condition
                      ;4th or 5th JOG selected
Z                     ;end of condition
                      ;SPINDLE JOG active and M5 not selected


      /* Handling COOLANT */

(I000                 ;if no emergency state
AI540)                ;and MON on
 (                    ;filtering start
 (F0131               ;if tool replacement execution enabled
 ANF0102)             ;and interruption disabled (process M6)
 O                    ;or,
 (F0132               ;if tool preparation execution enabled
 ANF0102)             ;and interruption disabled (process T)
 O                    ;or,
 (F0133               ;if gear range change execution enabled
 ANF0102)             ;and interruption disabled (process M11, ..., M18)
 O                    ;or,
 (F0147               ;if program controlling code execution enabled
 ANF0102)             ;and interruption disabled (process M0, ..., M30)
 )                    ;push-button disabled
 E                    ;else either S or M3, ... M19 under execution

  (V480AI480)         ;if M8 auto selected on control board 2
   Y480               ;if M8 auto active
      D480            ;inactivating M8 auto on control board 2
   E                  ;else
      U480            ;activating M8 auto on control board 2
   Z                  ;end of condition M8 auto active
  Z                   ;end of condition M8 auto selected on control board 2
```

```
  NY480           ;if coolant handling from push-buttons

   (V482AI482)    ;if M8 selected on control board 2
      U002        ;coolant pump on
   Z              ;end of condition M8 selected on control board 2

   (V481AI481)    ;if M9 selected on control board 2
      D002        ;coolant pump off
   Z              ;end of condition M9 selected on control board 2

  Z               ;end of condition
                  ;coolant handling from push-buttons

  Y480            ;if automatic coolant handling
      LRH070      ;programmed M8/M9 state
   =8             ;if M8 programmed
      U002        ;coolant pump on
   E              ;else
      D002        ;coolant pump off
   Z              ;end of condition M8 programmed

  Z               ;end of condition
                  ;automatic coolant handling

 Z                ;end of condition
                  ;no M06, T, M11, M30 under execution
Z                 ;end of condition no emergency and...

Y002              ;if coolant pump on
      U482        ;activating M8 on control board 2
      D481        ;inactivating M9 on control board 2
E                 ;else
      D482        ;inactivating M8 on control board 2
      U481        ;activating M9 on control board 2
Z                 ;end of condition
                  ;coolant pump on


      /* SUPERVISION */

/* reference point return and limit test */

(Y400             ;if REF mode active,
OI545)                  ;or G28

      LI020       ;REFX line
      SY550       ;1st axis reference position ready

      LI021       ;REFY line
      SY551       ;2nd axis reference position ready

      LI022       ;REFZ line
      SY552       ;3rd axis reference position ready

      LI023       ;REF4 line
      SY553       ;4th axis reference position ready

E                 ;else limit test

Z                 ;end of condition
                  ;REF mode active, or G28


/* spindle revolution check */
```

```
(F0114              ;if spindle started
ANF0134             ;and no command S under execution
ANF0135             ;and no spindle rotation under execution
AI650)              ;and command signal edge
 NI655              ;if no spindle fluctuation
      UF0115        ;spindle rotation
 E                  ;spindle fluctuation
      DF0115        ;no spindle rotation
      U710          ;SPINDLE REVOLUTION ERROR on
 Z                  ;end of condition no spindle fluctuation
Z                   ;end of condition spindle started ...

I657                ;if N=0
      DF0115        ;no spindle rotation
Z                   ;end of condition N=0


/* handling spindle rotation output flag */

(Y441               ;if MCH.LK state,
OY472               ;or function lock state
OY440)              ;or TEST state active
      U650          ;spindle rotates
E                   ;else, if none
      LF0115        ;clearing spindle rotation flag
      SY650         ;storing into spindle rotation output
Z                   ;end of condition
                    ;MCH.LK or function lock


/* process in case of spindle revolution error */

(F0114              ;if spindle started
ANF0134             ;and no command S under execution
ANF0135             ;and no spindle rotation under execution
AY710)              ;and SPINDLE REVOLUTION ERROR
      UF0127        ;initiate M5 state
Z                   ;end of condition SPINDLE REVOLUTION ERROR


/* initiating FEED HOLD */

(I005               ;if FEED HOLD line on
OF0104)             ;if test EMG timer
      U542          ;activating FEED HOLD state
E                   ;else, deceleration
      D542          ;inactivating FEED HOLD state
Z                   ;end of condition FEED HOLD line on


/* spindle stop in case of FEED HOLD and disabled override state */

(Y542               ;if FEED HOLD state active
AI552               ;and override disabled
AF0114              ;and spindle on
ANF0135)            ;and no spindle rotation under execution
      UF0127        ;initiating M5 state
Z                   ;end of condition FEED HOLD ...


/* push-buttons in case of HOLD state */

(I511AV511)         ;if FEED HOLD selected
      C011          ;function RESET
```

```
 F0117           ;if select M5 when suspending PLC
      UF0127     ;initiate M5 state
 Z
      DF0117     ;do not select M5 when suspending PLC
      UF0116     ;PLC suspended
      D443       ;inactivating BK.RST state
      D444       ;inactivating BK.RET state
Z                ;end of condition FEED HOLD selected


/* push-buttons in case of clearing HOLD state */

(NI511AV511)     ;if HOLD state cleared now

      DF0116     ;PLC not suspended
 (Y406           ;if AUTO mode active
 AY470           ;and START state
 ANY443)         ;if not BK.RST state
      C010       ;operations after return to AUTO
 Z               ;end of condition if AUTO mode ...

Z                ;end of condition
                 ;HOLD state cleared now


      /* receiving functions */

(NY441           ;if no machine lock state
ANY472           ;and no function lock state
ANY440)          ;and no TEST state

 I520            ;1st M function sent
      DF0120     ;no executable M code found
      LRH000     ;code of 1st M function
      C003       ;M code classification
 Z               ;end of condition 1st M function sent

 I521            ;2nd M function sent
      DF0120     ;no executable M code found
      LRH001     ;code of 2nd M function
      C003       ;M code classification
 Z               ;end of condition 2nd M function sent

 I522            ;3rd M function sent
      DF0120     ;no executable M code found
      LRH002     ;code of 3rd M function
      C003       ;M code classification
 Z               ;end of condition 3rd M function sent

 I523            ;4th M function sent
      DF0120     ;no executable M code found
      LRH003     ;code of 4th M function
      C003       ;M code classification
 Z               ;end of condition 4th M function sent

 I524            ;5th M function sent
      DF0120     ;no executable M code found
      LRH004     ;code of 5th M function
      C003       ;M code classification
 Z               ;end of condition 5th M function sent

 I525            ;if S function sent
      ,1         ;1 to OP
      SQ04       ;storing into S schedule counter
```

```
      LRH005      ;loading S function code to OP
      SF026       ;storing into S function code
                  ;to shadow register
      DF0134      ;revolution execution disabled
      UQ00        ;increment FIN counter
 Z                ;end of condition S function sent

Z                 ;end of condition
                  ;inactivating MCH.LK state

I526              ;if T function sent
 (NY441           ;if no machine lock
 ANY472           ;and no function lock
 ANY440)          ;and no test

      ,1          ;1 to OP
      SQ02        ;storing into T schedule counter
      LRH006      ;loading T function code to OP
      SF024       ;storing into T function code
                  ;to shadow register
      DF0132      ;tool preparation execution
                  ;disabled
      UQ00        ;increment FIN counter
 E                ;else test
      LRH006      ;loading T function code into OP
      SF082       ;gained T code
 Z                ;end of condition no function lock ...

Z                 ;end of condition T function sent



/* handling FIN flag */

      LQ00        ;loading FIN counter to OP
 =0               ;if content 0
      U547        ;functions executed by PLC
 E                ;else
      D547        ;execution in progress
 Z                ;end of condition content 0


/* clearing START / STOP state */

(NI546            ;if no executable block
                  ;in buffer
AY547             ;and FIN on
AI551             ;and interpolator empty
ANY507            ;and no FSBS state
ANF0101           ;and JOG push-buttons disabled
ANI545)           ;if no G28

      D470        ;inactivating START state
      D471        ;inactivating STOP state

Z                 ;end of condition
                  ;no executable ...
```

```
/* handling M3, M4, M5 */

NY483            ;if no spindle JOG push-button

      D474       ;inactivating M3 on control board 2
      D475       ;inactivating M4 on control board 2
      D476       ;inactivating M5 on control board 2
      LRH062     ;loading rotation code

 =3              ;if M3
      U474       ;activating M3 on control board 2
 Z               ;end of condition M3

 =4              ;if M4
      U475       ;activating M4 on control board 2
 Z               ;end of condition M4

 =5              ;if M5
      U476       ;activating M5 on control board 2
 Z               ;end of condition M5

Z                ;end of condition no spindle jog push-button



/* taking constant surface speed into account */

(NY440           ;if no test
ANY441           ;and no machine lock
ANY472)          ;and no function lock
 I653            ;if G96

      LRH012     ;calculated spindle revolution
      SRH060     ;storing
 Z               ;end of condition G96
Z                ;end of condition
                 ;if no test ...



/* tool number display */

(NY441           ;if no machine lock
ANY472           ;and no function lock
ANY440)          ;and no test
      LF080      ;loading active tool
E                ;else
      LF082      ;gained T code
Z                ;end of condition if no machine lock ...
      SRH064     ;storing for display



/* scrolling functions: FSBS */

(V507AI507)      ;if FSBS softkey selected
 NY507           ;if FSBS active
      U507       ;activating FSBS
      DF0130     ;function stop on
```

```
 E                 ;else
      D507         ;inactivating FSBS
      UF0130       ;activating function stop
 Z                 ;end of condition FSBS active
Z                  ;end of condition FSBS softkey selected

J1                 ;end of module :001

/* end of module :001 */


/* selecting M codes */

:003               ;M code classification

=6                 ;if equal to 6
      ,1           ;1 to OP
      SQ01         ;storing into M06 tool replacement phase counter
      DF0131       ;tool replacement execution disabled
                   ;function executions start from here
      UF0120       ;executable M code found
      G004         ;goto label :004
Z                  ;end of condition equal to 6

>=10               ;if greater than or equal to 10
 <=18              ;if less than or equal to 18
      SF016        ;storing into range code register
                   ;(value: 10, 11, ..., 18)
      ,1           ;1 to OP
      SQ03         ;storing into M10,...,M18 gear range change phase counter
      DF0133       ;gear range change execution disabled
      UF0120       ;executable M code found
      G004         ;goto label :004
 Z                 ;end of condition less than or equal to 18
Z                  ;end of condition greater than or equal to 10

>=3                ;if greater than or equal to 3
 <=5               ;if less than or equal to 5
      D483         ;spindle jog cancel
      SF018        ;storing into rotation code register
                   ;(value: 3, 4, 5)
      ,1           ;1 to OP
      SQ05         ;storing into M3,M4,M5,M19 spindle rotation phase counter
      DF0135       ;spindle rotation execution disabled
      UF0120       ;executable M code found
      DF0121       ;M3, M4 from program
      DF0122       ;M5 from program
      G004         ;goto label :004
 Z                 ;less than or equal to 4 end of condition
Z                  ;greater than or equal to 3 end of condition

=19                ;if equal to 19
      D483         ;spindle jog cancel
      SF018        ;storing into rotation code register
                   ;(value: 19)
      ,1           ;1 to OP
      SQ05         ;storing into M3,M4,M5,M19 spindle rotation phase counter
      DF0135       ;spindle rotation execution disabled
      UF0120       ;executable M code found
      DF0121       ;M3, M4 from program
      DF0122       ;M5 from program
      G004         ;goto label :004
Z                  ;end of condition equal to 19
```

```
>=8                 ;if greater than or equal to 8
 <=9                ;if less than or equal to 9
      SRH070        ;storing into programmed M8/M9 state
      G004          ;goto label :004
 Z                  ;end of condition less than or equal to 9
Z                   ;end of condition greater than or equal to 8

>=0                 ;if greater than or equal to 0
 <=2                ;if less than or equal to 2
      SF028         ;storing into program controlling code register
      ,1            ;1 to OP
      SQ19          ;storing into program controlling phase counter
      DF0147        ;program controlling command execution
                    ;disabled
      UF0120        ;executable M code found
      G004          ;goto label :004
 Z                  ;end of condition less than or equal to 2
Z                   ;end of condition greater than or equal to 0

=30                 ;if equal to 30
      SF028         ;storing into program controlling code register
      ,1            ;1 to OP
      SQ19          ;storing into program controlling phase counter
      DF0147        ;program controlling command execution
                    ;disabled
      UF0120        ;executable M code found
      G004          ;goto label :004
Z                   ;end of condition equal to 30

:004                ;label :004
F0120               ;if executable M code found
      UQ00          ;incrementing FIN counter
Z                   ;end of condition
                    ;executable M code found

R                   ;return from M code classification


/* operations before interruption of AUTO */

:009                ;operations before interruption of AUTO

      LQ00          ;loading FIN counter to OP
      SF050         ;storing into FIN counter saving register
      LQ01          ;loading tool replacement (M06) phase counter
                    ;to OP
      C015          ;auxiliary module: if OP<0 then OP=1
      SF052         ;storing into tool replacement (M06) phase counter
                    ;saving register
      LQ02          ;loading tool preparation (T) phase counter
                    ;to OP
      C015          ;auxiliary module: if OP<0 then OP=1
      SF054         ;storing into tool preparation (T) phase counter
                    ;saving register
      LQ03          ;loading gear range change (M10, M11, ..., M18)
                    ;phase counter to OP
      C015          ;auxiliary module: if OP<0 then OP=1
      SF056         ;storing into gear range change (M10, M11, ..., M18)
                    ;phase counter saving register
      LQ04          ;loading spindle revolution (S) phase counter
                    ;to OP
      C015          ;auxiliary module: if OP<0 then OP=1
      SF058         ;storing into spindle revolution (S) phase counter
                    ;saving register
```

```
      LQ05        ;loading spindle rotation (M3, M4, M5, M19)
                  ;phase counter to OP
      C015        ;auxiliary module: if OP<0 then OP=1
      SF060       ;storing into spindle rotation (M3, M4, M5, M19)
                  ;phase counter saving register
      LQ06        ;loading coolant (M8, M9) phase counter to OP
      C015        ;auxiliary module: if OP<0 then OP=1
      SF062       ;storing into coolant (M8, M9) phase counter
                  ;saving register
      LQ19        ;loading program controlling codes (M00, M01, M02,
                  ;M30) phase counter to OP
      C015        ;auxiliary module: if OP<0 then OP=1
      SF078       ;storing into program controlling codes (M00, M01, M02,
                  ;M30) phase counter saving register

R                 ;return from
                  ;operations before interruption of AUTO


/* for auxiliary module :009 */

:015
>0                ;if function under execution
      ,1          ;start function execution from the beginning
Z                 ;end of condition function ...
R


/* operations after return to AUTO */

:010              ;operations after return to AUTO

      LF050       ;loading FIN counter saving register
                  ;to OP
      SQ00        ;storing into FIN counter
      LF052       ;loading tool replacement (M06) phase counter
                  ;saving register to OP
      SQ01        ;storing into tool replacement (M06) phase counter
      LF054       ;loading tool preparation (T) phase counter
                  ;saving register to OP
      SQ02        ;storing into tool preparation (T)
                  ;phase counter
      LF056       ;loading range code (M10, M11, ..., M18)
                  ;phase counter saving register to OP
      SQ03        ;storing into range code (M10, M11, ..., M18)
                  ;phase counter
      LF058       ;loading spindle revolution (S) phase counter
                  ;saving register to OP
      SQ04        ;storing into spindle revolution (S)
                  ;phase counter
      LF060       ;loading spindle rotation (M3, M4, M5, M19)
                  ;phase counter saving register to OP
      SQ05        ;storing into spindle rotation (M3, M4, M5, M19)
                  ;phase counter
      LF062       ;loading coolant (M8, M9) phase counter
                  ;saving register to OP
      SQ06        ;storing into coolant (M8, M9) phase counter
      LF078       ;loading program controlling codes (M00, M01, M02,
                  ;M30) phase counter saving register
                  ;to OP
      SQ19        ;storing into program controlling codes (M00, M01, M02,
                  ;loading M30) phase counter

R                 ;return from
```

```
                  ;operations after return to AUTO


/* function RESET */

:011              ;function RESET

      DF0130      ;clearing function stop
      DF0131      ;tool replacement execution disabled
      DF0132      ;tool preparation
                  ;execution disabled
      DF0133      ;gear range change execution disabled
      DF0134      ;spindle revolution
                  ;execution disabled
      DF0135      ;spindle rotation execution disabled
      DF0147      ;program controlling command
                  ;execution disabled
      DF0103      ;interruption enabling
                  ;reset enabled
      UF0102      ;interruption enabled
      ,0          ;0 to OP
      SQ00        ;clearing FIN counter
      SQ01        ;clearing tool replacement (M06) phase counter
      SQ02        ;clearing tool preparation (T)
                  ;phase counter
      SQ03        ;clearing range code (M10, M11, ..., M18)
                  ;phase counter
      SQ04        ;clearing spindle revolution (S)
                  ;phase counter
      SQ05        ;clearing spindle rotation (M3, M4, M5, M19)
                  ;phase counter
      SQ06        ;clearing coolant (M8, M9) phase counter
      SQ19        ;program controlling codes
                  ;(M00, M01, M02, M30)
                  ;clearing phase counter

R                 ;return from function RESET


/* start push-buttons RESET */

:012              ;start push-buttons RESET

      D470        ;inactivating START state
      D471        ;inactivating STOP state
      DF0110      ;clearing test JOG push-buttons for START
      DF0111      ;clearing initiate START state
      DF0112      ;clearing initiate STOP state
      DF0101      ;clearing JOG push-buttons enabled
      LY42        ;loading line Y42
      A.007F      ;clearing JOG bits Y427,Y430,...,Y437
      SY42        ;storing
      LY44        ;loading line Y42
      A.007F      ;clearing JOG lamps Y427,Y430,...,Y437
      SY44        ;storing
      DF0125      ;clearing initiate M3 state
      DF0126      ;clearing initiate M4 state
      DF0127      ;clearing initiate M5 state

R                 ;return from start push-buttons RESET


/* interface board RESET */
```

```
:013                ;interface board RESET

      ,0            ;0 to OP
      SY00          ;1st interface board Y000...Y017 output lines off
      SY02          ;1st interface board Y020...Y037 output lines off
      SY10          ;2nd interface board Y100...Y117 output lines off
      SY12          ;2nd interface board Y120...Y137 output lines off
      SY20          ;3rd interface board Y200...Y217 output lines off
      SY22          ;3rd interface board Y220...Y237 output lines off
      SY30          ;4th interface board Y300...Y317 output lines off
      SY32          ;4th interface board Y320...Y337 output lines off

R                   ;return from interface board RESET


/* output flags RESET */

:014                ;output flags RESET

      D650          ;no spindle rotation
      D652          ;1st spindle command signal output disabled
      DF0114        ;spindle not started
      ,5            ;5 to OP
      SRH062        ;storing into 1st spindle rotation state register
      ,9            ;9 To OP
      SRH070        ;storing into M9
      D470          ;inactivating START state
      D471          ;inactivating STOP state
      D540          ;inactivating MON output line

R                   ;return from output flags RESET


/* spindle rotation from push-buttons */

:016

NF0121              ;if end of M3, M4 from push-buttons
      LQ05          ;loading M3,M4,M5,M19 phase counter
      SF032         ;storing into Q05 spindle rotation (M3, M4, M5, M19)
                    ;phase counter
      LF018         ;loading rotation code register
      SF030         ;storing into rotation code
E                   ;else, if no save needed under process
      DQ00          ;decrementing FIN counter
Z                   ;end of M3, M4 from control board

F0125               ;if initiate M3 state
      ,3            ;3 to OP
Z                   ;end of condition initiate M3 state
F0126               ;if initiate M4 state
      ,4            ;4 to OP
Z                   ;end of condition initiate M4 state
F0127               ;if initiate M5 state
      DF0121        ;M3, M4 not under process
      ,5            ;5 to OP
Z                   ;end of condition initiate M5 state
      SF018         ;storing into rotation code register
                    ;(value: 3, 4)
      ,1            ;1 to OP
      SQ05          ;storing into spindle rotation (M3,M4,M5,M19) phase counter
      UQ00          ;increment FIN counter
      UF0135        ;spindle rotation execution enabled
      DF0102        ;interruption disabled
```

```
R                  ;end of module


/* start of module :000 */

:000               ;module :000 started

Y507               ;if FSBS operation

 F0130             ;if function stop
      G196         ;goto end module
 E                 ;else
      UF0130       ;making a cycle
                   ;and requesting FUNCTION STOP
      UF0112       ;initiate STOP state
 Z                 ;end of condition function stop

Z                  ;end of condition FSBS operation


/* function dispatcher */

Y470               ;if START state

 I553              ;if spindle rotation request

  (NF0133          ;if no range code enabled
  ONF0134          ;or revolution enabled
  ONF0135)         ;or spindle rotation under process

   (NY710          ;if no SPINDLE REVOLUTION ERROR
   ANY711)         ;and no SPINDLE RISING/FALLING EDGE
      LQ05         ;loading spindle rotation phase counter
    =0             ;if not started
     NY650         ;if no spindle rotation
      U713         ;SPINDLE ROTATION REQUEST message on
     E             ;if rotation
      D713         ;SPINDLE ROTATION REQUEST message off
      UF0131       ;tool replacement execution enabled
     Z             ;end of condition no spindle rotation
    E              ;else, if started
      LF018        ;loading rotation code shadow register
     =3            ;if M3
      UF0133       ;range execution enabled
      D713         ;SPINDLE ROTATION REQUEST message off
     Z             ;end of condition M3
     =4            ;if M4
      UF0133       ;range execution enabled
      D713         ;SPINDLE ROTATION REQUEST message off
     Z             ;end of condition M4
     NF0133        ;if no command M3 or M4
      NY650        ;if no spindle rotation
      U713         ;SPINDLE ROTATION REQUEST message on
      E            ;else
      D713         ;SPINDLE ROTATION REQUEST message off
      Z            ;end of condition no spindle rotation
     Z             ;end of condition no command M3 or M4
    Z              ;end of condition not started
   E               ;else SPINDLE REVOLUTION ERROR
      D713         ;SPINDLE ROTATION REQUEST message off
   Z               ;end of condition no SPINDLE REVOLUTION ERROR
  Z                ;end of condition no ... under process

 E                 ;interpolator does not request spindle rotation...
```

```
      UF0131      ;tool replacement execution
                  ;enabled
 Z                ;end of condition
                  ;interpolator requests spindle rotation,

Z                 ;end of condition START state


/* function executions */


      /* M6 tool replacement execution */

F0131             ;if M6 execution enabled,

      LQ01        ;loading Q01 to OP
 =0               ;if no M6
      DF0131      ;M6 execution disabled
      UF0132      ;T execution enabled
 Z                ;end of condition no M6

 =1               ;if 1st phase: test
  I551            ;if interpolator empty
      ,0          ;0 to OP
      SQ01        ;clearing M6 phase counter (no action)
      DQ00        ;decrementing FIN counter
      UF0102      ;interruption enabled
  Z               ;end of condition empty interpolator
      ,1          ;1 To OP
 Z                ;end of condition 1st phase

Z                 ;end of condition
                  ;M6 execution enabled


      /* T execution */

F0132             ;if T execution enabled

      LQ02        ;loading Q02 to OP
 =0               ;if no T
      DF0132      ;T execution disabled
      UF0133      ;range code execution enabled
                  ;enabled
 Z                ;end of condition no T

 =1               ;if 1st phase: test,
                  ;requesting STOP state
  I551            ;if empty interpolator
      DF0102      ;interruption disabled
      LF080       ;code of tool in spindle to OP
   =LF024         ;tool in spindle=programmed tool
      ,0          ;0 to OP
      SQ02        ;clearing T command (no action)
      DQ00        ;decrementing FIN counter
      UF0102      ;interruption enabled
   E              ;if not equal
      UF0112      ;initiate STOP state
      UQ02        ;goto 2nd phase
   Z              ;end of condition
                  ;tool in spindle=programmed tool
  Z               ;end of condition empty interpolator
      ,1          ;1 to OP
 Z                ;end of condition 1st phase
```

```
 =2               ;if 2nd phase: requesting spindle stop
  Y471            ;if STOP state
      LRH062      ;loading 1st spindle rotation state to OP
   =5             ;if M5 state
      ,4          ;4 to OP
      SQ02        ;storing into phase counter Q02
   E              ;else, if spindle rotation
      C005        ;preparing spindle stop
      UQ02        ;incrementing Phase counter Q02
   Z              ;end of condition M5 state
  Z               ;end of condition STOP state
      ,2          ;2 to OP
 Z                ;end of condition 2nd phase

 =3               ;if 3rd phase: resetting spindle
                  ;rotation code
      LQ05        ;loading phase counter M3,M4,M5,M19
  =0              ;cmmand M5 executed
      C006        ;resetting spindle rotation code
      UQ02        ;incrementing phase counter Q02
  Z               ;end of condition command M5 executed
      ,3          ;3 To OP
 Z                ;end of condition 3rd phase

 =4               ;if 4th phase: coolant stop
      LY002       ;loading coolant pump state
      SF0123      ;saving coolant pump state
      D002        ;coolant pump off
      UQ02        ;incrementing phase counter Q02
      ,4          ;4 to OP
 Z                ;end of condition 4th phase

 =5               ;if 5th phase: messageing tool number
      LRH006      ;loading T code to OP
      BCD         ;binary BCD conversion
      SRH090      ;into T code message register in decimal form
      U700        ;requesting 1st indexed message
      UQ02        ;goto 3rd phase
      ,5          ;5 to OP
 Z                ;end of condition 5th phase

 =6               ;if 6th phase
  (I700           ;if 1st indexed message on screen
  AY470)          ;and START state
      LF024       ;loading T function code to OP
      SF080       ;code of tool in 1st spindle
      D700        ;inactivating 1st indexed message
      LF0123      ;loading coolant pump state
      SY002       ;activating coolant pump
      ,0          ;0 to OP
      SQ02        ;clearing T phase counter (no push-button)
      DQ00        ;decrementing FIN counter
      UF0102      ;interruption enabled
  Z               ;end of condition 1st indexed
                  ;message on screen and START state
      ,6          ;6 to OP
 Z                ;end of condition 6th phase

Z                 ;end of condition
                  ;T execution enabled


      /* spindle gear range change execution */
```

```
F0133             ;if gear range change execution
                  ;enabled

      LQ03        ;loading Q03 to OP
 =0               ;if no gear range change command
      DF0133      ;gear range change execution disabled
      UF0134      ;S execution enabled
 Z                ;end of command no gear range change command


 =1               ;if 1st phase: test,
                  ;requesting STOP state
      DF0102      ;interruption disabled
      LRH063      ;1st spindle range state to OP
  =LF016          ;=programmed tool
      ,0          ;0 to OP
      SQ03        ;clearing gear range change phase counter (no push-button)
      DQ00        ;decrementing FIN counter
      UF0102      ;interruption enabled
  E               ;if not equal
      UF0112      ;initiate STOP state
      UQ03        ;goto 3rd phase
  Z               ;end of condition =programmed tool
      ,1          ;1 to OP
 Z                ;end of condition 1st phase

 =2               ;if 2nd phase: requesting spindle stop
  Y471            ;if STOP state
      LRH062      ;loading 1st spindle rotation state to OP
   =5             ;if M5 state
      ,4          ;4 to OP
      SQ03        ;storing into phase counter Q03
   E              ;else, if rotation
      C005        ;preparing spindle stop
      UQ03        ;incrementing phase counter Q03
   Z              ;end of condition M5 state
  Z               ;end of condition STOP state
      ,2          ;2 to OP
 Z                ;end of condition 2nd phase

 =3               ;if 3rd phase: resetting spindle rotation code
      LQ05        ;loading phase counter M3,M4,M5,M19
  =0              ;command M5 executed
      C006        ;resetting spindle rotation code
      UQ03        ;incrementing phase counter Q03
  Z               ;end of condition command M5 executed
      ,3          ;3 to OP
 Z                ;end of condition 3rd phase

 =4               ;if 4th phase: requesting coolant stop
      LY002       ;loading coolant pump state
      SF0123      ;saving coolant pump state
      D002        ;coolant pump off
      UQ03        ;incrementing phase counter Q03
      ,4          ;4 to OP
 Z                ;end of condition 4th phase

 =5               ;if 5th phase
      LF016       ;loading range code to OP
      -10         ;subtracting 10
      BCD         ;binary BCD conversion
      SRH091      ;range code to message register in decimal form
      U701        ;requesting 2nd indexed message
      UQ03        ;goto 7th phase
```

```
      ,5            ;5 to OP
 Z                  ;end of condition 5th phase

 =6                 ;if 6th phase
  (I701             ;if 2nd indexed message
  AY470)            ;and START state
      LF016         ;loading range code to OP
      SRH063        ;code of 1st spindle range
      D701          ;clearing 2nd indexed message
      LF0123        ;loading coolant pump state
      SY002         ;activating coolant pump
      ,0            ;0 to OP
      SQ03          ;clearing gear range change phase counter
                    ;(no action)
      DQ00          ;decrementing FIN counter
      UF0102        ;interruption enabled
  Z                 ;end of condition
                    ;2nd indexed message and START
      ,6            ;6 to OP
 Z                  ;end of condition 6th phase


Z                   ;gear range change execution
                    ;end of condition enabled


      /* S execution */

F0134               ;if S execution enabled

      LQ04          ;loading phase counter Q04 to OP
 =0                 ;if no command S
      DF0134        ;S execution disabled
      UF0135        ;spindle rotation execution
                    ;enabled
 Z                  ;end of condition no command S


 =1                 ;if 1st phase
      DF0102        ;interruption disabled
      LF026         ;code of S function to OP
      SRH060        ;loading 1st spindle current
                    ;revolution register
  F0114             ;if spindle started
      ,5            ;5 to OP
      SH00          ;loading spindle timer
      UQ04          ;incrementing phase counter
  E                 ;else no spindle rotation
      DQ00          ;decrementing FIN counter
      UF0102        ;interruption enabled
      ,0            ;loading 0 to OP
      SQ04          ;clearing phase counter
  Z                 ;end of condition spindle rotation

      ,1            ;1 to OP
 Z                  ;end of condition 1st phase

 =2                 ;if 2nd phase
  NH00              ;loading timer
                    ;if terminated
      ,0            ;0 to OP
      SRH061        ;loading spindle JOG command signal register
      U654          ;1st spindle command signal direct output
      D652          ;1. spindle command signal output enabled
```

```
      D001         ;drive enabled
      DF0114       ;spindle not started
      UF0112       ;initiate STOP state
      ,5           ;M5
      SRH062       ;loading 1st spindle rotation state register
      U711         ;SPINDLE RISING/FALLING EDGE message on
      UF0102       ;interruption enabled
  E                ;else
   (I650           ;if 1st spindle command signal ready
   AI656)          ;and N=Ns
      DQ00         ;decrementing FIN counter
      UF0102       ;interruption enabled
      ,0           ;loading 0 To OP
      SQ04         ;clearing phase counter
   Z               ;end of condition
                   ;1st spindle command signal ready
  Z                ;loading timer
      ,2           ;2 to OP
 Z                 ;end of condition 2nd phase

Z                  ;end of condition
                   ;S execution enabled


      /* spindle rotation execution */

F0135              ;if spindle rotation execution enabled

      LQ05         ;loading Q05 to OP
 =0                ;if no spindle rotation command
      DF0135       ;spindle rotation execution disabled
      UF0147       ;program controlling commands enabled
 Z                 ;end of condition
                   ;no spindle rotation command


 =1                ;if 1st phase
      DF0102       ;interruption disabled
      LF018        ;loading rotation code register to OP
  =5               ;if M5
      ,10          ;M5 start from 10th phase
  Z                ;end of condition M5
  =19              ;if M19
      ,10          ;M19 start from 10th phase
  Z                ;end of condition M19
  =3               ;if M3
      ,50          ;M3 start from 50th phase
  Z                ;end of condition M3
  =4               ;if M4
      ,50          ;M4 start from 50th phase
  Z                ;end of condition M4
      SQ05         ;storing into phase counter
      ,1           ;1 to OP
 Z                 ;end of condition 1st phase


      /* cycles M5, M19 */

 =10               ;if 10th phase (M5, M19 start)
  F0122            ;if M5 from push-buttons
      ,11          ;goto 11th phase
  E                ;else from program
      ,15          ;goto 15th phase
  Z                ;end of condition M5 from push-buttons
```

```
     SQ05         ;storing into phase counter
     ,10          ;10 to OP
Z                 ;end of condition
                  ;10th phase (M5, M19 start)

=11               ;if 11th phase (evaluations if M5
                  ;from control board, or flag F0127)
 NI552            ;if override enabled
  (I553           ;if spindle rotation request
  ANY710)         ;and no SPINDLE REVOLUTION ERROR message
     UF0112       ;initiate STOP state
     ,12          ;goto 12th phase
  E               ;else, if no spindle
                  ;rotation request ...
     ,20          ;goto stop
  Z               ;end of condition spindle rotation request
     SQ05         ;storing into phase counter
 E                ;else, if override disabled
     ,20          ;goto stop
     SQ05         ;storing into phase counter
 Z                ;end of condition override enabled
     ,11          ;11 to OP
Z                 ;end of condition 11th phase

=12               ;if 12th phase (did feed stop)
 (I550            ;if interpolator standstill
 AY471)           ;and STOP state active
     ,20          ;goto stop
     SQ05         ;storing 20 to phase counter
 Z                ;interpolator standstill and STOP state active
     ,12          ;12 to OP
Z                 ;end of condition 12th phase

=15               ;if 15th phase (test
                  ;if M5, M19 from program)
 (NI553           ;if no spindle rotation request
 OF0133)          ;or gear range change occurs
     ,20          ;goto stop
     SQ05         ;storing into phase counter
 Z                ;end of condition no spindle rotation
                  ;request or gear range change occurs
     ,15          ;15 to OP
Z                 ;end of condition 15th phase

=20               ;if 20th phase (initiating stop)
     D651         ;orientation request off
 I651             ;if loop closed on 1st spindle
 E                ;else, if not
     LF018        ;loading spindle rotation code register into OP
  =19             ;if M19
     LRH063       ;loading 1st spindle rangen code
   =11            ;if M11
     .00FF        ;zero pulse search rate in 1st range
     SRH061       ;storing 1st spindle jog command signal register
     ,11          ;11 back to OP
   Z              ;end of condition M11
   =12            ;if M12
     .00FF        ;zero pulse search rate in 2nd range
     SRH061       ;storing 1st spindle jog command signal register
     ,12          ;12 back to OP
   Z              ;end of condition M12
   =13            ;if M13
     .00FF        ;zero pulse search rate in 3rd range
     SRH061       ;storing 1st spindle jog command signal register
```

```
     ,13            ;13 back to OP
   Z                ;end of condition M13
   =14              ;if M14
     .00FF          ;zero pulse search rate in 4th range
     SRH061         ;storing 1st spindle jog command signal register
     ,14            ;14 back to OP
   Z                ;end of condition M14
   =15              ;if M15
     .00FF          ;zero pulse search rate in 5th range
     SRH061         ;storing 1st spindle jog command signal register
     ,15            ;15 back to OP
   Z                ;end of condition M15
   =16              ;if M16
     .00FF          ;zero pulse search rate in 6th range
     SRH061         ;storing 1st spindle jog command signal register
     ,16            ;16 back to OP
   Z                ;end of condition M16
   =17              ;if M17
     .00FF          ;zero pulse search rate in 7th range
     SRH061         ;storing 1st spindle jog command signal register
     ,17            ;17 back to OP
   Z                ;end of condition M17
   =18              ;if M18
     .00FF          ;zero pulse search rate in 8th range
     SRH061         ;storing 1st spindle JOG command signal register
     ,18            ;18 back to OP
   Z                ;end of condition M18

   Y653             ;if - yes
     LRH061         ;
     NSRH061        ;command signal sign reversal for
                    ;orientation in spindle rotation direction
   Z                ;end of condition - yes

   F0114            ;if spindle started
     ,25            ;5 to OP
     SH00           ;storing spindle timer
     U654           ;direct 1st spindle command signal transfer
     U652           ;enabling 1st spindle command signal transfer
     U001           ;enabling main drive
     ,30
     SQ05           ;30th phase
   E                ;spindle not started
     ,25            ;5 to OP
     SH00           ;storing spindle timer
     D654           ;direct 1st spindle command signal transfer off
     U651           ;orientation request
     U652           ;enabling 1st spindle command signal transfer
     U001           ;enabling main drive
     ,31
     SQ05           ;31st phase
   Z                ;end of condition spindle started

  E                 ;else M5
     DF0114         ;spindle not started
     ,5             ;5 to OP
     SH00           ;storing spindle timer
     U654           ;direct 1st spindle command signal transfer
     ,0             ;ö to OP
     SRH061         ;storing 1st spindle JOG command signal register
     U652           ;enabling 1st spindle command signal transfer on
     U001           ;enabling main drive on
     UQ05           ;incrementing phase counter
  Z                 ;end of condition =19
```

```
 Z              ;end of condition loop closed on 1st spindle
     ,20        ;20 to OP
Z               ;end of condition 20th phase


     /* cycle M5 */

=21             ;if 21st phase
 NH00           ;testing timer
                ;if terminated
     D652       ;disabling 1st spindle command signal transfer
     D001       ;disabling main drive off
     UF0112     ;activate STOP state
     SRH062     ;storing 1st spindle rotation state register
     U711       ;SPINDLE RISE/FALL ERROR on
     UF0102     ;enabling interrupt
 E              ;else
  (I650         ;if spindle command signal ramped down
  AI657)        ;and spindle stopped
     D652       ;disabling 1st spindle command signal transfer
     D001       ;disabling main drive
     SRH062     ;storing 1st spindle rotation state register
     DQ00       ;decrementing FIN counter
   F0103        ;if disabling resetting
                ;enabling interrupt
   E            ;else
     UF0102     ;enabling interrupt
   Z            ;end of condition disabling resetting
                ;enabling interrupt
     ,0         ;0 to OP
     SQ05       ;clearing phase counter
  Z             ;end of condition spindle command signal ramped down
 Z              ;displaying timer
     ,21        ;21 to OP
Z               ;end of condition 21st phase


     /* cycle M19 */

=30             ;if 30th phase
 NH00           ;testing timer
                ;if terminated
     ,0         ;0 to OP
     SRH061     ;storing 1st spindle JOG command signal register
     U654       ;direct 1st spindle command signal transfer
     D652       ;disabling 1st spindle command signal transfer
     D001       ;disabling main drive
     DF0114     ;spindle not started
     UF0112     ;activate STOP state
     U712       ;SPINDLE ORIENTATION ERROR on
     ,5         ;M5
     SRH062     ;storing 1st spindle rotation state register
     UF0102     ;enabling interrupt
 E              ;else
  (I650         ;if command signal ready
  AI656)        ;and n=ns
     ,25        ;5 to OP
     SH00       ;storing spindle timer
     D654       ;direct 1st spindle command signal transfer off
     U651       ;orientation request
     U652       ;enabling 1st spindle command signal transfer
     UQ05       ;incrementing phase counter
  Z
 Z              ;end of condition NH00
```

```
     ,30          ;30 to OP
Z                 ;end of condition =30

=31               ;if 31st phase
 NH00             ;displaying timer
                  ;if terminated
     ,0           ;0 to OP
     SRH061       ;storing 1st spindle JOG command signal register
     D651         ;orientation request off
     U654         ;direct 1st spindle command signal transfer
     D652         ;disabling 1st spindle command signal transfer
     D001         ;disabling main drive
     DF0114       ;spindle not started
     UF0112       ;activate STOP state
     U712         ;SPINDLE ORIENTATION ERROR on
     ,5           ;M5
     SRH062       ;storing 1st spindle rotation state register
     UF0102       ;enabling interrupt
 E                ;else
  (I651           ;if 1st spindle loop closed and oriented
  AI652)          ;and spindle in position
     DF0114       ;spindle not started
     LF018        ;loading spindle rotation code register to OP
     SRH062       ;storing 1st spindle rotation state register
     DQ00         ;incrementing FIN counter
   F0103          ;if disabling resetting
                  ;enabling interrupt
   E              ;else
     UF0102       ;enabling interrupt
   Z              ;end of condition disabling resetting
                  ;enabling interrupt
     ,0           ;0 to OP
     SQ05         ;clearing phase counter
  Z               ;end of condition 1st spindle loop closed and oriented
 Z                ;end of condition
                  ;displaying timer
     ,31          ;31 to OP
Z                 ;end of condition 31st phase


     /* cycles M3, M4 */

=50               ;if 50th phase (M3, M4 start)
 (I552            ;if override disabled
 AY542)           ;and FEED HOLD
                  ;exit and no start
     DQ00         ;decrementing FIN counter
  F0103           ;if interruption enabling
                  ;reset disabled
  E               ;else
     UF0102       ;interruption enabled
  Z               ;end of condition interruption enabling
                  ;reset disabled
     ,0           ;loading 0 to OP
     SQ05         ;clearing phase counter
 E                ;else
     D651         ;inactivating orientation request
  I651            ;if 1st spindle loop closed
  E               ;else, if not
   (I552          ;if override disabled
   ANY470)        ;and no START state
     U714         ;START REQUEST message on
   E              ;else
     D714         ;START REQUEST message off
```

```
      LF018       ;loading spindle rotation code register to OP
     =3           ;if M3
      U653        ;1st spindle command signal + polarity
     E            ;else M4
      D653        ;1st spindle command signal - polarity
     Z            ;end of condition M3
      D654        ;1st spindle command signal
                  ;direct output disabled
      U652        ;1st spindle command signal output enabled
      U001        ;drive enabled
      UF0114      ;spindle started
      ,5          ;5 to OP
      SH00        ;storing into spindle timer
      UQ05        ;incrementing phase counter
    Z             ;end of condition override disabled ...
   Z              ;end of condition 1st spindle loop closed
  Z               ;end of condition override disabled ...
      ,50         ;50 to OP
 Z                ;end of condition 50th phase (M3, M4 start)

 =51              ;if 51st phase
  NH00                  ;if revolution ready timer terminated
      ,0          ;0 to OP
      SRH061      ;storing into spindle JOG command signal register
      U654        ;1st spindle command signal direct output
      D652        ;1st spindle command signal output disabled
      D001        ;drive disabled
      DF0114      ;spindle not started
      UF0112      ;initiate STOP state
      ,5          ;M5
      SRH062      ;loading 1st spindle rotation state register
      U711        ;SPINDLE RISING/FALLING EDGE message on
      UF0102      ;interruption enabled
  E               ;else
   (I650          ;if spindle command signal ready
   AI656)         ;and N=Ns
      LF018       ;loading spindle rotation code register
                  ;to OP
      SRH062      ;storing into 1st spindle rotation state
                  ;register
      DQ00        ;decrementing FIN counter
    F0103         ;if interruption enabling
                  ;reset disabled
    E             ;else
      UF0102      ;interruption enabled
    Z             ;end of condition interruption enabling
                  ;reset disabled
      ,0          ;loading 0 to OP
      SQ05        ;clearing phase counter
   Z              ;spindle command signal ready
  Z               ;end of condition
                  ;testing revolution ready timer
      ,51         ;51 to OP
 Z                ;end of condition 51st phase

Z                 ;spindle rotation execution
                  ;end of condition enabled


      /* execution of program controlling commands */

F0147             ;if program controlling command
                  ;execution enabled
```

```
     LQ19           ;loading Q19 to OP
=0                  ;if no program controlling command
     DF0147         ;program controlling code execution disabled
Z                   ;end of condition no program controlling command

=1                  ;if 1st phase: waiting for end of block
 I551               ;if empty interpolator
     DF0102         ;interruption disabled
     UQ19           ;incrementing phase counter
     LF028          ;program controlling code loading to OP
  =1                ;if M1: conditional STOP
   Y445             ;if CND.SP (conditional STOP) state active
   E                ;else, if inactive
     DQ00           ;decrementing FIN counter
     ,0             ;0 to OP
     SQ19           ;clearing phase counter: exit
     UF0102         ;interruption enabled
   Z                ;end of condition CND.SP state active
  Z                 ;end of condition M1
 Z                  ;end of condition empty interpolator
     ,1             ;1 to OP
Z                   ;end of condition 1st phase

=2                  ;if 2nd phase: requesting M5
     C005           ;preparing spindle stop
     UQ19           ;incrementing phase counter Q19
     ,2             ;2 to OP
Z                   ;end of condition 2nd phase

=3                  ;if 3rd phase
     LQ05           ;loading phase counter M3,M4,M5,M19
 =0                 ;command M5 executed
     LF028          ;loading program controlling code to OP
  >1                ;if M2, or M30
     ,9             ;loading 9 to OP
     SRH070         ;storing into programmed coolant code
     D002           ;coolant pump off
     D470           ;inactivating START state
     D471           ;inactivating STOP state
     ,0             ;loading 0 to OP
     SQ00           ;clearing FIN counter
     SQ05           ;clearing phase counter M3,M4,M5,M19
     SQ19           ;clearing program controlling commands
                    ;phase counter Q19, exit
     UF0102         ;interruption enabled
     DF0103         ;interruption enabling
                    ;reset enabled
  E                 ;else M0, or M1
     LY002          ;loading coolant pump state
     SF0123         ;saving coolant pump state
     D002           ;coolant pump off
     UF0112         ;initiate STOP state
     UQ19           ;incrementing phase counter Q19
  Z                 ;end of condition M2, or M30
 Z                  ;end of condition command M5 executed
     ,3             ;3 to OP
Z                   ;end of condition 3rd phase

=4                  ;if 4th phase:
 Y471               ;if STOP state active
     UQ19           ;incrementing phase counter Q19
 Z                  ;end of condition stop state
     ,4             ;4 to OP
Z                   ;end of condition 4th phase
```

```
 =5                ;if 5th phase: waiting, waiting for START,
                   ;and spindle back
   Y470            ;START state active
      C006         ;resetting spindle rotation code
      UF0135       ;spindle rotation execution
                   ;enabled
      UF0103       ;interruption enabling
                   ;reset disabled
      UQ19         ;incrementing phase counter Q19
   Z               ;START state active
      ,5           ;5 to OP
 Z                 ;end of condition 5th phase

 =6                ;if 6th phase: waiting for spindle
                   ;rotation, resetting coolant
      LQ05         ;loading phase counter M3,M4,M5,M19
  =0               ;spindle command executed
      LF0123       ;loading coolant pump state
      SY002        ;storing into coolant pump line
      DF0103       ;interruption enabling
                   ;reset enabled
      DQ00         ;decrementing FIN counter
      DF0147       ;program controlling command execution
                   ;disabled
      ,0           ;0 to OP
      SQ19         ;clearing phase counter: exit
      UF0102       ;interruption enabled
  Z                ;end of condition command M5 executed
      ,6           ;6 to OP
 Z                 ;end of condition 6th phase

Z                  ;program controlling command execution
                   ;enabled end of condition

:196               ;skip module of module :000

J0                 ;end of module :000


/* end of module :000 */


:005               ;preparing spindle stop

      LQ05         ;loading phase counter (M3,M4,M5,M19)
      SF032        ;storing into spindle rotation Q05 (M3, M4, M5, M19)
                   ;phase counter
>0                 ;rotation command waits
      LF018        ;loading rotation code register
E                  ;else rotation command does not wait
      LRH062       ;loading 1st spindle rotation state
                   ;register
Z                  ;rotation command waits

      SF030        ;saving rotation code
      DF0122       ;M5 from program
      ,5           ;loading 5 to OP
      SF018        ;M5 to rotation code register
      ,1           ;loading 1 to OP
      SQ05         ;storing into phase counter (M3,M4,M5,M19)
      UF0135       ;spindle rotation execution
                   ;enabled
      UQ00         ;incrementing FIN counter
      UF0103       ;interruption enabling
```

```
                  ;reset disabled

R                 ;end of module :005


:006              ;resetting spindle rotation code

      DF0135      ;spindle rotation execution disabled
      LF030       ;loading rotation code save
      SF018       ;storing into rotation code register
      ,1          ;loading 1 to OP
      SQ05        ;1st phase phase counter (M3,M4,M5,M19)
      LF032       ;loading spindle rotation Q05
                  ;(M3, M4, M5, M19)
                  ;phase counter

=0                ;if rotation not programmed
      UQ00        ;incrementing FIN counter
Z                 ;end of condition rotation not programmed

      DF0103      ;interruption enabling
                  ;reset enabled

R                 ;end of module :006


/* labels of PLC softkeys */

:197
,                 ;Y500
,                 ;Y501
,                 ;Y502
,                 ;Y503
,                 ;Y504
,                 ;Y505
,                 ;Y506
FSBS,             ;Y507
$

/* end of labels of PLC softkeys */



/* PLC messages */

:198
TOOL REPLACEMENT T,     ;Y700, RH090
RANGE,                  ;Y701, RH091
$

/* end of PLC messages */



/* PLC error messages */

:199
SPINDLE REVOLUTION ERROR,           ;Y710
SPINDLE RISING/FALLING EDGE,        ;Y711
SPINDLE ORIENTATION ERROR,          ;Y712
SPINDLE ROTATION REQUEST,           ;Y713
START REQUEST,                      ;Y714
,                                   ;Y715
,                                   ;Y716
```

```
,                                   ;Y717

,                                   ;Y720
,                                   ;Y721
,                                   ;Y722
,                                   ;Y723
,                                   ;Y724
,                                   ;Y725
,                                   ;Y726
,                                   ;Y727

,                                   ;Y730
$

/* end of PLC error messages */



/* PLC program code */

:200 MILLSAMPLE.PLC PROGRAM
MACHINE CONTROL BOARD 2
- RAPID TRAVERSE OVERRIDE:
CONST39=0 FROM SOFTKEYS,
CONST39=1 FROM F% ROTARY SWITCH, 4 STEPS
CONST39=2 MACHINE CONTROL BOARD Push-buttonS
CONST39=3 FROM F% ROTARY SWITCH, 13 STEPS
CONST39=4 FROM F% ROTARY SWITCH, 9 STEPS
$

/* end of PLC program code */
```

6.9 The Axrandom.plc Sample Program

Below excerpts of the sample program are shown. Expect of those below the program corresponds to example.plc program.
Tool preparation is implemented as the effect of T code, while replacement is executed by means of M06. The magazine handle is of random access, thus PLC uses tool pot table and PLC table. Code M20 empties tool from spindle.
If the called tool is not in the magazine PLC initiates manual replacement. Manual replacement and manual empty are activated by the use of codes M6 and M20.
Magazine rotation is bidirectional and realised by PLC axis. Running to position always occurs from positive direction. In case of magazine rotation in negative direction it overruns by one tool pot and runs to position in positive direction. Magazine rotation is executed at rapid traverse rate except for the last tool pot period which is done at feed rate.

```
/*

inner variables:
.........
F1000 -     incoming T code
F1001 -     new T=T in spindle
F1002 -     put tool manually in spindle
F1003 -     put tool from magazine in spindle

F1004 -     rotate magazine to called tool
F1005 -     magazine has reference position
F1006 -     magazine rotation direction=0: positive
F1007 -     magazine rotation

F1010 -     spindle empty command: M20
F1011 -     empty spindle
F1012 -     tool in spindle placed manually
F1013 -     tool in spindle placed from magazine

F1014 -     rotate magazine to returning tool
F1015 -     magazine error
F1016 -
F1017 -


F102  -     code of called tool
F104  -     pot of called tool in magazine

F106  -     code of returning tool
F108  -     pot of returning tool in magazine

F110  -     current magazine position (in front of spindle)
F112  -     target position for magazine rotation

F114  -     relative path for magazine rotation
F116  -
F118  -     magazine length/2

F120  -     HF120 format register
F122  -     start address of table
F124  -     table length
F126  -     mask register
F128  -     address register

F130  -     PF130 format register
```

```
F132   -     search from this line
F134   -     address register

F140   -     start address of PLC table

F150...F157 -     operand A: 8 byte
F158...F161 -     operand B: 4 byte
F162...F169 -     operand C: 8 byte

F170...F177 -     MUL170 registers
F180...F187 -     MW180 registers

F190...F193 -     magazine position (display at #190)


F500   -

...

F[501+2*MAGAZINE] end of magazine table

F[502+2*MAGAZINE] start address of PLC table
n      -     =0: empty spindle
             =1: tool in spindle placed manually
             =2: tool in spindle placed from magazine
             =4: cycles M6, M20 not closed


...

F[501+2*MAGAZINE+2*PLC_TAB] end address of PLC table


counters:

....
Q20    -     magazine rotation phase counter


H10    -     magazine rotation timer
H11    -     M6 timer


1-minute timers

M0     -     timer of magazine actions



PLC constants:

CONST037    -     rate/10000
CONST038    -     pulse number between two magazine positions
CONST039    -     magazine length


PLC axes:
```

3rd axis selected as PLC axis

```
modifications in connection with axis movement:

-     initializing
```

```
-     emergency stop handle
-     MON handle
-     magazine rotations

*/

/* start of module :001 */

:001              ;20-msec cyclical PLC module

      /* INITIALIZING */

I510              ;if first module :001 after power-on

      U520        ;mode selection from SW control panel
      U521        ;axis selection from SW control panel
      U522        ;increment selection from SW control panel
      U523        ;status selection from SW control panel
      U524        ;PLC buttons from SW control panel
      U525        ;R% from SW control panel
      D526        ;S% from SW control panel
      D527        ;F% from SW control panel
      U407        ;selecting EDIT mode
      UF0102      ;interrupt enabled
      ,0          ;0 to OP
      SRH060      ;S0
      SRH064      ;T0
      ,5          ;5 to OP
      SRH062      ;M5
      ,11         ;11 to OP
      SRH063      ;M11
      ,9          ;9 to OP
      SRH070      ;M9

                  ;***********register storing for search for new tool
      LRP039      ;magazine length
      /2          ;divided by 2
      SF118       ;storing
      .0002       ;word
      SF120       ;storing format register
      .0500       ;start address of table
      SF122       ;defining start address
      LRP039      ;magazine length
      *2          ;
      +2          ;table length
      SF124       ;defining length
      +500
      BCD         ;start address of PLC table
      SF140       ;start address of PLC table
      .3FFF       ;mask
      SF126       ;defining mask
                  ;***********register storing for
                  ;returning tool
      .0102       ;bidirectional search, word
      SF130       ;entering format

      .0004       ;4 bytes
      SF170       ;writing into format register MUL170
      .0150       ;start address of multiplicand (A)
      SF172       ;storing address register
```

```
      .0158         ;start address of multiplicator (B)
      SF174         ;storing address register
      .0162         ;start address of product (C)
      SF176         ;storing address register

      .0004         ;no decimal point, 4 bytes
      SF180         ;storing format register MW180
      .0001         ;writing at macro variable
      SF182         ;storing segment register
      .0190         ;at macro variable #190
      SF184         ;storing index register
      .0190         ;start address of magazine position
      SF186         ;storing address register

      U632          ;3rd Axis from PLC


Z                   ;end of condition
                    ;first module :001 after power-on

***************************************************


F0113               ;if activate EMERGENCY STOP state

 Y000               ;if spindle enabled
      D651          ;orientation request off
      U654          ;direct 1st spindle command signal transfer
      ,0            ;0 to OP
      SRH061        ;storing spindle JOG command signal register
 Z                  ;spindle enabled
                    ;**********************************change
      D920          ;3rd axis interpolator STOP
      D921          ;3rd axis interpolator strobe signal off
      D924          ;3rd axis run to reference position off
      U925          ;3rd axis interpolator RESET
      DF1005        ;magazine has no reference position
      UF1015        ;magazine error
      ,0            ;
      SQ20          ;clearing rotation phase counter
      DF1007        ;not under revolution
                    ;**********************************change
      C011          ;calling function RESET
      C012          ;calling start buttons RESET
      ,50           ;50 to OP (1 sec lag)
      ST00          ;storing emergency stop timer
      UF0104        ;check emergency stop timer
      DF0113        ;clearing activate EMERGENCY STOP state

Z                   ;end of condition
                    ;activate EMERGENCY STOP


***************************************************

F0105               ;if check MON timer

 T01                ;checking MON timer

  I003              ;if no emergency stop
      DF0105        ;check MON timer cleared
                    ;******************************change
   F1015            ;if magazine error
      U742          ;MAGAZINE ERROR on
```

```
   Z              ;magazine error
                  ;******************************change
  Z               ;no emergency stop

 E                ;else terminated
      D540        ;MON output off
      D506        ;MON lamp off
      DF0105      ;check MON timer cleared
 Z                ;end of condition clock still active

Z                 ;end of condition check MON timer



/* receiving magazine rotation command */

NF1007            ;if magazine not rotated

 F1004            ;if rotate magazine to called tool
      LF104       ;place of called tool in magazine
      SF112       ;target position for magazine rotation
      DF1015      ;no magazine error
      DF1004      ;clearing rotate magazine to called tool
      UF1007      ;magazine under rotation
      ,1          ;
      SQ20        ;clearing phase counter
 Z                ;end of condition rotate magazine to called tool

Z                 ;end of condition magazine not rotated


NF1007            ;if magazine not rotated

 F1014            ;if rotate magazine to returning tool
      LF108       ;place of returning tool in magazine
      SF112       ;target position for magazine rotation
      DF1015      ;no magazine error
      DF1014      ;clearing rotate magazine to returning tool
      UF1007      ;magazine under rotation
      ,1          ;
      SQ20        ;clearing phase counter
 Z                ;a end of condition rotate magazine to returning tool

Z                 ;end of condition magazine not rotated


/* magazine rotation */

F1007             ;if magazine under rotation

      LQ20        ;loading Q20 to OP
 =0               ;if no rotation
      DF1007      ;magazine not rotated
 Z                ;end of condition magazine not rotated

 =1               ;if 1st phase
  F1005           ;if magazine has reference position
      LF112       ;target position
   =LF110         ;if =current position
      DF1007      ;clearing magazine under rotation
      ,0
      SQ20        ;no duty
   E              ;if not =
    <LF110        ;if target position is less
```

```
                      ;then current position
       +LRP039        ;plus magazine length
     Z                ;end of condition less
       -LF110         ;mínus current magazine position
     >LF118           ;if greater then magazine length/2
       SF114          ;storing
       LRP039         ;magazine length
       -LF114         ;mínus stored value
       +1             ;in case of magazine rotated in negative direction
                      ;position is overrun by 1
                      ;to run from + direction
                      ;to position
       SF114          ;relative offset for magazine rotation
       UF1006         ;magazine rotation direction=1: negative
     E                ;if less
       -1             ;one subtracted
       SF114          ;relative offset for magazine rotation
       DF1006         ;magazine rotation direction=0: positive
     Z                ;end of condition greater then ...
       LF114          ;relative offset for magazine rotation
      =0              ;if 0
       ,21            ;
       SQ20           ;goto 21st phase
      E               ;not 0
       SF150          ;A lower word=relative offset
       ,0             ;
       SF152          ;A upper word=0
       LRP038         ;pulse number between two magazine positions
       SF158          ;B lower word=pulse number
       ,0             ;
       SF160          ;B upper word=0
       MUL170         ;multiplication C=A*B
       F1006          ;if magazine rotation direction=1: negative
       LF162          ;
       SF150          ;A lower word=C lower word
       LF164          ;
       SF152          ;A upper word=C upper word
       .FFFF          ;-1
       SF158          ;B lower word=-1
       SF160          ;B upper word=-1
       MUL170         ;multiplication C=A*B
       Z              ;end of condition negative rotation direction
       LF162          ;
       SRH160         ;3rd axis position command lower word
       LF164          ;
       SRH161         ;3rd axis position command upper word
       D920           ;3rd axis interpolator STOP
       U921           ;3rd axis interpolator strobe signal off
       D922           ;3rd axis move at rapid traverse rate
       U923           ;3rd axis incremental movement
       D924           ;3rd axis run to reference position off
       D925           ;3rd axis interpolator RESET off
       ,20            ;
       SQ20           ;goto 20th phase
     Z                ;end of condition =0
    Z                 ;end of condition =current position
   E                  ;if no reference position
;            D920     ;3rd axis interpolator START
       U920
       D921           ;3rd axis interpolator strobe signal off
       U924           ;3rd axis run to reference position
       D925           ;3rd axis interpolator RESET off
;            ,40      ;
       ,41
```

```
      SQ20          ;goto 40th phase
  Z                 ;end of condition magazine has reference position
      ,1            ;
 Z                  ;end of condition 1st phase



 =20                ;if 20th phase
  NI921             ;if 3rd axis received data
      U920          ;3rd axis interpolator START
      D921          ;3rd axis interpolator strobe signal off
      UQ20
  Z                 ;end of condition 3rd axis received data
      ,20           ;
 Z                  ;end of condition 20th phase

 =21                ;if 21st phase
  (I921             ;3rd axis interpolator terminated
  AI562)            ;and 3rd axis in position
      LRP038        ;pulse number between two magazine positions
      SRH160        ;3rd axis position command lower word
      ,0            ;
      SRH161        ;3rd axis position command upper word
      LRP037        ;rate constant
      SF150         ;A lower word=rate constant
      ,0            ;
      SF152         ;A upper word=0
      ,10000        ;constant
      SF158         ;B lower word=constant
      ,0            ;
      SF160         ;B upper word=0
      MUL170        ;multiplication C=A*B
      LF162         ;C lower word
      SRH162        ;writing rate command lower word
      LF164         ;C upper word
      SRH163        ;writing rate command upper word
      D920          ;3rd axis interpolator STOP
      U921          ;3rd axis interpolator strobe signal off
      D922          ;3rd axis move at feed rate
      U923          ;3rd axis incremental movement
      D924          ;3rd axis run to reference position off
      D925          ;3rd axis interpolator RESET off
                    ;increments last unit in positive direction
      UQ20          ;goto 22nd phase
  Z                 ;end of condition 3rd axis interpolator terminated
 Z                  ;end of condition 21st phase

 =22                ;if 22nd phase
  NI921             ;if 3rd axis received data
      U920          ;3rd axis interpolator START
      D921          ;3rd axis interpolator strobe signal off
      UQ20
  Z                 ;end of condition 3rd axis received data
      ,22           ;
 Z                  ;end of condition 22nd phase

 =23                ;if 23rd phase
  (I921             ;if 3rd axis interpolator terminated
  AI562)            ;and 3rd axis in position
      D920          ;3rd axis interpolator STOP
      DF1015        ;no magazine error
      LF112         ;loading target position
      SF110         ;=current position
      ,0
```

```
      SQ20         ;no duty
      DF1007       ;clearing magazine under rotation
  Z                ;end of condition 3rd axis interpolator terminated
      ,23          ;
 Z                 ;end of condition 23rd phase


/*
 =40               ;if 40th phase
  NI921            ;if 3rd axis received data
      U920         ;3rd axis interpolator START
      D924         ;3rd axis run to reference position off
      UQ20
  Z                ;end of condition 3rd axis received data
      ,40          ;
 Z                 ;end of condition 40th phase
*/


 =41               ;if 41st phase
  (I923            ;if reference position on 3rd axis
  AI562)           ;and 3rd axis in position
      D920         ;3rd axis interpolator STOP
      D924         ;3rd    axis    run    to    reference    position    off
***************************
      UF1005       ;reference position
      ,1           ;position
      SF110        ;storing current position
      ,1           ;
      SQ20         ;end of condition
  Z                ;goto 1st phase
      ,41          ;
 Z                 ;end of condition 41st phase vége


Z                  ;end of condition magazine under rotation



/* PLC axis reference point return */

Y924               ;if 3rd axis run to reference position
      LI055        ;REFZ switch
      SY552        ;3rd axis reference position switch
Z



/* MAGAZINE RESET */

(I505AV505)        ;if MAGAZINE RESET button pressed
 F1007             ;if magazine under rotation
      ,0           ;
      SQ20         ;zeroing phase counter
      DF1005       ;magazine has no reference position
      DF1007       ;clearing magazine under rotation
      UF1015       ;magazine error
      D920         ;3rd axis interpolator STOP
      U921         ;3rd axis interpolator strobe signal off
      D924         ;3rd axis run to reference position off
      D925         ;3rd axis interpolator RESET off
 Z                 ;end of condition magazine under rotation
Z                  ;end of condition MAGAZINE RESET button pressed
```

```
/* displaying magazine position */

      LRH110      ;3rd axis current position lower word
      SF190       ;storing
      LRH111      ;3rd axis current position upper word
      SF192       ;storing
      MW180       ;writing at #190


J1                ;end of module :001


/* end of module :001 */

/* selecting M codes */

:003              ;M code selection

=6                ;if equals to 6
      ,1          ;1 to OP
      SQ01        ;storing phase counter M06, M20
      DF0131      ;disabling tool replacement execution
                  ;function execution starts from here
      DF1010      ;not spindle empty command: not M20, but M6
      UF0120      ;executable M code found
      G004        ;goto label :004
Z                 ;end of condition equals to 6

=20               ;if equals to 20
      ,1          ;1 to OP
      SQ01        ;storing phase counter M06, M20
      DF0131      ;disabling tool replacement execution
                  ;function execution starts from here
      UF1010      ;spindle drift command: M20
      UF0120      ;executable M code found
      G004        ;goto label :004
Z                 ;end of condition equals to 20

********************************************

/* function execution */


      /* M6, M20 execution */

F0131             ;if M6 execution enabled,
                  ;and function execution start

      LQ01        ;loading Q01 to OP
 =0               ;if no M6
      DF0131      ;disabling M6 execution
      UF0132      ;enabling T execution
 Z                ;end of condition no M6


 =1               ;if 1st phase: test
  I551            ;if interpolator terminated
      DF0102      ;disabling interrupt
      C021        ;state set before replacement cycle
   (Y733          ;If READ ERROR
   OY740          ;or REPLACEMENT CYCLE NOT CLOSED
   OY732)         ;or WRITE ERROR
   E              ;if OK
```

```
    ((F1000         ;if incoming T code
    ANF1001         ;and new T not=T in spindle
    ANF1010)        ;and command M6
    O(F1010         ;or spindle empty command: M20
    ANF1011))       ;and spindle not empty
      LRH070        ;loading coolant state register
                    ;to OP
     =9             ;if state M9
      ,3            ;3 to OP
      SQ01          ;storing phase counter Q01
     E              ;else state M8
      C007          ;preparing coolant stop
      UQ01          ;incrementing phase counter Q01
     Z              ;and of condition state M9
    E               ;else if no incoming T code ...
      C022          ;decoding flags and exit
                    ;*******************************************exit
    Z               ;end of condition incoming T ocde ...
   Z                ;end of condition READ ERROR ...
  Z                 ;end of condition interpolator terminated
      ,1            ;1 to OP
Z                   ;end of condition 1st phase

=2                  ;if 2nd phase
      LQ06          ;loading phase counter M8, M9
  =0                ;command M9 executed
      C008          ;resetting coolant code
      UQ01          ;incrementing phase counter Q01
  Z                 ;end of condition command M9 executed
      ,2            ;2 to OP
Z                   ;end of condition 2nd phase

=3                  ;if 3rd phase
      LQ05          ;loading phase counter M3,M4,M5,M19
      SF032         ;saving spindle rotation (M3, M4, M5, M19)
                    ;phase counter Q05
  >0                ;rotation command waiting
      LF018         ;loading rotation code register
  E                 ;else no rotation command change
      LRH062        ;loading 1st spindle
                    ;rotation state register
  Z                 ;rotation command waiting
      SF030         ;saving rotation code
      DF0122        ;M5 from program
      ,19           ;19 to OP
      SF018         ;rotation code into register M19
      ,1            ;1 to OP
      SQ05          ;storing phase counter M3,M4,M5,M19
      UF0135        ;enabling
                    ;spindle rotation execution
      UQ00          ;incrementing FIN counter
      UF0103        ;disabling resetting
                    ;enabling interrupt
      UQ01          ;incrementing phase counter
      ,3            ;3 to OP
Z                   ;end of condition 3rd phase

=4                  ;if 4th phase
      LQ05          ;loading phase counter M3, ... M19
  =0                ;command M19 executed
      DF0135        ;disabling spindle rotation execution
      LF030         ;loading saved rotation code
      SF018         ;resetting rotation code register
      LF032         ;loading spindle rotation (M3, M4 M5, M19)
```

```
                        ;phase counter Q05
      SQ05              ;
      DF0103            ;disabling resetting
                        ;enabling interrupt off
   F1011                ;if empty spindle
    (NF1010             ;if M6
    AF1002)             ;and place tool manually
      ,60               ;60 to OP
      SQ01              ;storing phase counter
      UF0112            ;activate STOP state
                        ;*************************************manual placement
    Z                   ;end of condition place tool manually
    (NF1010             ;if M6
    AF1003)             ;and place tool form magazine
      ,20               ;20 to OP
      SQ01              ;storing phase counter
                        ;**********************************automatic replacement
                        ;**********************************empty spindle-tool in
    Z                   ;end of condition place tool from magazine
   E                    ;spindle not empty
    F1012               ;tool in spindle placed manually
      UF0112            ;activate STOP state
      UQ01              ;goto 5th phase
                        ;***************************************manual removal
    E                   ;tool on spindle placed from magazine
      ,20               ;20 to OP
      SQ01              ;storing phase counter
                        ;**********************************automatic replacement
                        ;**********************************tool out-tool in
                        ;**********************************or tool out
    Z                   ;end of condition ... placed manually
   Z                    ;end of condition empty spindle
  Z                     ;end of condition command M9 executed
      ,4                ;4 to OP
 Z                      ;end of condition 4th phase

 =5                     ;if 5th phase: test
  Y471                  ;if STOP state
      LRH064            ;loading T in spindle to OP
      BCD               ;binary BCD conversion
      SRH092            ;to tool out message register in decimal form
      U702              ;requesting T-index message TOOL OUT
      UQ01              ;increasing phase counter
  Z                     ;end of condition STOP state
      ,5                ;5 to OP
 Z                      ;end of condition 5th phase

 =6                     ;if 6th phase
  (I702                 ;if TOOL OUT T
  AY470)                ;and START
      D702              ;clearing message TOOL OUT T
      ,0                ;0 OP-ba
      SRH064            ;T in spindle
      SF500             ;tool table note
      UF1011            ;empty spindle
      DF1012            ;tool in spindle not placed manually
      DF1013            ;tool in spindle not placed from magazine
   (NF1010              ;if M6
   AF1002)              ;and place tool manually
      ,60               ;60 to OP
      SQ01              ;storing phase counter
      UF0112            ;activate STOP state
                        ;***************************************manual placement
   Z                    ;end of condition place tool manually
```

```
   (NF1010          ;if M6
   AF1003)          ;if place tool from magazine
      ,20           ;20 to OP
      SQ01          ;storing phase counter
                    ;***********************************automatic replacement
                    ;*****************************************spindle
                    ;*****************************************empty-tool in
   Z                ;end of condition place tool from magazine
   F1010            ;if spindle empty command: M20
      ,0            ;empty spindle
      C023          ;exit tool replacement
                    ;*******************************************exit
   Z                ;end of condition spindle empty command: M20
  Z                 ;end of condition TOOL OUT T...
      ,6            ;6 to OP
 Z                  ;end of condition 6th phase


 =20                ;if 20th phase
  NF1015            ;no magazine error
   NF1007           ;if magazine not rotating
    (NF1010         ;if M6
    AF1003)         ;and place tool from magazine
      LF104         ;place of called tool in magazine
     =LF110         ;current magazine position (in front of spindle)
      ,2            ;
      SH11          ;
      ,40           ;
      SQ01          ;goto 40th phase
                    ;arm manipulation: removing tool from spindle
                    ;and from magazine
                    ;***********************************tool out, in branch   Z
     E              ;if not equal
      U743          ;MAGAZINE POSITION ERROR on
     Z              ;end of condition current...
    Z               ;end of condition M6 ...
    (F1010          ;if M20
    O(NF1010        ;or M6
    AF1002))        ;and place tool manually
      C020          ;searching empty pot
     (Y736          ;if SEARCH ERROR WITH P
     OY737)         ;or NO EMPTY POT
     E              ;else
      UF1014        ;rotate magazine to returning tool
      UQ01          ;
                    ;********************************tool out branch
     Z              ;end of condition SEARCH ERROR ... feltétel vége
    Z               ;end of condition M20...
   Z                ;end of condition magazine not rotating ...
  E                 ;magazine error
      U742          ;MAGAZINE ERROR on
  Z                 ;end of condition no magazine error
      ,20           ;20 to OP
 Z                  ;end of condition 20th phase


 =21                ;if 21st phase
  NF1015            ;if no magazine error
   (NF1007          ;if magazine not rotating
   ANF1014)         ;and rotate magazine to returning tool
                    ;command received
      LF108         ;pot of returning tool in magazine
    =LF110          ;current magazine position (in front of spindle)
      ,2            ;
```

```
      SH11         ;
      UQ01         ;arm manipulation starts for placing tool back
    E
      U743         ;MAGAZINE POSITION ERROR on
    Z              ;
   Z               ;end of condition magazine not rotating ...
  E                ;magazine error
      U742         ;MAGAZINE ERROR on
  Z                ;end of condition no magazine error
      ,21          ;
 Z                 ;end of condition 21st phase


 =22               ;if 22nd state
  H11              ;if timer not terminated
  E                ;terminated
                   ;end of arm manipulation tool placed back
    NF1011         ;if spindle not empty
      LF500        ;loading tool code in spindle to OP
      SFI134       ;writing in tool table
    Z              ;end of condition spindle not empty
   (F0080          ;if syntax error,
   OF0082)         ;or not decimal number
      U732         ;WRITE ERROR
   E               ;if OK
      ,0           ;0 to OP
      SRH064       ;T in spindle
      SF500        ;note in tool table
      UF1011       ;empty spindle
      DF1012       ;tool in spindle not placed manually
      DF1013       ;tool in spindle not placed from magazine
    (NF1010        ;if M6
    AF1002)        ;and place tool manually
      UF0112       ;requesting STOP state
      ,60          ;
      SQ01         ;goto 60th phase
                   ;*********************************goto manual  replacement
    Z              ;end of condition M6 ...
    F1010          ;if M20
      ,0           ;empty spindle
      C023         ;exit tool replacement
                   ;*********************************exit
    Z              ;end of condition M20
   Z               ;end of condition syntax error ...
  Z                ;end of condition timer terminated
      ,22          ;
 Z                 ;end of condition 22nd phase


 =40               ;if 40th phase
  H11              ;if timer not terminated
  E                ;terminated
                   ;end of arm manipulation:
                   ;tool removed from spindle and from magazine
   F1011           ;if empty spindle
      ,2           ;
      SH11         ;
      ,42          ;
      SQ01         ;arm manipulation starts for placing tool back
                   ;********************************
   E               ;if not empty
      LF102        ;code of called tool
      A.C000       ;keeping width code, cutting tool number
      SFI128       ;clearing called tool from table
```

```
   (F0080        ;if syntax error,
   OF0082)       ;or not decimal number
     U732        ;WRITE ERROR
   E             ;if OK
     C020        ;searching empty pot
    (Y736        ;if SEARCH ERROR WITH P
    OY737)       ;or NO EMPTY POT
    E            ;else
     LF108       ;pot of returning tool in magazine
     =LF110      ;if equal to current magazine position
                 ;goto arm manipulation
     ,2          ;
     SH11        ;
     ,42         ;
     SQ01        ;arm manipulation starts for placing tool back
                 ;********************************
      E          ;if not, magazine must be rotated
     UF1014      ;rotate magazine to returning tool
     UQ01        ;
     Z           ;end of condition if equal ...
    Z            ;end of condition SEARCH ERROR
   Z             ;end of condition syntax error
  Z              ;end of condition empty spindle
 Z               ;end of condition timer terminated
     ,40         ;
Z                ;end of condition 40th phase


=41              ;if 41st phase
 NF1015          ;if no magazine error
  (NF1007        ;if magazine not rotating
  ANF1014)       ;and rotate magazine to returning tool
                 ;command received
     LF108       ;place of returning tool in magazine
   =LF110        ;current magazine position (in front of spindle)

     ,2          ;
     SH11        ;
     UQ01        ;arm manipulation starts for placing tool back
   E
     U743        ;MAGAZINE POSITION ERROR on
   Z
  Z              ;end of condition magazine not rotating ...
 E               ;magazine error
     U742        ;MAGAZINE ERROR on
 Z               ;end of condition no magazine error
     ,41         ;
Z                ;end of condition 41st phase

=42              ;if 42nd phase
 H11             ;if timer not terminated
 E               ;terminated
                 ;end of arm manipulation tool replaced
     LF102       ;code of called tool
     A.C000      ;keeping width code, cutting tool number
     SFI128      ;clearing called tool from table
  (F0080         ;if syntax error,
  OF0082)        ;or not decimal number
     U732        ;WRITE ERROR
  E              ;if OK
   NF1011        ;if spindle not empty
     LF500       ;loading tool code in spindle to OP
     SFI134      ;writing in tool table
   Z             ;end of condition spindle not empty
```

```
    (F0080          ;if syntax error
    OF0082)         ;or not decimal number
      U732          ;WRITE ERROR
    E               ;if OK
      LF102         ;code of called tool
      SF500         ;note in tool table
      A.3FFF        ;cutting width code
      SRH064        ;displaying T code in spindle
      ,2            ;in spindle from magazine
      C023          ;exit tool replacement
                    ;*********************************exit
    Z               ;end of condition syntax error ...

   Z                ;end of condition syntax error ...
  Z                 ;end of condition timer terminated
      ,42           ;
 Z                  ;end of condition 42nd phase


 =60                ;if 60th phase: test
  Y471              ;if STOP state
      LF102         ;code of called tool
      BCD           ;binary BCD conversion
      SRH093        ;into tool in message register in decimal form
      U703          ;requesting T-indexed message TOOL IN
      UQ01          ;goto 62nd phase
  Z                 ;end of condition STOP state
      ,61           ;60 to OP
 Z                  ;end of condition 60th phase

 =61                ;if 61st phase
  (I703             ;if TOOL IN T on screen
  AY470)            ;and START
      LF102         ;code of called tool to OP
      SF500         ;note in tool table
      A.3FFF        ;cutting width code
      SRH064        ;displaying T in spindle
      D703          ;1st indexed message off
      ,1            ;tool in spindle placed manually
      C023          ;exit tool replacement
                    ;****************************************exit
  Z                 ;end of condition TOOL IN T on screen
                    ;and START
      ,61           ;61 to OP
 Z                  ;end of condition 61st phase


Z                   ;end of condition
                    ;M6 execution enabled


      /* executing T */

F0132               ;if T execution enabled

      LQ02          ;loading Q02 to OP
 =0                 ;if no T
      DF0132        ;T execution disabled
      UF0133        ;enabling
                    ;gear range change execution
 Z                  ;end of condition no T

 =1                 ;if 1st phase: test
      DF0102        ;disabling interrupt
```

```
      UF1000       ;incoming T ocde
      LF024        ;code of called tool
      HF120        ;search
   F0080           ;if search error
      U735         ;SEARCH ERROR WITH H,
   E               ;else search OK
    F0081          ;if data not found: MANUAL REPLACEMENT
      DF1001       ;new T not =T in spindle
      UF1002       ;place tool manually
      DF1003       ;clearing place tool from magazine
      LF024        ;code of called tool
      SF102        ;saving code of called tool
      ,0           ;
      SF104        ;pot of called tool in magazine
    E              ;if data found
      LF128        ;data adress
     =.0500        ;if tool in spindle
      UF1001       ;new T=T in spindle
      DF1002       ;clearing place tool manually
      DF1003       ;clearing place tool from magazine
      LFI128       ;loading code and width of called tool
     (F0080        ;if syntax error
     OF0082)       ;or not decimal number
      U733         ;READ ERROR,
     E             ;if OK
      SF102        ;saving code of called tool
      ,0           ;
      SF104        ;pot of called tool in magazine
     Z             ;end of condition syntax error
     E             ;if tool in magazine
      DF1001       ;new T not =T in spindle
      DF1002       ;clearing place tool manually
      UF1003       ;place tool from magazine
      DF1006       ;magazine not rotated to new tool
      LFI128       ;loading code and width of called tool
     (F0080        ;if syntax error
     OF0082)       ;or not decimal number
      U733         ;READ ERROR,
     E             ;if OK
      SF102        ;saving code of called tool
      LF128        ;tool address
      BIN          ;binary conversion
      -500         ;by subtracting magazine start address
      /2           ;generating line number
      SF104        ;pot of called tool in magazine
      UF1004       ;rotate magazine to called tool
                   ;***************************************
      Z            ;end of condition syntax error
     Z             ;end of condition tool in spindle
    Z              ;end of condition data not found
      DQ00         ;decrementing FIN counter
      UF0102       ;enabling interrupt
      ,0           ;0 to OP
      SQ02         ;clearing T phase counter
   Z               ;end of condition search error
      ,1           ;1 to OP
 Z                 ;end of condition 1st phase

Z                  ;end of condition
                   ;T execution enabled

****************************************

J0                 ;end of module :000
```

```
*********************************************

/* searching empty pot */

:020                ;module 20
      LF110         ;current magazine position (in front of spindle)
      *2            ;byte conversion
      +500          ;generating address
      BCD           ;BCD form for search
      SF132         ;search for empty pot starts from this address
      LF500         ;code and width of tool in spindle
      PF130         ;searching empty pot for tool of above width
    F0080           ;if search error
      U736          ;SEARCH ERRO WITH F
    E               ;else search OK
     F0081          ;if data not found
      U737          ;error message NO EMPTY POT
     E              ;data found
      LF134         ;number of found pot to OP
      BIN           ;binary conversion
      -500          ;by subtracting magazine start address
      /2            ;generating line number
      SF108         ;pot of returning tool in magazine
     Z              ;end of condition data not found
    Z               ;end of condition search error
R                   ;end



/* setting states before replacement cycle */

:021
      LFI140        ;reading 1st line of PLC table
(F0080              ;if syntax error
OF0082)             ;or not decimal number
      U733          ;READ ERROR
E                   ;if OK
 =0                 ;if empty spindle
      UF1011        ;empty spindle
      DF1012        ;tool in spindle not placed manually
      DF1013        ;tool in spindle not placed from magazine
 E                  ;not empty
  =1                ;if tool in spindle placed manually
      DF1011        ;spindle not empty
      UF1012        ;tool in spindle placed manually
      DF1013        ;tool in spindle not placed from magazine
  E                 ;if tool in spindle not placed manually
   =2               ;if tool in spindle placed from magazine
      DF1011        ;spindle not empty
      DF1012        ;tool in spindle not placed manually
      UF1013        ;tool in spindle placed from magazine
   E                ;else interrupted replacement cycle
      U740          ;REPLACEMENT CYCLE NOT CLOSED
   Z                ;end of condition tool in spindle placed from magazine
  Z                 ;end of condition tool in spindle placed manually
 Z                  ;end of condition empty spindle
      ,4            ;replacement cycle in progress
      SFI140        ;writing 1st line of PLC table
 (F0080             ;if syntax error
 OF0082)            ;or not decimal number
      U732          ;WRITE ERROR
 Z                  ;end of condition syntax error ...
```

```
Z                  ;end of condition syntax error ...

R



/* decoding flags and exit */


:022
F1011              ;if empty spindle
      ,0           ;
E                  ;if not empty
 F1012             ;if tool in spindle placed manually
      ,1           ;
 E                 ;if not placed manually
  F1013            ;if tool in spindle placed from magazine
      ,2           ;
  E                ;if not placed from magazine
      U741         ;RECORDING ERROR
  Z                ;tool in spindle placed from magazine
 Z                 ;tool in spindle placed manually
Z                  ;end of condition empty spindle
NY741              ;if recording OK
      C023         ;
Z                  ;end of condition recording OK
R

/* exit tool replacement */

:023
      SFI140       ;writing 1st line of PLC table
(F0080             ;if syntax error,
OF0082)            ;or not decimal number
      U732         ;WRITE ERROR
E                  ;if no error
      DF1000       ;no incoming T
      DF1001       ;new T not =T in spindle
      DF1002       ;clearing place tool manually
      DF1003       ;clearing place tool from magazine
      ,0           ;0 to OP
      SQ01         ;cleaing T phase counter (no action)
      DQ00         ;decrementing FIN counter
      UF0102       ;enabling interrupt
Z                  ;end of condition syntax error ...
R



/* PLC softkey labels */

:197
SPINDLE JOG,                ;Y500
X LOCK,                     ;Y501
Y LOCK,                     ;Y502
Z LOCK,                     ;Y503
FUNKC LOCK,                 ;Y504
MAGZN REST,                 ;Y505
MON,                        ;Y506
FSBS,                       ;Y507
```

```
$

/* end of PLC softkey labels */



/* PLC messages */

:198TOOL REPLACEMENT T,         ;Y700
RANGE,                          ;Y701
TOOL OUT T,                     ;Y702
TOOL IN T,                      ;Y703
,                               ;Y704
,                               ;Y705
,                               ;Y706
,                               ;Y707
$

/* end of PLC messages */



/* PLC error messages */
:199
SPINDLE REVOLUTION ERROR,       ;Y710
SPINDLE RISE/FALL ERROR,        ;Y711
SPINDLE ORIENTATION ERROR,      ;Y712
SPINDLE ROTATION REQUEST,       ;Y713
,                               ;Y714
,                               ;Y715
,                               ;Y716
,                               ;Y717
MACRO READ ERROR,               ;Y720
MACRO WRITE ERROR,              ;Y721
ADD ERROR,                      ;Y722
SUBTRACT ERROR,                 ;Y723
MULTIPLY ERROR,                 ;Y724
DIVIDE ERROR,                   ;Y725
COMPARE ERROR,                  ;Y726
EQUAL,                          ;Y727
LESS,                           ;Y730
GREATER,                        ;Y731
WRITE ERROR,                    ;Y732
READ ERROR,                     ;Y733
WRITE/READ ERROR,               ;Y734
SEARCH ERROR WITH H,            ;Y735
SEARCH ERROR WTIH P,            ;Y736
NO EMPTY POT,                   ;Y737
CHANGE CYCLE NOT TERMINATED,    ;Y740
RECORDING ERROR,                ;Y741
MAGAZINE ERROR,                 ;Y742
MAGAZINE POSITION ERROR,        ;Y743
,                               ;Y744
,                               ;Y745
,                               ;Y746
,                               ;Y747
LUBRICATION X,                  ;Y750
LUBRICATION Y,                  ;Y751
LUBRICATION Z,                  ;Y752
,                               ;Y753
,                               ;Y754
,                               ;Y755
,                               ;Y756
,                               ;Y757
```

```
$

/* end of PLC error messages */

/* PLC program identifier */

:200 RANDOM MAGAZINE HANDLE AND INCREMENTAL AXIS MOVEMENT FROM PLC
ON THE BASIS OF EXAMPLE.PLC PROGRAM$

/* END OF PLC program identifier */
```

ALPHABETICAL INDEX