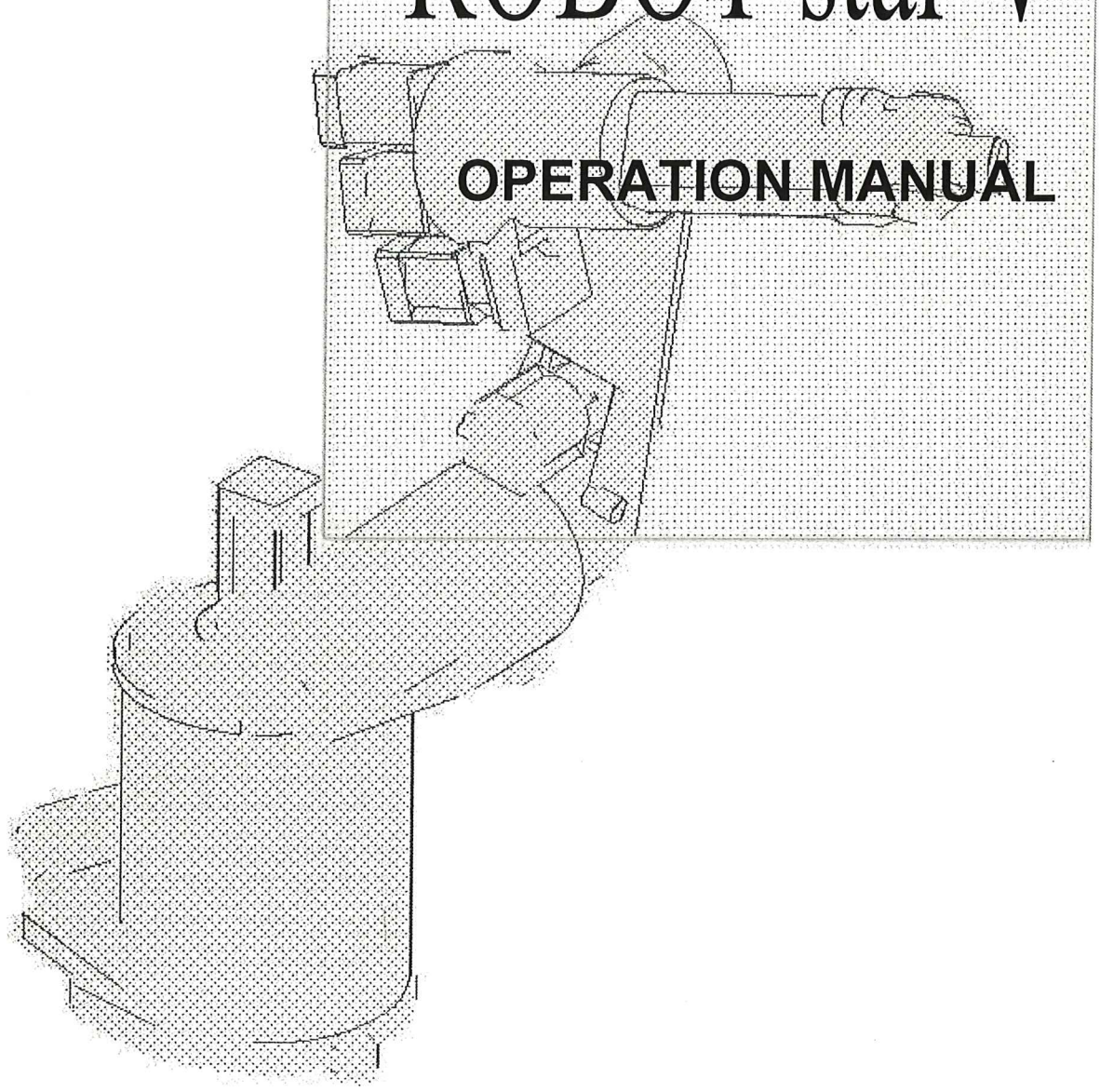
REIS ROBOTICS

ROBOT star V

OPERATION MANUAL



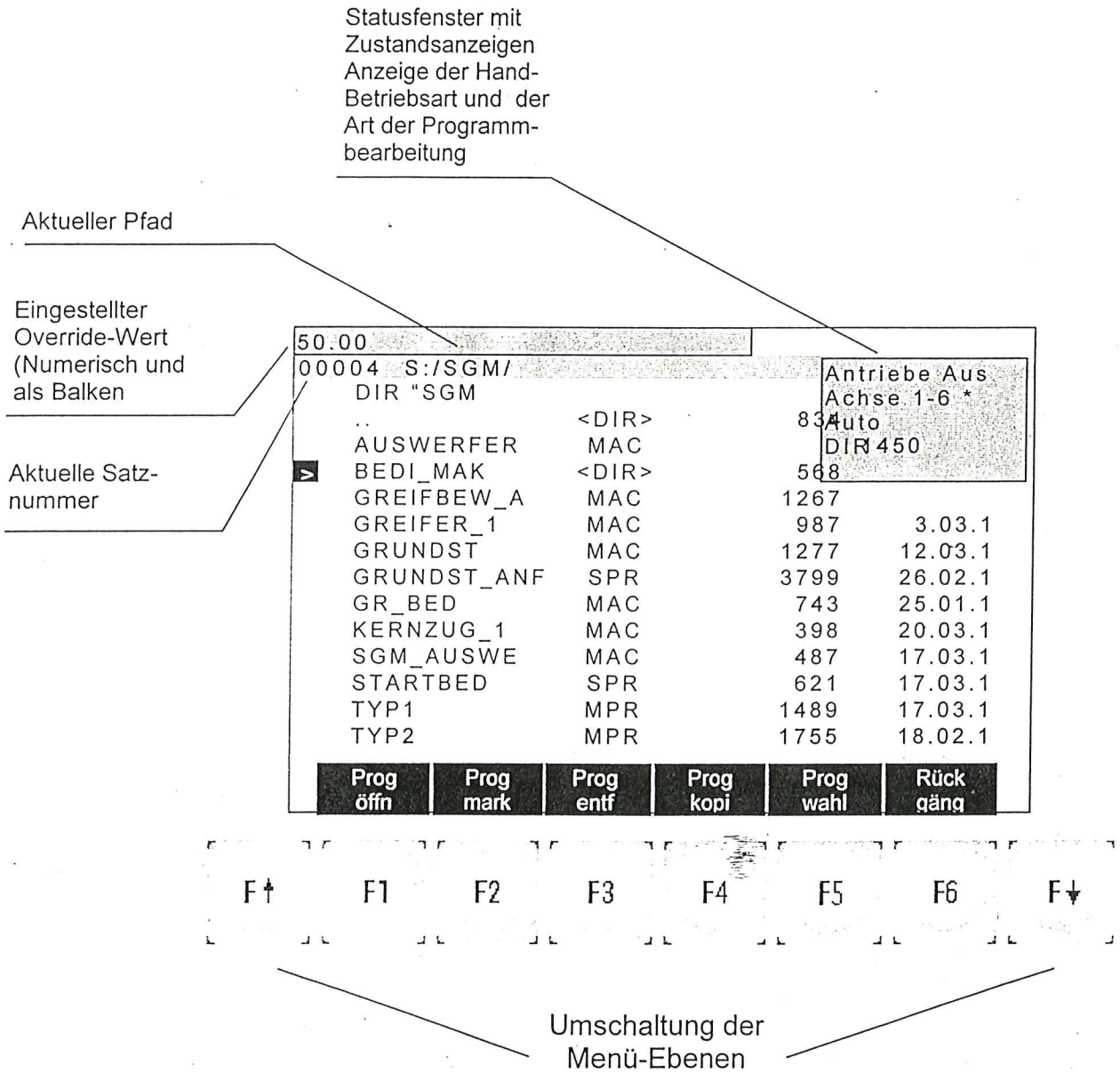
Version: RSV V7.x 1.2000 GB

REIS ROBOTICS

Operating manual ROBOTstar V

BDA-RSV/07.98/E

1.2 Display of the PHG
DIR-Mode



2 Operating modes of the control ROBOTstar V

2.1.6 FKT - mode

Is activated by operating the key "Funct" .
This mode is used for the following operations:

programming the system timer
modification of the program protection
interruption of archival storage
reset of the DNC-interface

2.1.7 CWIN - mode

Is activated by operating the key "Window".
With this mode the display can be divided into two windows, resp. the division can be reset.

3.1 Movement modes of the robot

3.1.1 Keyboard

Depending on the selected movement mode the robot can be moved via the movement keys (red double row on the right hand side of the keyboard). Movement is executed in dead man operation, i.e. the robot only moves so long as the movement key remains operated. Simultaneous operation of several key is generally possible, in order for instance to move several axes at the same time.

The speed of movement can even be changed during movement with the override regulator.

3.1.2 Mouse

For intuitive movement of the tool into any random direction it is possible to have connected a 6D-mouse. This mouse, depending on the point of fixture, must be calibrated. For this option as a standard feature there is provided a fixture point on the PHG and up to two fixture points directly at the tool. Direct guiding of the tool is possible with the mouse.

3.1.3 Position Control
Only effective with movement mode Cartesian

With the special key "I" under the override regulator it is possible to switch over to Position Control . The first line in the display of the PHG is represented in green and the number value that was adjusted with the override regulator before (e.g. 54.00 %) gets the unit millimeter (e.g. 54.00 mm). If now for instance the key "+ X" is operated, the TCP can be moved - by turning at the override regulator - into X-direction (to the right to X plus, to the left to X minus). The number value in the first line indicates by how many millimeters the TCP moves with one turn of the override regulator. The speed can directly be influenced by the rotation speed at the override regulator. With Position Control also the orientation of the tool can be changed.

Position Control is also possible in conjunction with the 6D-mouse.

With the special key "O" switch over to the override mode is executed again.

4.1 Program types of the ROBOTstar V

4.1.1 MPR Main program

One step after the other is processed in automatic operation. The last step - "END" - includes the command for return to step 1, i. e. the program will be repeated. In the test operating modes running in sequential step mode, return to step 1 is executed after the end and the program stops.

Subprograms (command "CALL"), main programs (command "PROGRAM") and macros can be called in a main program.

4.1.2 SPR Subprogram

A subprogram can be called from a main program, from another subprogram or from a macro. Processing is started with step 1. The subprogram is processed step by step. The last step „END" includes the command for return to the calling program. Return is made to the step following the call.

If a subprogram was selected by the operator, or if the subprogram was stopped and scrolling or another modification was made, no return will be possible with processing of the last step „END". The following error message is displayed:

Operating error 4017

Further subprograms and macros can be called in a subprogram. (Nesting depth12!)

4.1.3

4.1.4 MAC Macro

A macro can be called from a main program, a subprogram or another macro. The call of a macro is programmed like a system command. The commands in a macro are processed like in a subprogram, but no program change is executed. In the single step modes a macro call is processed like a single command.

A macro must not contain any positions!
Further macros and subprograms can be called in a macro. (Nesting depth 12!).

4.1.6 PLC PLC program

The control ROBOTstar V has an integrated PLC that is programmed like a robot program. When being once started, the PLC program is cyclically running in the background, independent from the selected operating mode and from state of the robot.

The total of commands approximately corresponds to the Siemens - PLC S5/100U.

The integrated PLC makes access to the same inputs and outputs like the robot program. Communication between PLC and robot program is ensured via system variable. Further system markers contain information about axis actual values of the robot, state of system inputs and system outputs, error states etc.

5.1 Programming of Movement Sequences

Movement sequence of the robot is realized via programmed positions in space. During program run the robot approaches the positions in such order as they are in the user program. When doing this, it always keeps the direct way from one position to the next.

5.2 Entering the actual position into the program

TOOL **(instruction set Move)**

Before a position can be taken over into a program, first of all a valid tool variable must be named. This is effected in the instruction set "Move" by selection of the command "TOOL" and subsequent input of the tool variable the name of which must begin with the letter "T" .

POSITION **(instruction set Pos)**

In order to enter a position, the TCP is moved to the required position in operating mode "AXIS 1 - 6" or "CARTESIAN". Additionally, orientation of the tool can be adjusted and optimized. For entering this position into the program it is sufficient to select the instruction set "Pos" and to press the key "ENTER". Thus, the position is taken over into the program as normal position.

The current actual position can be defined in the instruction set POS as follows:

- with F1 as normal position in robot coordinates #N
- with F2 as oscillation auxiliary point in robot coordinates #P

5.2.4 Velocities and accelerations

PTP_VELOC<value in %>
PTP_ACCEL <value in %>

PATH_VELOC <value in mm/sec>
PATH_ACCEL <value in %>

Speed and acceleration can be programmed for the movement modes PTP and CP. The corresponding instructions are to be found in the instruction set "Move".

The PTP-speed is programmed in percent. The percentage refers to the maximum axis speed of the robot determined in the machine data.

The PTP-speed is no dimension for the speed of the TCP. It only defines the speed by which the pilot axis moves. The pilot axis is the axis having the greatest movement path. The TCP moves more slowly or faster depending on its distance from the rotary point of this axis.

Acceleration for PTP-movements is also programmed in percent. The reference parameter is the maximum axis acceleration which is also defined in the machine data.

Acceleration is not only a dimension for the increase of velocity when starting from a position, but also the dimension for the deceleration by which the axes reduce the speed before reaching a programmed position.

The speed for CP-movements is programmed in millimeters per second. This speed is valid for the TCP.

The path acceleration is also valid for the TCP. It is programmed in percent. The maximum value is defined in a system variable.

6.2 CALC_REL <name of the vector variable>

With the command "CALC_REL" the relative shift between the actual position of the TCP and the next programmed position is calculated. The calculated shifting values will be stored in the vector variable which is named in the command.

Calculation of the vector is made as follows:

Relative vector = actual position - programmed position

6.3 Example:

```
S12
S13
S14
S15   POSITION   A
S16   CALC_REL   VEKTOR_1
S17   POSITION   B
S18   POSITION   C
S19   RELATIVE   VEKTOR_1
S20   POSITION   C
S21
.
S28   RELATIVE   _VNULL
S29   POSITION   X
```

When treating step 15 the robot approaches position A. Step 16 effects the calculation of the shifting vector between positions A and B (values of position A minus values of position B), and the calculated values will be entered into the variable "VEKTOR_1". Position B in step 17 serves for calculation only and is not approached. Subsequently the robot approaches position C in step 18. Displacement is not yet activated, this will only be effected in step 19.
In step 20 position C is approached once again, but shifted by the calculated vector. All following positions will also be approached shifted by the a.m. vector.
In step 28 the relative shift is switched off again. All the following positions won't be displaced any more.

Digital signals are used for communication between peripheral equipment and robot control.

User inputs provide the control with information about the peripheral conditions (e.g. finish message of a processing machine, rotary table in correct position, clamping device closed, deposit free etc.).

User outputs give information to the peripheral equipment and execute commutations (e.g. signal to a processing machine, that the robot is moving into the operating range of the machine, switching on or off a conveyor, switching on a spindle, open/close gripper, feed out pallet etc.)

Analog outputs are used for regulation of peripheral equipment by means of a variable direct voltage, e.g. speed of a machine, weld parameters etc.

Also direct voltages coming from peripheral equipment units can be interrogated in the user program.

7.1.1 Allocation of the binary inputs and outputs to system variables

by means of the example of the user inputs

Variable	Byte - number			
_IBIN_IN[1]	3 Bit 76543210	2 Bit 76543210	1 Bit 76543210	0 Bit 76543210
_IBIN_IN[2]	7 Bit 76543210	6 Bit 76543210	5 Bit 76543210	4 Bit 76543210
_IBIN_IN[3]	11 Bit 76543210	10 Bit 76543210	9 Bit 76543210	8 Bit 76543210
_IBIN_IN[4]	15 Bit 76543210	14 Bit 76543210	13 Bit 76543210	12 Bit 76543210
_IBIN_IN[5]	19 Bit 76543210	18 Bit 76543210	17 Bit 76543210	16 Bit 76543210
_IBIN_IN[6]	23 Bit 76543210	22 Bit 76543210	21 Bit 76543210	20 Bit 76543210
_IBIN_IN[7]	27 Bit 76543210	26 Bit 76543210	25 Bit 76543210	26 Bit 76543210
_IBIN_IN[8]	31 Bit 76543210	30 Bit 76543210	29 Bit 76543210	28 Bit 76543210
_IBIN_IN[9]	35 Bit 76543210	34 Bit 76543210	33 Bit 76543210	32 Bit 76543210
_IBIN_IN[10]	39 Bit 76543210	38 Bit 76543210	37 Bit 76543210	36 Bit 76543210

7.3 Inquiry of digital input signals

The input signals are treated in the same manner like the output signals. Bits are set resp. reset according to the node number via the CAN bus. I.e., if a connection is applied with 24 volt on the hardware side, then the corresponding bit is set, if 0 volt are applied, it is reset.

A different reaction on input signals is possible:
Waiting for a signal in the program sequence is possible or a conditional branch can be executed depending on an input signal.

7.3.1 Waiting for input signal

Before a part can be removed from a processing machine, the control waits until the signal "processing finished" is given. Acc. to circuit diagram this signal is active on input 0.5.

The following must be programmed:

WAIT_BIT #EINGANG, Pegel: **1**, Byte: **0**, Bit_Nr: **5**, Max_Zeit: **0.0**, Marke: **X**

The program will stop at this command so long until 24 volt will be applied at the input 0.5. Then, processing of the program will be continued with the next step. The parameter "Max_Zeit" (Max_Time) is indicated with 0.0 here, which means a waiting without time limitation.

Another value can be programmed in the parameter "Max_Zeit":

WAIT_BIT #EINGANG, Pegel: **1**, Byte: **0**, Bit_Nr: **5**, Max_Zeit: **10.0**, Marke: **ABC**

Here, the program also stops and waits for the signal. When the signal is active within the indicated time (10 seconds in the example), processing is continued with the next step. If the programmed time is exceeded without the signal being given, then a program branch is executed to that label that was indicated in the last parameter (ABC in the example).

7.4 Control of the analog outputs

ANA_OUTP <rated voltage in volts, number of the analog output>

In the instruction set Peri the output of a direct voltage can be programmed at an analog output with this command. After selection of this command first the voltage has to be indicated in Volts (-10 volt to +10 volt) and then the number of the analog output has to be entered where the voltage shall be applied.

Voltage can be entered via the numeral keys, but it can also be taken from a real constant or from a real variable; in this case the first parameter to be indicated is the name of the constant or of the variable.

7.5 Inquiry of analog inputs

ANA_INP <number of the analog input, name of the destination variable>

The analog voltage which is applied to an analog input can be filed in a real variable for further treatment.

Selection of the command "ANA_INP" is effected from the function group "Peri". After selection, first of all the number of the analog input the analog voltage shall be taken over from has to be entered, and then, the name of the variable where the voltage value is to be stored has to be indicated.

Example:

```
S1    MPR          ANALOG_TEST
S2    LOK_VAR      RDRUCK
.     .            .     .
S14   POSITION
S15   SCHR_BIT #AUSGANG, 1, 0, 4
S16   MARKE        WIEDERHOLE
S17   ANA_EING     2,RDRUCK
S18   TESTE #VARIABLE, RDRUCK, #<, 5.0, WIEDERHOLE
S19   U_PROG       BESCHICHTE
.     .            .     .
S39   ENDE
```

In step 16 a dosing system is switched on via output 0.4 which gradually builds up the required operating pressure for coating. The current pressure is signalized to analog input 2 via an analog signal.
The applied analog voltage is loaded in the variable RDRUCK in step 17 and is compared with the minimum value 5.0 volt in step 18. The subprogram for coating is only called after this minimum voltage is applied at analog input 2, i.e. when the minimum pressure has been reached.

8.1 Change into a main program

PROGRAM <name of the main program>

The command "PROGRAM" from the command group „Prog." effects a branch from a main program to step 1 of another main program. After selection of the command the name of the main program has to be entered change shall be effected to. If the program with the indicated name does not exist or if the indicated program is no main program, corresponding error messages will be emitted during run in automatic or test mode.

8.2 Call of a subprogram

CALL <name of the subprogram>

After selection of the command "CALL" from the command group „Prog" the name of the subprogram you want to branch to must be entered. In automatic or test mode branching is effected to step 1 of the indicated subprogram when processing this command. This subprogram will be treated, and after the last step of the subprogram the calling program will be returned to. Program treatment here is continued with the command following the subprogram call.

9 Programmed waiting time

WAIT <time in sec>

Delay times are programmed with the command "WAIT" of the instruction set "Contr" . As parameter that time has to be entered in seconds within which the program shall be stopped before the program cycle will be continued with the next command. The waiting time is effective in automatic mode and in all test operating modes.

10.1 Branch destination

LABEL <name of the destination mark>

The command for the branch destination is named "LABEL". It is selected from function group "Contr" and after input of a freely selectable name (e.g. LABEL_1) is taken over into the program with key "ENTER". Within one program, another name has to be entered for each command "LABEL". Since branches from one program to another are impossible, same names can be used in different programs.

10.2 Absolute branch

BRANCH <name of the destination mark>

For programming of an absolute (unconditional) branch the command "BRANCH" has to be selected and the name of the label (branch destination) has to be entered (e.g. LABEL_1"). The indicated label must be in the same program as the branch command. When this command is processed in automatic or test mode, processing of program is continued at that branch label that has the same name. If no label with the same name exists in the program, the corresponding error message will be given.

10.3.2 Test of a byte or of a variable

In order to test the content of a complete byte or a variable and to execute a branch depending from this, the command "TEST" from the group "Contr" must be selected. In principle this command compares the content of the byte resp. of the variable with a number and depending on the result executes the branch or not. After selection the following has to be entered:

Select what shall be tested.

#EINGANG	test of an input byte
#AUSGANG	test of an output byte
#MERKER	test of a marker byte
#VARIABLE	test of a variable

Input of the operand 1 (the number) the operand 2 shall be compared to

Selection of the branch condition

#=	branch when both operands are identical
#<>	branch when both operands are different
#<	branch when operand 1 is smaller than operand 2
#>	branch when operand 1 is higher than operand 2
#<=	branch when operand 1 is equal to or smaller than operand 2
#>=	branch when operand 1 is equal to or higher than operand 2

Enter the byte number, when input, output or marker are the parameters to the tested or enter the variable name, when a variable shall be tested.

Input of the branch destination

When a variable shall be tested, then as first operand its name can be indicated and as second operand the comparison parameter.

11 Variables

11.1.5 Tool variable

Consists of 16 components. The components 4, 8 and 12 contain the position of the TCP referring to the tool flange.

11.1.6 String variable

Contains a random character string. The length of the variable must be entered after the name in a bracket, e.g. STRING(20). The variable STRING might contain a maximum of 20 digits.

11.2 Definition of variables

Variables can be defined in any program. Variable definitions must always be made at the beginning of the program. Exceptions are commentary lines, they may stand in front of definitions.

With definition the range of validity (global or local) and the name of the variable are defined, the first character determining the type.

11.2.1 Global variables

A variable is defined as global variable with the command "VAR" from the command group "Var" .

e.g. **VAR** Name: **I123**

Global means, that access to this variable with the name I123 is possible an any program. The name of a global variable must exist only once as variable name in the complete program memory.

11.2.2 Local variables

The command "LOC_VAR" also from the program group "Var" defines a variable as local.

e.g. **LOC_VAR** Name: **IABC**

Local means, that access to this variable is only possible in that program where it is defined, or in the sub-programs that are called from this program.

12 Mathematical operations

For arithmetical operations the 4 fundamental operations are available. The commands are in function group "Math" . When executing the arithmetical commands, operand 1 (variable, constant or entered number) is calculated with the content of the destination variable and the result is written into the destination variable.

The commands for arithmetical operations are:

ADD <Quelle, Zielvariable> Zielvariable = dest. variable plus operand 1
SUB <Quelle, Zielvariable> Zielvariable = dest. variable minus operand 1
MUL <Quelle, Zielvariable> Zielvariable = dest. variable multiplied by operand 1
DIV <Quelle, Zielvariable> Zielvariable = dest. variable divided by operand 1
MODULO <Quelle, Zielvariable> Zielvariable =division rest of the division dest. variable by operand 1.
NEG negation of the variable content (negation of sign)

With the command "VEC_VALUE" the length of a straight distance can be determined, for instance.

Programming example:
At the end of a programmed linear movement the length of the distance A --> B shall be entered in the variable "RWEG".

```
POSITION      A              <start point>
INTERPOL      #CP_LIN
POSITION      B              <end point>
CALC_REL      V_BAHN
POSITION      A
VEC_VALUE     V_BAHN,RWEG
```

An integer variable contains a 32-digit binary number. This binary number (operand 2) can be linked logically with another number (operand 1). The operation is effected in such a manner that each bit of operand 1 is linked with the equivalent bit of operand 2. The result is again a 32-digit binary number being filed in operand 2.

Contrary to the arithmetical operations, the content of the named integer variable will be interpreted as a number without sign.

Operand 1 may be:

an integer constant	
an integer variable	
a decimal number,	which is entered via the numeral keys
a hexadecimal number,	After the number the letter "H" must be entered.
a binary number,	After the number the letter "B" must be entered.

The content of the variable can be linked via AND-function, OR-function and EXCL_OR-function. The result of the linkage is always in the destination variable (2nd operand).

14.1 AND-linkage

AND <integer value, variable name>

32 bits of one variable are linked simultaneously with the 32 bits of the 1st operand. When a bit of the integer variable is linked with 1 via AND, the value of this bit is taken over into the result. If a zero is in this bit, the result becomes zero. If a 1 is in this bit, the result becomes 1.

With an AND-operation with zero the result is always a zero, independent of the bit status.

The command "AND" is selected from instruction set "Log".

First the 1st operand has to be entered (the number resulting from the 32 individual bits). Then, the variable is named the content of which shall be linked with operand 1.

The command "NEG" only converts the sign of the variable content, i. e., the amount remains the same (conversion in two's complement). The command "INVERT" changes the content of the variable in one's complement, i.e., the amount changes. All bits being logic zero become logic one and vice versa.

14.5 Shifting operations

SHIFT_L <number of digits, variable name>

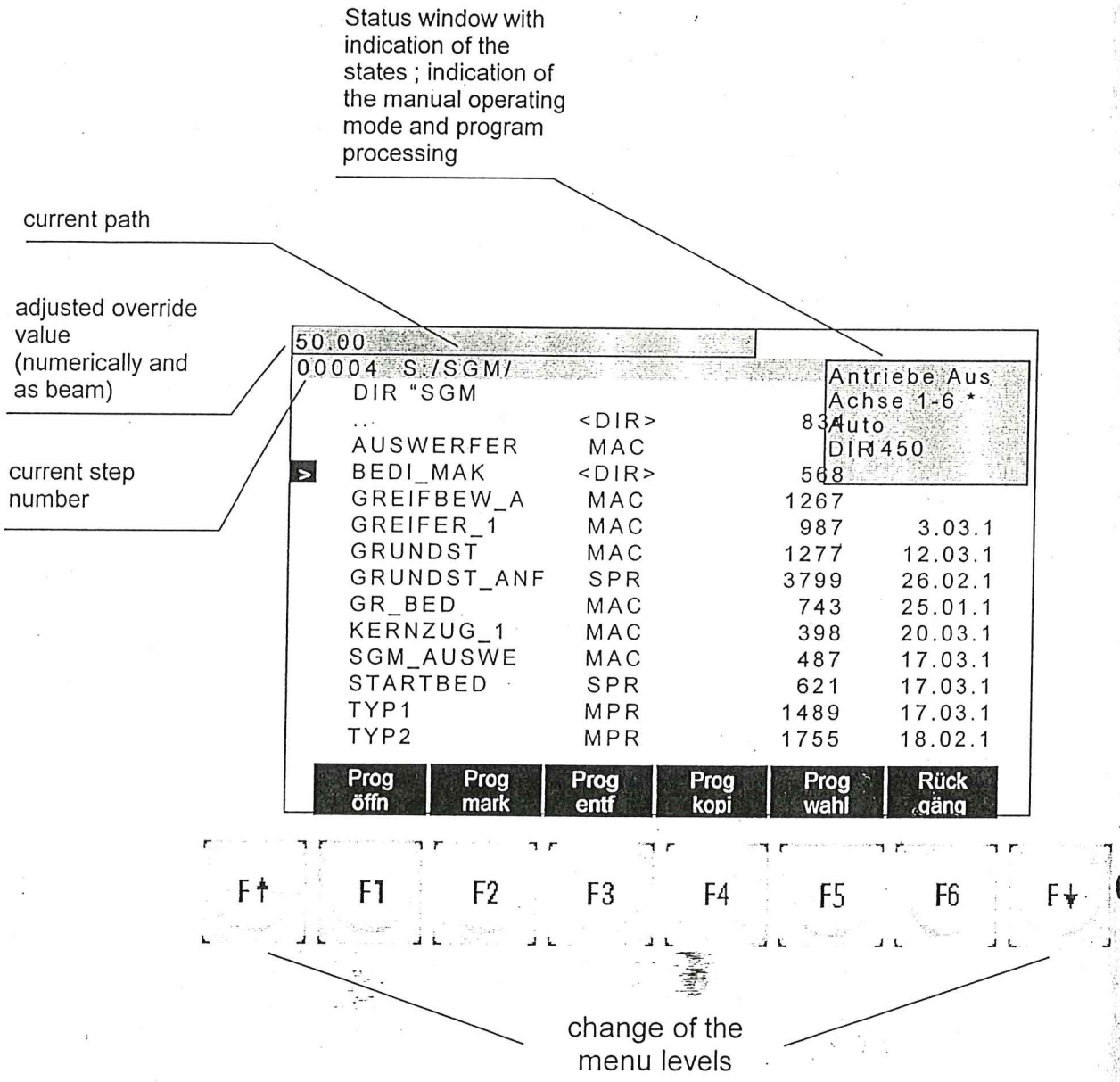
The 32 bits of an integer variable can be shifted to the left with this command. In the first parameter you must indicate by how many digits shifting shall be effected. Each digit corresponds to a multiplication by 2 (1 digit = *2; 2 digits = *4; 3 digits = *8; 4 digits = *16 a.s.o.). The bits which exceed the 32 bits due to shifting to the left will be ignored and the "gaps" at the right side resulting from shifting to the left will be filled up with zeros.
The highest value bit (2**31) also here is the sign bit.

SHIFT_R <number of digits, variable name>

The 32 bits of an integer variable are shifted by n digits to the right. Each digit corresponds to a division by 2 (1 digit = :2, 2 digits = :4, 3 digits = :8 a.s.o.). The bits falling away due to the shift to the right are ignored and the "gaps" resulting on the left side will be filled up with zeros.

After selection of "SHIFT_R" in the 2nd menu of instruction set "Log" with function key F2 the first parameter to be indicated is the number of digits by which shifting shall be effected. After a comma the name of the variable is indicated the content of which is manipulated.

Display of the PHG DIR-Mode



If the key "SHIFT" was operated prior to pressing the arrow key, the text jumps to the right resp. left hand side by 20 characters. With key "ALT" and subsequent operation of the arrow key on the left the display jumps to the beginning of the lines. "ALT" and arrow key on the right places the end of the lines into the middle of the window.

Program protection: The operating possibilities of programs / directories are represented by letters:

L/A/E/S/U

L	The program / directory can be deleted
A	The program can be executed in automatic or test mode
E	Commands can be edited in the program
S	The structure of the program / directory can be modified (commands can be inserted, deleted or invalidated)
U	The program / directory can be renamed.

If one of the letters is replaced by a slash the corresponding operation then is locked.
e.g.: -/A/E/-/-
The program marked in this way can now be executed and edited.

Copying of protected programs remains possible.

Remark concerning global definitions

If variables or constants are defined in a program, "Var" stands after the program protection in the directory.
If the program stands in a sub-directory, then the complete path, i.e. all superimposed directories are marked with „Var".

Operation in DIR-mode

Selection of the possible operation functions is ensured via the function keys under the display. The keys on the left and right hand side of the function keys are used for switch-over between the menu levels.

First menu

Prog open

Establish new program or sub-directory.
After selection the name of the new program must be entered and be finished with ENTER ; then, the program type is selected with the function keys. After operation of the key ENTER the new program resp. directory is established.

Prog mark

Mark program or directory besides the cursor or cancel its marking.
Marked programs (not directories!) can be copied into another directory or on disk.
Marked programs and directories can be deleted; directories must be empty!

Prog del

Deletion of programs and/or directories. Directories must be empty!
If no marking exists, the following message is displayed:

Acknowledge the program to be deleted:

In the input line below there is indicated the path and name of the program where the cursor stands. By operation of the key ENTER this program is deleted. By input of another name (perhaps with indication of the path) a random other program can be deleted.

If programs / directories are marked, the following message is given:

Do you really want to delete
all marked programs?

After selection of the menu item "YES" with the function key F1 the marked programs are **only deleted in the current directory**

Second menu

Find repl

Search of a program or directory name **only in the current directory.**
Replacement of character strings in not possible in the DIR-mode!
If programs are marked, the indicated character string is searched for in the marked programs. This character string can be replaced by another one, if it is not component of a program name, a command word, a describer or a system equate.

Disk read

Read in of programs from disk (drive A:).
After selection of the menu item the programs are indicated in the master directory of the disk (no subdirectories!)
The programs to be read in are marked and then copied into the desired directory on the control with "Prog kopi".
If a selected program name already exists in the destination directory, you are asked whether the existing program shall be overwritten.
Overwriting is only possible for programs that are not protected against deletion!

Disk frmt

Formatting of a disk in drive A:
After selection the following message is given:

Do you really want to
format disk?

Formatting is started with function key F1 "YES" , with function key F2 "NO" selection is reset.

Mem edit

Not for general use! Only for customer service!

Third menu

Flsh read	Not for general use!
Flsh frmt	Not for general use!

Operation in RUN-Mode

The RUN-mode is activated with the white operating mode key "Run".
The RUN-mode determines in which operating mode program processing will be made (automatic or test mode). Further possibilities of operation are the following ones:

Synchronization of the robot
Approaching reference points
Start of the integrated PLC
Test of the integrated PLC

Selection of the possible operating modes is made with the function keys under the display. The keys on the left and right hand side of the function keys are used for change between the menu levels.

For the TEST-operating modes distinction is made between single step mode and sequential step mode

Single step mode:
Always the current command the cursor points to is executed. When processing of the step is finished, the cursor jumps to the next step, the start key must be released and pressed again, in order to continue processing. If the current step is a programmed position, it will be directly approached in dead man operation, i.e. the movement of the robot stops as soon as the START key is let off. In this manner any position in the program can be directly approached in single step mode after selection of the corresponding step.

Sequential step mode:
a) Set key selector switch on the teach pendant (PHG) to "Auto" or "Auto-Test" (position right or left h.s.):
After operation of the "START" key the program automatically runs to the end and then stops after a return to the beginning of the program. During cycle the program can be stopped with the key "STOP" and started again with the key "START".

b) Set key selector switch on the teach pendant (PHG) to "Cal/Sync" (setting) (medium position)
Also in this case, one step after the other is automatically processed, but programmed positions are approached in dead man operation, i.e., the robot executes movements only so long as the key START will be kept pressed. At the end of the main program the program is also stopped after the return to step 1.

In some TEST-operating modes the input signals must be simulated, i.e., when a command is processed, that has access to a user input, then the function keys F1 and F2 are activated. With F1 simulation of level 0, with F2 simulation of level 1 is possible.

Test 5

Single step mode; program execution by the executer.
The selected operating mode for the interpreter remains (AUTO; TEST1 to TEST4)
The last line of the status window indicates the mode EXEC. Processing of the current step is made with the key "EXEC". After processing the cursor jumps to the next step. No return to step 1 is executed at the end of a main program. Some commands cannot be executed by the executer:

WAIT_BIT
POSITION
velocities
accelerations
OSCILLATION
WELDING
etc.

Auto

With activated operating mode "AUTO" the current program can be started via the key "START", starting from the current step. The program is processed forward step by step, all valid instructions being executed. During run the program can be stopped at any position with the key "STOP" and restarted with the key "START". When the program has been stopped with the key "STOP", the robot can be moved in operating mode "HAND" after having pressed the key „Quit/Esc". If then the program is started again via "START", it will be continued exactly from that position where interruption took place, i.e. when the robot had been moved, it first of all returns to the position where the movement being just executed had been interrupted and continues the interrupted movement sequence from there. This, however, is no longer valid when the program was scrolled, when a step had been selected via the key "Step", or when the program was modified.

After stopping of the program in operating mode "AUTO" it can be continued in one of the test operating modes and vice versa any time.

Test 6

Single step mode (program execution by the interpreter)
Processing is the same as in TEST 1, but macros are also run through in single step mode.

Test PLC

Switching on resp. off the control window in order to check a selected running PLC program.

Additionally to the displayed PLC commands, the corresponding interlinkage results and the contents of the accumulators are indicated. Commands that are omitted are marked with "NOP".

Operation in COOR-Mode

The COOR- mode is activated by the white operating mode key "Coord".
COOR - mode determines the manner of manual movement of the robot. Robot movement generally is made in dead man operation, i.e. the robot only moves so long as the corresponding movement key will be operated. Robot speed can be influenced with the override regulator during movement.

First menu

AXIS 1 - 6

Axes 1 - 6 can be moved in negative or positive direction via the movement keys "1-", "1+" to "6-", "6+". If the keys of several axes are operated at the same time, the corresponding axes are moving simultaneously.
During movement the actual values of the individual axes are indicated in a window; for rotation axes in degrees, for linear axes in millimeters

tXYZ rABC

Movement of the tool in the basic coordinate system.
Contrary to sub-operating mode „Axis 1 - 6" the TCP (Tool Center Point) moves on a straight line during cartesian movement and the orientation of the tool remains constant during the movement. The TCP can be moved in the basic coordinate system via the keys X-(1-), X+(1+), Y-(2-), Y+(2+), Z-(3-) and Z+(3+).
The orientation angles A, B , and C of the tool can be changed via the keys 4-, 4+, 5-, 5+, 6- and 6+. The TCP keeps its cartesian X, Y, and Z values during the orientation change (presumed that the correct tool data are activated!), i.e., during the robot movements the TCP stands still and only the selected angle changes.
During cartesian movement the X-Y-Z-coordinates of the TCP are indicated in mm and the orientation angles A, B, C are indicated in degrees.

tUVW

Movement of the tool in the hand coordinate system.
Contrary to sub-operating mode „Axis 1 - 6" the TCP (Tool Center Point) moves on a straight line during cartesian movement and the orientation of the tool remains constant during the movement. The TCP can be moved in the hand coordinate system via the keys X-(1-), X+(1+), Y-(2-), Y+(2+), Z-(3-) and Z+(3+).
During cartesian movement the X-Y-Z-coordinates of the TCP are indicated in mm and the orientation angles A, B, C are indicated in degrees.

Operation in INFO - Mode Key "Info"

Second menu

Eing dig

Binary representation of the user inputs. Each input is represented by a bit indicating the current status (zero or one). Counting manner:

First line beginning at the right h.s.: Byte 0 Bit 0 to 7; Byte 1 Bit 0 to 7; Second line beginning at the right h.s.: Byte 2 Bit 0 to 7; Byte 3 Bit 0 to 7 etc.

Ausg dig

Binary representation of the user outputs. Each output is represented by a bit indicating the current status (zero or one). Counting manner:

First line beginning at the right h.s.: Byte 0 Bit 0 to 7; Byte 1 Bit 0 to 7; Second line beginning at the right h.s.: Byte 2 Bit 0 to 7; Byte 3 Bit 0 to 7 etc.

SPS Merk

Representation of the PLC markers in hexadecimal format (system variable _SPS[n]). In the first line there are the markers _SPS[1] and _SPS[2]; in the second line there are the markers _SPS[3] and _SPS[4] a.s.o.

Operation in FKT - Mode

The FKT - mode is activated with the white operating mode key "Funct".
The following operations are possible:

setting the system clock;
modification of the program protection;
interruption of archival storage and

Selection of the possible operating functions is made via the function keys under the display.

First menu

Dat/ time Programming the system clock. One after the other the following inputs are expected, the numbers below 10 can be entered with 1 digit. After each input the key "ENTER" must be operated or a comma must be entered.

day (1...31)
month (1...12)
year (98; 99; 00; 01...)
hour (0...23)
minute (0...59)
second (0...59)

The last input (second) must be finished with the key "ENTER" . The following message is displayed:

real time clock is programmed

DNC rst initialization of the DNC-interface after a transmission error

Operation in CWIN - mode key "Windows"

Operation in EDIT - Mode program selected, no further key operated

Description of the indication in the display

The EDIT-mode is activated as soon as a program (no directory) has been selected.
The display consists of 20 lines in total.
The first line indicates the set value of the override regulator. This value is indicated as number and as red beam the length of which varies with setting of the regulator.

The second line (blue) contains the number of the step where the cursor is located. Behind the step number the current directory and the name of the selected program are indicated the instructions of which are indicated in the lines 3 to 18 .

The lines 19 and 20 contain the selection menus for the function keys.

The first step of a program always contains a program type followed by the name, the last step is always named "END".

For instructions containing several parameters the display might run out of the window to the right hand side. In order to read the rest of a line, lateral displacement of the text in the window is possible with the arrow keys on the right and left hand side. Each operation of the key displaces the text by one character, if the key is kept pressed the text is continuously running into the selected direction.
If the key "SHIFT" was operated prior to pressing the arrow key, the text jumps to the right resp. left hand side by 20 characters. With key "ALT" and subsequent operation of the arrow key on the left the display jumps to the beginning of the lines. "ALT" and arrow key on the right places the text so far to the left hand side that the last character of the current line is at the right edge of the window (when the normal display consists of more than 37 characters).

For scrolling in the current program there are used the keys arrow up and arrow down. The cursor is moved line by line, if the key is kept pressed, the cursor is continuously running up or down.

The combination "SHIFT" and arrow key moves the cursor 15 lines up resp. down. The combination "ALT" arrow key upward sets the cursor on the first line (step 1), the combination "ALT" arrow key downward on the last line of the program ("END").

Direct jump on a step inside the program is possible via the key "Step" after input of a number and operation of the key "ENTER".

Operation in EDIT-Mode

Selection of the possible operation functions is ensured via the function keys under the display. The keys on the left and right hand side of the function keys are used for switch-over between the menu levels.

First menu

Step edit

Editing the current step in a program .

The cursor can be moved in the current step with the arrow key on the right resp. left hand side.

The combination "Shift -arrow key" moves the cursor by 20 characters to the right resp. left hand side.

The combination "Alt -arrow right" puts the cursor to the end of the line. The combination "Alt -arrow left" puts the cursor to the beginning of the line.

The command word and the describers cannot be edited.

Step mark

Marking the current step; various block operations can be executed with marked steps (see below)

Step del

Deletion of the current step from the program. After deletion the cursor jumps to the preceding step. The first and the last step of a program cannot be deleted.

There is no inquiry! Deletion of a step at the time being cannot be reset!

Step v/i

Validate resp. invalidate current step. An invalid step is marked by the sign "\" (backslash) before the command word. Invalid means that this step is not executed in automatic or test mode.

Second menu (block operations)

Blck imp

Insertion of a block from the "intermediate file" (The block stands in the program S:/$SYSTEM/$TEMP/$PASTE.MPR)

The block is inserted into the current program at the cursor position

Blck exp

Copy marked block of the current program into the "intermediate file". (program S:/$SYSTEM/$TEMP/$PASTE.MPR is generated resp. overwritten)

Only one block must be marked in the current program (one line or several continuous lines without blank).

Blck del

Deletion of all marked steps in the current program

There is no inquiry! Deletion of a block at the time being cannot be reset!

Blck vali

All marked steps of the current program are validated.

Blck inv

All marked steps of the current program are invalidated .

(S. step g/u)

Blck copy

Copy block in the current program. One continuous block must be marked only (one line or several lines without blank).

The block is inserted at the cursor position.

Access authorization ROBOTstarV

	Customer			REIS	
Level	1	2	3	4	5
access via	without code number	code word 1	code word 2	PIN	TAN
user	operator	setter	programmer	programmer (AEI,BK)	EE
offered menus	none	application-spec. operation macros	all application-spec. system instructions and macros	all system instructions	all open
operation	START / STOP home position with/without robot HNZE	execution of macros reprogramming editing archival storage program and step selection	programming of all user-specific system instructions programming of macros	all open programming of all system instructions programming of macros	all open

Organization of the binary inputs and outputs in system variables

by the example of the user inputs

Variable	Byte - Number			
_IBIN_IN[1]	3 Bit 76543210	2 Bit 76543210	1 Bit 76543210	0 Bit 76543210
_IBIN_IN[2]	7 Bit 76543210	6 Bit 76543210	5 Bit 76543210	4 Bit 76543210
_IBIN_IN[3]	11 Bit 76543210	10 Bit 76543210	9 Bit 76543210	8 Bit 76543210
_IBIN_IN[4]	15 Bit 76543210	14 Bit 76543210	13 Bit 76543210	12 Bit 76543210
_IBIN_IN[5]	19 Bit 76543210	18 Bit 76543210	17 Bit 76543210	16 Bit 76543210
_IBIN_IN[6]	23 Bit 76543210	22 Bit 76543210	21 Bit 76543210	20 Bit 76543210
_IBIN_IN[7]	27 Bit 76543210	26 Bit 76543210	25 Bit 76543210	26 Bit 76543210
_IBIN_IN[8]	31 Bit 76543210	30 Bit 76543210	29 Bit 76543210	28 Bit 76543210
_IBIN_IN[9]	35 Bit 76543210	34 Bit 76543210	33 Bit 76543210	32 Bit 76543210
_IBIN_IN[10]	39 Bit 76543210	38 Bit 76543210	37 Bit 76543210	36 Bit 76543210

TABLE OF CONTENTS

1.2 ADJUSTMENT OF PARAMETERS BY MEANS OF USER COMMANDS

1.2.1 The interpolation mode

Command: **BEWEG_ART** #PTP
#LINEAR
#SPLINE
#ZIRK_MIN_ORI
#ZIRK_BAHN_ORI

The type of interpolation is controlled with the command BEWEG_ART.

The movement mode #SPLINE connects programmed points by a path without corners.

The ZIRK movement modes are used for circular interpolation. The system constant #ZIRK_MIN_ORI (RSIV: ZIRK1) indicates that readjustment of orientation is made on the shortest way here. The constant #ZIRK_BAHN_ORI (RSIV: ZIRK2) activates the circular interpolation with orientation readjustment in a coordinate system moving along the circular contour.

Attention: In circular interpolation, the orientation in the auxiliary point is taken into consideration for the interpolation.

1.2.2 The path speed

Command: **BAHN_GESCHW** [mm/s]:<value>

Used with: **all interpolation modes except #PTP**

Definition of the path speed in the modes #LINEAR, #ZIRK_MIN_ORI, #ZIRK_BAHN_ORI and #SPLINE (RSIV: GESCH_CP).

1.2.3 The path acceleration

Command: **BAHN_BESCHL** [%]:<value>

Used with: **all interpolation modes except #PTP**

Definition of the path acceleration in the modes #LINEAR, #ZIRK_MIN_ORI, #ZIRK_BAHN_ORI and #SPLINE. The percentage refers to a maximum acceleration which can be accessed to via the system variable _RACC_MAX. This variable is allocated in the initialization file for the system variables (→chapter 1.5).

1.2.7 Activation of the approximation mode for PTP and CP

Command: **UEBERSCHL** #EIN
#AUS

There is no distinction between activation of CP-/PTP-approximation. The parameters for configuration of approximation are only considered with activated approximation:

The following description parameters are used:

- BAHN_RADIUS and BAHN_DISTANZ
- CP-speed in the approximation point _IV_FLYBY_S (RSIV: UEBER_CP)
- Orientation speed in the approximation point _IV_FLYBY_O
- ...

With activated approximation, **always** a path approximation is executed. Pure speed approximation **doesn't** exist any more.

The movement mode SPLINE is a particularity. Since the path already doesn't have any corners in this case, a programmed BAHN_RADIUS and/or BAHN_DISTANZ won't be considered.

1.2.8 The path approximation radius

Command: **BAHN_RADIUS** [mm]:<value>

Used with: **ÜBERSCHL #EIN in all interpolation modes except #PTP**

Definition of the "path approximation sphere" on a CP-path. (RSIV:RAD_CP). In a distance of BAHN_RADIUS before reaching a programmed position, the robot leaves the programmed path and only enters into the next path in a distance of BAHN_RADIUS after the position. A connection path is moved in-between.

1.3 ADJUSTMENT OF PARAMETERS VIA SYSTEM VARIABLES

1.3.1 Speed approximation factor of the path

Action: **KOPIERE** source:<value>,destination:**_IV_FLYBY_S**

Explication of name: **V** stands for speed, **FLYBY** for approximation parameter and **S** for the path.

Used with: **ÜBERSCHL #EIN in all interpolation modes except #PTP**

Range of values: 0[%] - 100[%]

The system variable _IV_FLYBY_S (RSIV: Command UEBER_CP) contains a percentage factor describing the passage speed in the approximation point. The approximation point is situated in the middle of the approximation path.

$$v_{flyby} = (v_1 + v_2) / 2 * \_IV\_FLYBY\_S / 100;$$

v_1 : path speed of the current path;
v_2 : path speed of the following path;
v_{flyby} : path speed in the approximation point.

1.3.2 Speed approximation factor of the orientation

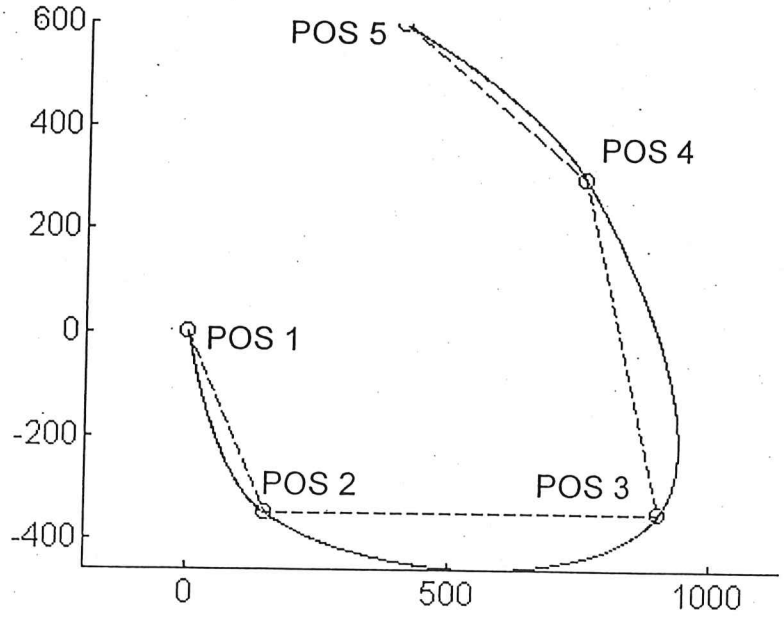
Action: **KOPIERE** source:<value>,destination:**_IV_FLYBY_O**

Explication of name: **V** stands for speed, **FLYBY** for approximation parameters and **O** for the effect to the orientation.

Used with: **ÜBERSCHL #EIN in all interpolation modes except #PTP**

Range of values: 0[%] - 100[%]

The a.m. copy action effects a speed approximation for the path orientation with approximation command being switched on. The system variable represents a percentage factor describing the orientation speed in the approximation point similar to the effect of _IV_FLYBY_S.

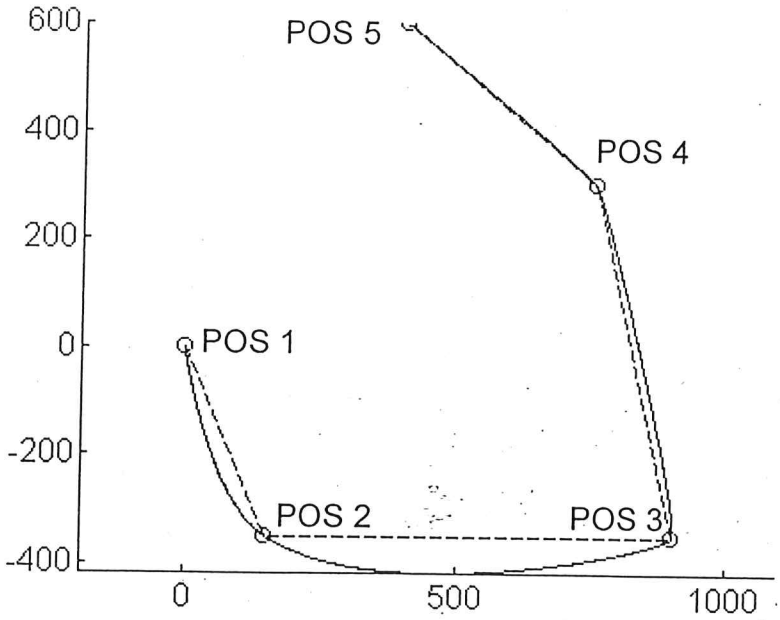


Teach Procedure:

1. POSITION 1

2. BEWEG_ART: #SPLINE

3. POSITION 2
4. POSITION 3
5. POSITION 4
6. POSITION 5



Teach Procedure:

1. POSITION 1

2. BEWEG_ART: #SPLINE

3. POSITION 2

4. _ISPL_TENS = 20
5. POSITION 3
6. POSITION 4

7. _ISPL_TENS = 100
8. POSITION 5

1.3.5 Curvature parameters of the SPLINE interpolation for the course of the orientation path

Action: **KOPIERE** source:<value>,destination:**_IORI_TENS**

Explication of name: **ORI** stands for orientation, **TENS** for tension

Used with: **BEWEG_ART #SPLINE or UEBERSCHL #EIN in the interpolation mode #LINEAR, #ZIRK_MIN_ORI and #ZIRK_BAHN_ORI**

Range of values: 0[%] - 100[%]

The factor indicates the tension of the spline for the course of the orientation. The closer the factor is to zero, the stronger the orientation path approximates the interpolation on the shortest way, the bigger consequently the discontinuity will be at the programmed position.

Since with activated approximation the orientation is interpolated in the spline concept in all CP-interpolation modes, this variable is of great importance in these cases.

1.3.6 Weighting of the orientation

Action: **KOPIERE** source:<value>,destination:**_IORI_WEIGHT**

Explication of name: **ORI** stands for orientation, **WEIGHT** for weight.

Used with: **all interpolation modes except #PTP independent from the approximation action**

Range of values: 0[%] - 100[%]

The factor indicates a measure in how far the orientation of a programmed point shall be considered for interpolation. Accordingly, 100% means that the programmed orientation is passed through.

The smaller the factor is adjusted, the less the programmed orientation will be considered at the corresponding point.

With 0% the programmed orientation won't be taken into account at all. In this case the path control calculates an average orientation by means of the neighbor orientations.

Due to this way of functioning it is **impossible** to allocate a weight _IORI_WEIGHT ≠ 100% to two successive positions. The command is inde-

1.4 ADJUSTMENT OF PARAMETERS IN THE MACHINE DATA

The constants for the "spline mathematics" are filed in the machine data. These machine data are pre-adjusted by the system and don't have to be changed by the user. For the sake of completeness they are listed in the following. The following system constants are needed:

1.4.1 RSPOS_APPROX_QUAL / RSORI_APPROX_QUAL

Significance: positive approximation quality of the reparameterization for position [in mm] and orientation [in degree].

Range of values: RS..._APPROX_QUAL > 0

Default: RSPOS_APPROX_QUAL = 1e-6
RSORI_APPROX_QUAL = 1e-5

1.4.2 RDELTA_OV_MAX

Significance: max. admissible override change per interpolation cycle.

Range of values: 0 < RDELTA_OV_MAX <= 1.0

Default: 0.1

1.4.3 RAPPR_FACT_MAX

Significance: restriction of the approximation area relative to the path length.

Range of values: 0 < RAPPR_FACT_MAX < 0.5

Default: 0.4

1.4.4 RSYNC_WEIGHT

Significance: weight of the reference phase duration for the synchronization of speed profiles:

Range of values: $0.0 \leq RSYNC\_WEIGHT \leq 1.0$

Default: 1.0

1.4.9 RS_ORI_EPS

Significance: minimum distance of programmed orientations in degrees for determination of paths without orientation way to be traveled.

Range of values: RS_ORI_EPS > 0.0

Default: 0.1

1.4.10 RS_AX_EPS

Significance: minimum distance of programmed additional axes positions in degrees for determination of path zero steps in PTP and for additional axis zero steps.

Range of values: RS_AX_EPS > 0.0

Default: 0.001

2 Guidelines for programming in SPLINE operation for processing of free form surfaces

2.1 INTRODUCTION

Besides the path control modes LINEAR, ZIRK_MIN_ORI and ZIRK_BAHN_ORI, another movement mode is at the programmer's disposal in the RSV-surface: SPLINE. This movement mode connects programmed points with a continuous path without corners. Thus, arbitrary curved path courses can be generated in the best possible manner with a minimum of programmed points.

In addition, the orientations are connected in a similar way without any discontinuity. All this happens without leaving or shortening the programmed positions and orientations.

The **SPLINE interpolation mode** is therefore ideal for the processing of free form surfaces **and should always be used for these application cases.**

Programming of such paths requires special strategies regarding the choice of programming positions. In addition, the path behavior can be adapted to the corresponding requirements by means of many parameters. The default adjustment of the path parameters creates a course of path suitable for many applications. Optimization of same, however, has to be effected by adaptation of these parameters.

The SPLINE movement mode creates at programmed points already smooth paths without discontinuity. Therefore no additional measures need to be taken regarding path approximation. Programming of the user commands **BAHN_RADIUS** and **BAHN_DISTANZ** is consequently **omitted.**

All movement parameters, their significance and default are described in chapter 1.

2.3 SETTING OF POSITIONS

As a matter of principle, programming should be started with as few positions as possible. For the choice of positions you should particularly take care of curvature changes of the contour to be processed.

Points **must** be set

- in each **curvature change,** e.g. at the outlet from a small radius to a path only slightly curved.
- in each **curvature peak,** e. g. in the middle of a small radius.
- in each orientation change

Points **should** be set

- in the **neighborhood of a curvature change,** in order to restrict the effects of the curvature change to a small area.

Since a constantly high speed has to be maintained also in small radii, it has to be observed for adjustment of the orientation that the contribution of the head axes to the robot movement is as small as possible.

The choice of positions is illustrated by characteristic two-dimensional examples. The general three-dimensional case doesn't raise any new problems, since curvature changes and peaks will occur also in this case. In each change of curvature direction a position has to be set here, too.

The conventional method of programming by means of circle segments is compared in order to see the differences. Here a circle end point must be programmed at each curvature.