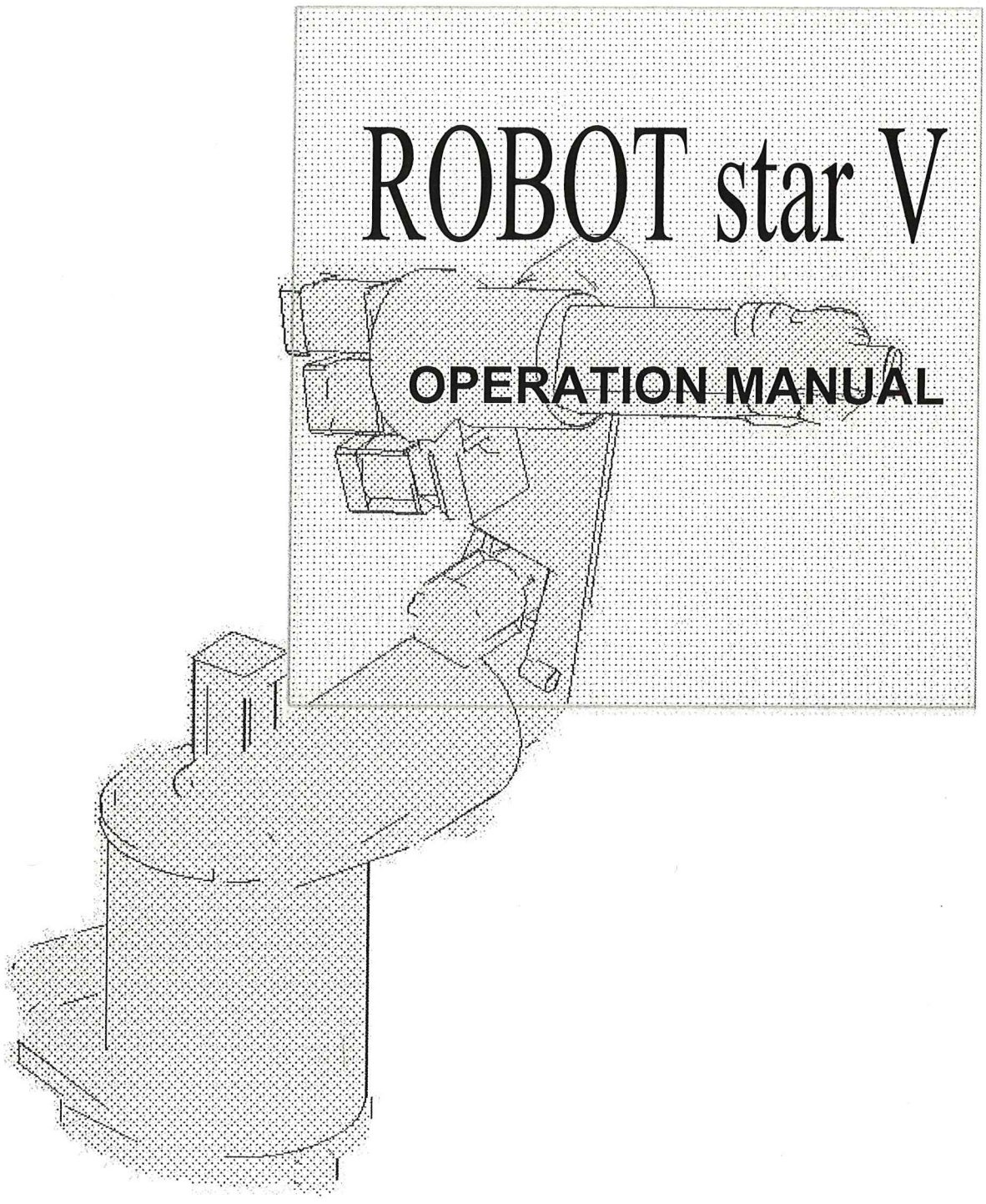
REIS ROBOTICS

ROBOT star V

OPERATION MANUAL



Version: RSV V7.x 1.2000 GB

REIS ROBOTICS

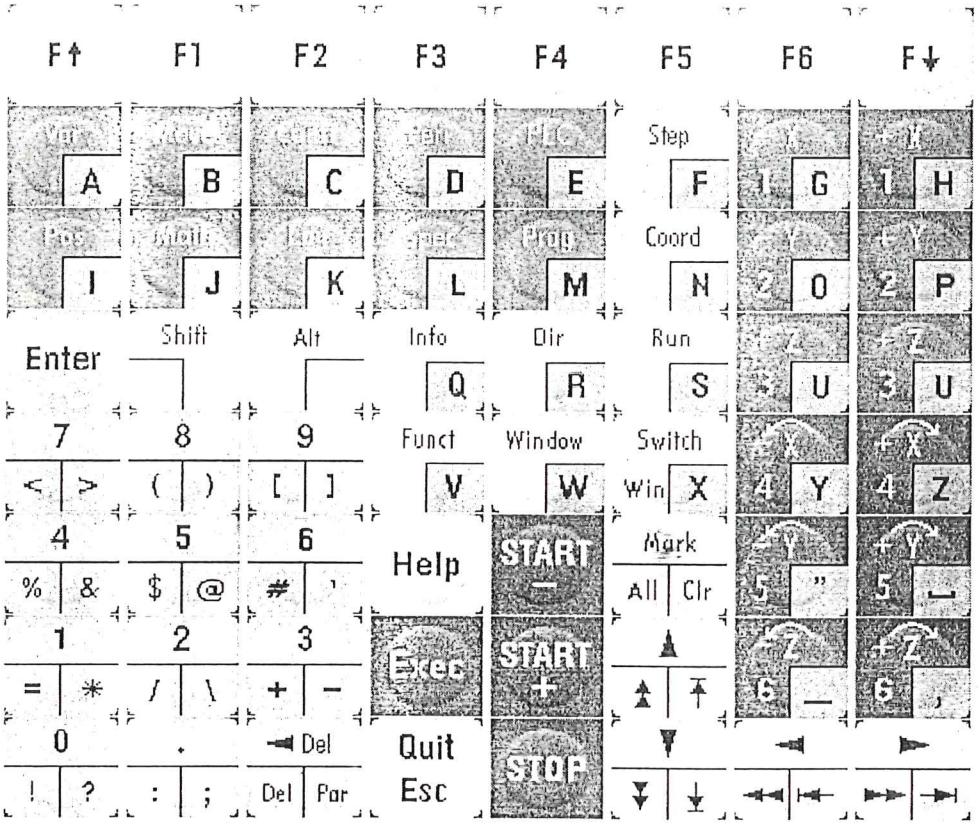
Operating manual ROBOTstar V

BDA-RSV/07.98/E

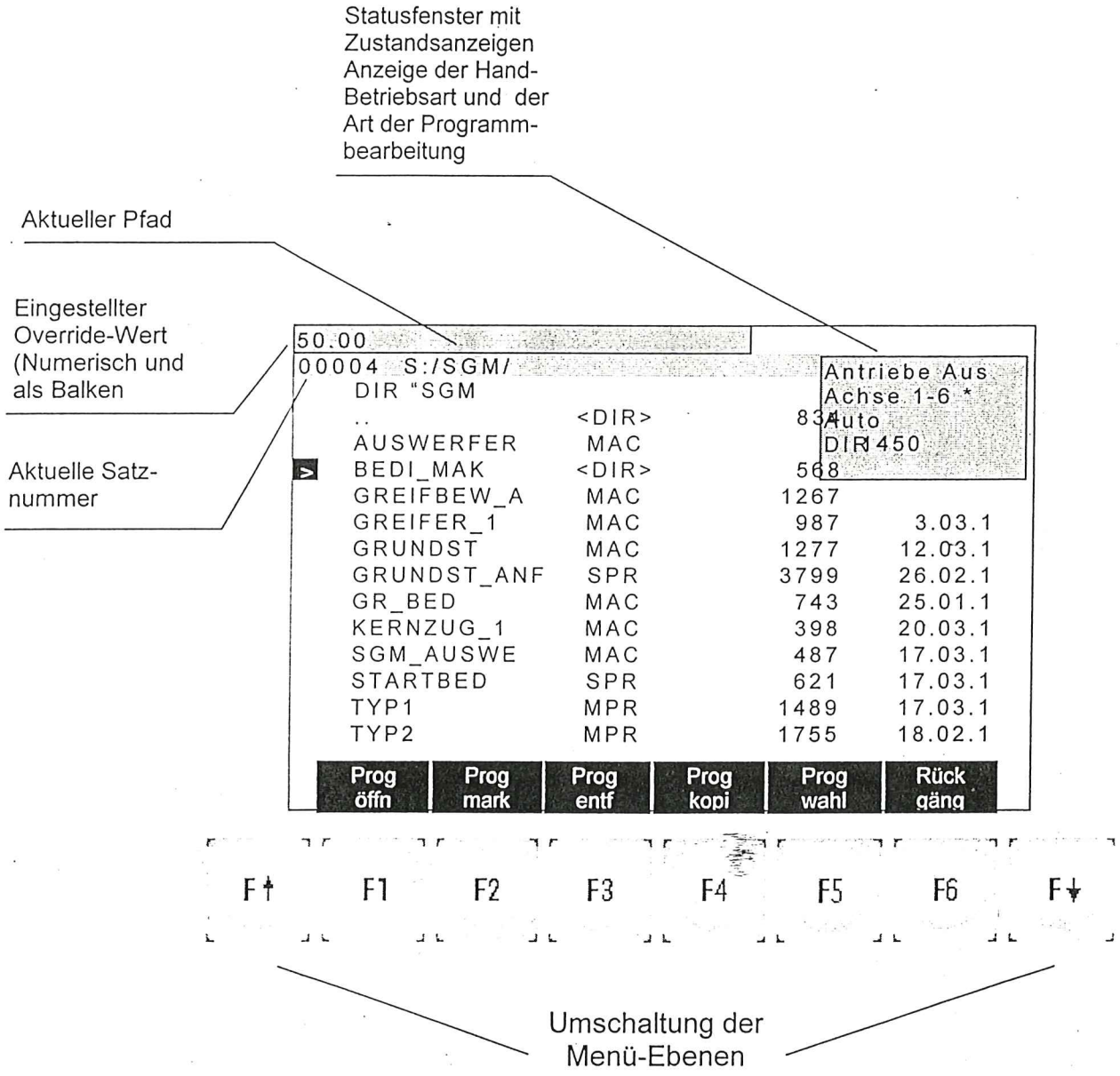
REIS ROBOTICS

1 Portable teach pendant PHG

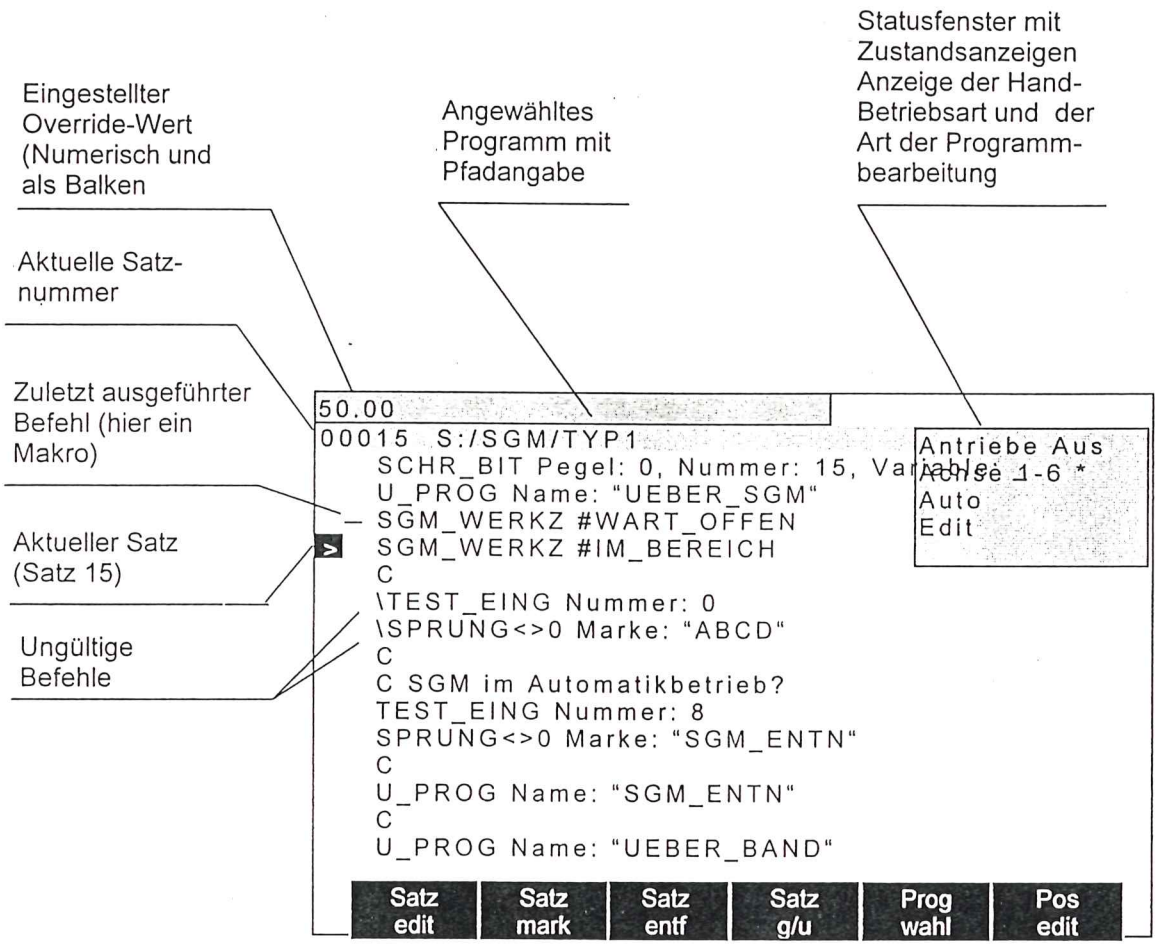
1.1 Arrangement of the keyboard on the PHG



1.2 Display of the PHG
DIR-Mode



1.3 Display of the PHG EDIT- Mode



2 Operating modes of the control ROBOTstar V

2.1 Operating modes

The last line in the status window shows the current operating mode of the control. The mode changes according to the operating state. Depending on the active mode, there will be various possibilities for operation.

The two main modes are the following ones:

2.1.1 DIR - mode

The display of the teach pendant (PHG) shows the inputs of a directory. You reach the DIR-mode by operating the key "Dir".

2.1.2 EDIT - mode

The display of the teach pendant (PHG) shows the content of a program. You reach the EDIT-mode by selecting a program.

Change to further modes, in order to select operating functions, is possible from the DIR- as well as from the EDIT-mode. After selection of the function the control returns to the original mode.

2.1.3 RUN - mode

Is activated by operating the key "Run"
The following settings are possible in RUN-mode:

- kind of program processing (automatic, test)
- start of the integrated PLC
- switching on the VKE- and status display for the integrated PLC
- approach of the reference points
- synchronization resp. calibration of the robot axes

2.1.4 COOR - mode

Is activated by operating the key "Coord". This mode determines in which manner the robot is moved manually.

2.1.5 INFO - mode

Is activated by operating the key "Info".
The following information can be called in Info-mode:

- actual states of the robot
- contents of variables and PLC markers
- suppressed error messages
- actual states of binary inputs and outputs
- status window (switching on and off)

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 Movement modes of the robot

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 Program types

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!).

4.1.5 PAL Pallet program

Contrary to other program types pallet programs have a fixed format. They must contain the following commands in the indicated order:

- four definitions for global or local integer variables
- three definitions for global or local vector variables
- three definitions for local integer constants
- one tool data step and
- four positions

The names of the variables and constants to be defined are freely selectable, observing the rules for type identification. Commentary lines can be added in the program.

From a pallet program the control calculates the following characteristics of the pallet:

- geometrical dimensions
- alignment of the pallet
- parts number in one row
- number of rows in one layer
- number of the layers
- total number of the parts
- distance of the parts towards each other
- distance of the rows towards each other
- distance of the layers towards each other

The pallet programs only serve for calculation of shifting vectors.

If a pallet shall be processed, the movement sequence for the first part of the pallet must be programmed. This movement sequence must be allocated to the corresponding pallet program. With the palletizing function, during program cycle, the entered positions are shifted by the calculated vector each. Calculation is made depending from internal counters which are modified with each access to the pallet (the four defined integer variables).

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 Programming of Movement Sequences

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

5.2.1 Move modes

5.2.2
INTERPOL (instruction set Move)

There are two totally different movement modes by means of which the TCP moves from one programmed position to the other.

PTP movements and CP movements
PTP - Point to Point; CP = Controlled Path

The movement mode PTP (point to point) is preadjusted as default value. PTP-movement is determined by the movement way of the axes only. They start at the same time and simultaneously stop in the corresponding end position. Therefore, the TCP does not follow a straight line but moves on a curved path (only with articulated arm robots).

CP means that the TCP will approach the next following position or the next following positions on a mathematically defined path. CP movements are interpolation modes "#LINEAR", "#SPLINE", "#ZIRK_MIN_ORI" and "#ZIRK_BAHN_ORI".

After selection of the command "INTERPOL" another menu is opened from which the required movement mode of the robot can be selected.

For movements on a straight line select movement mode "#LINEAR". The TCP will approach the next following position(s) on a straight line.
If orientation in the destination point is different from the start point, readjustment is made continuously during TCP movement.

The two circular interpolation modes differ due to the execution of orientation changes on the circular path. "#ZIRK_MIN_ORI" effects that orientation changes will be executed on the shortest possible way.

With "#ZIRK_BAHN_ORI" orientation change will be effected like on a straight line which has been curved in the shape of a circular path.

In order to enable the control to calculate the circular path, an auxiliary position on the circular path (circular auxiliary point) must be entered after the command "#ZIRK_MIN_ORI" or "#ZIRK_BAHN_ORI". It is approached and entered like a normal position. After the auxiliary position, the input of the end point of the circular path is effected. Several circular segments can be programmed directly one after the other.

After switch-on of the circular interpolation always an even number of positions must have been programmed before commutation of the operation mode is possible.
After "#ZIRK ... " the control interprets each first, third, fifth etc. position as circular auxiliary point, each second, fourth, sixth etc. position as end point of a circular path and simultaneously as start point of the next circular segment, until the movement mode will be switched over.

In the movement mode #SPLINE the TCP moves on a path that connects the programmed positions without any corners. (For detailed description please refer to the documentation "path control)

5.2.3 Example for circular interpolation

S1	MPR	BEWEGUNG (INTERPOL)
S2	WERKZEUG	T
S9	POSITION	A
S10	POSITION	B
S11	BEWEG_ART	#LINEAR
S12	POSITION	C
S13	BEWEG_ART	#ZIRK_BAHN_ORI
S14	POSITION	D
S15	POSITION	E
S16	POSITION	F
S17	POSITION	G
S18	BEWEG_ART	#PTP
S19	POSITION	H

If no movement mode is programmed at the beginning of a program, all following positions will be approached in the default adjustment #PTP. In the example the TCP will approach the positions A and B in movement mode PTP. Position C in step 12 will be approached on a straight line. The movement mode ZIRK_BAHN_ORI in step 13 effects that an arc of a circle will be calculated from the positions C, D, and E on which the TCP approaches position E in step 15. Since circular interpolation is still active, another arc of a circle will be calculated from the positions E, F, and G which is directly attached to the first one. Position G will be approached on this second arc of a circle. Position H and the following ones will be approached in movement mode PTP again.

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 Displacement of programmed movement sequences

6.1 RELATIVE <name of the vector variable>

Programmed positions can be displaced by a vector. The shifting values must be filed in a vector variable. A vector variable contains the following values:

Component	Description	Unit
1	Frame	
2	shifting in X-direction	[mm]
3	shifting in Y-direction	[mm]
4	shifting in Z-direction	[mm]
5	modification of the orientation angle A	[degree]
6	modification of the orientation angle B	[degree]
7	modification of the orientation angle C	[degree]

The content of the vector variables can be modified with the commands "COPY_OFFS", "VEC_ADD" and "VEC_SUB".

After selection of the command "RELATIVE" of function group "Move" the name of the vector variable has to be entered in which the shifting values are stored.
Before the relative shifting can be activated in the user program, valid values must be allocated to the components of the vector variables.
After treatment of the command "RELATIVE V...", the values contained in the vector variable V... will be added to all following positions, and the robot approaches the positions resulting from this.
The activated shifting is maintained until another vector variable will be activated or until shifting is switched off completely. For switching off the relative shift the following command must be programmed:

"RELATIVE Vector: _VNULL

_VNULL is a system variable and contains the value 0 in all six components.

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.

7
Control of the peripheral equipment

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 Organization of the binary inputs and outputs

Pick-up of the inputs and outputs is effected via two system variable arrays:

Binary inputs: _IBIN_IN[10]

Binary outputs: _IBIN_OUT[10]

Both arrays consist of 10 variables with 4 bytes each.
Each bit of this variable represents an input resp. output.

The inputs and outputs are picked up via byte- and bit numbers. I.e., addressing of 40 bytes is possible. The **bytes** are numbered from **0 to 39.**
In each byte there are available the **bits** number **0 to 7.**

The following instructions are used for direct access to inputs and outputs:

WRITE_BIT	command group Log
TEST_BIT	command group Contr
WAIT_BIT	command group Contr

A system equate has to be entered as first parameter:

#AUSGANG	access to a user output
#EINGANG	access to a user input

Addressing is made in the two following parameters:

Byte:	Byte number of the input resp. output (0 to 39)
Bit_Nr:	Bit number of the a.m. byte (O to 7)

The functions of the inputs and outputs are shown in the circuit diagram of the robot. Numbering of the inputs and outputs in the robot program and in the circuit diagrams are identical.

Example: Setting an output:

0,,39 0,,7

WRITE_BIT #AUSGANG, Pegel: 1, Byte: 5, Bit_Nr: 3

With this command that output would be set that is indicated with 5.3 in the circuit diagram and that is shown in the variable _IBIN_OUT[2] in bit 11.

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

7.2 Switching of digital outputs

As already described, switching of digital outputs is effected via the system variables _IBIN_OUT. If a bit is set (level 1) or reset (level 0) in one of the variables, then the CAN bus switches on resp. off the programmed output at the corresponding module (node). 8 or a multitude of outputs can be connected to a CAN bus module, i.e., the connections of a module correspond to one or several bytes of the system variable _IBIN_OUT. Programming therefore is made via setting resp. resetting of the corresponding bytes and their bits. This is executed with the command WRITE_BIT of the command group Log.

7.2.1 Switching on an output

The output 1.6 (acc. to circuit diagram switching on a belt) is to be set.
The following steps must be entered:

Select command "WRITE_BIT"
Select "#AUSGANG" in the opened menu
enter as level a "1"
enter byte number "1"
enter bit number "6"

Then, the command stands in the program as follows:

WRITE_BIT #AUSGANG, Pegel: **1**, Byte: **1**, Bit: **6**

7.2.2 Switching off an output

The a.m. belt shall be stopped. For programming proceed as mentioned above, only a 0 has to be entered as level. The command is as follows:

WRITE_BIT #AUSGANG, Pegel: **0**, Byte: **1**, Bit: **6**

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).

7.3.2 Conditional branch depending on the input signal

It is possible to program program branches that are only executed when level 1 resp. level 0 is applied to an input.

This is realized with the command "TEST_BIT"

After selection of the command and input of the parameter "#EINGANG the branch condition must be programmed:

#=0 the branch is executed when 0 volt are applied to the input
#=1 the branch is executed when 24 volt are applied to the input

Then, byte and bit number must be indicated, and as last parameter the label name the branch shall be executed to.

If the branch condition is not fulfilled, processing of the program is continued with the next step.

TEST_BIT #EINGANG, #=**1**, Byte: **0**, Bit_Nr: **6**, Marke: "**CDE**"
U_PROG Name: "**REINIGEN**"
MARKE "CDE"

In the above example the subprogram REINIGEN (CLEANING) is only executed when no 24 volt are applied to the input 0.6 . Otherwise the subprogram call is omitted.

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 Program calls

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.

8.3 Program call via indirect addressing

The indirect program call for RSV is made via a string variable. For definition of this variable (e.g. in program VAR) the string length must be indicated in round brackets. The name starts with S.
Example:

VAR Name: **S_PROG(20)**

With the **COPY**- instruction a string with max. 20 characters can be copied into this variable. The string must stand in inverted commas.
Example:

KOPIERE Quelle: **"BEREICH_1"**, Ziel_Var: **S_PROG**

Then, the program the name of which is contained in the variable "S_PROG" can be called with the instruction **PROGRAM** resp. **CALL.**

It is important that the variable name does **not** stand between inverted commas, because then the control would search for the program with the name S_PROG!
Examples:

PROGRAM Name: **S_PROG (no inverted commas!!)**
(here, now the main program BEREICH_1 would be called)

KOPIERE Quelle: **"3344"**, Ziel_Var: **S_PROG**
CALL Name: **S_PROG**
(here, the subprogram 3344 would be called)

KOPIERE Quelle: **"3344"**, Ziel_Var: **S_PROG**
CALL Name: **"S_PROG"**
(here, the subprogram S_PROG would be searched!)

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 Branches within a program

Conditional and unconditional (absolute) branches can be programmed within a program. For all branch commands a branch destination must be indicated from where program treatment shall be continued. It does not matter whether the branch shall be executed forward or backward. Branches from one program into another are not possible.

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 Conditional branches

Contrary to the absolute branches, with the command for a conditional branch decision is made up whether the branch is executed or whether processing of the program is continued with the next step. The condition may be the state of a single bit or the content of a byte resp. of a variable.

10.3.1 Test of a bit

In order to test a single bit and to execute a branch depending on its state (0 or 1), the command "TEST_BIT" from the group "Contr" must be selected. After selection the following must be entered:

Selection of the operand that contains the bit to be tested.

#EINGANG	test of a user input
#AUSGANG	test of a user output
#MERKER	test of a PLC - marker
#VARIABLE	test of a bit in an integer variable

Selection of the branch condition

#=0	branch, when the bit is not set
#=1	branch, when the bit is set

Input of the byte number, when the input, output or marker was indicated as operand resp. input of the variable name, when variable is programmed as operand.

Input of the number of the bit that has to be tested

Input of the branch destination

Examples:

TEST_BIT #MERKER, **#=1**, Byte: **52**, Bit_Nr: **5**, Marke: "**ABC**"

When the PLC marker 52.5 is set, a program branch to the label ABC is executed, otherwise processing of the program is continued with the next step.

TEST_BIT #VARIABLE, **#=1**, Variable: **I123**, Bit_Nr: **5**, Marke: "**ABC**"

When the bit 5 of the variable I123 is set, a program branch to the label ABC is executed, otherwise processing of the program is continued with the next step.

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.

Examples:

TESTE #AUSGANG, Op_1: **0**, **#<**, Byte: **1**, Marke: "**ABC**"

When no output is set in the output byte 1 (output 1.0 to 1.7), a branch is executed to the label ABC; when at least one output of this byte is set, processing of the program will be continued with the next step.

TESTE #VARIABLE, Op_1: **I123**, **#<**, Op_2 **10**, Marke: "**ABC**"

When the number indicated in the variable I123 is smaller than 10, a branch is executed to the label ABC , otherwise processing of the program will be continued with the next step.

11 Variables

A variable is a parameter that can be changed. There is a multitude of applications when programming, e.g. counting functions, indirect program call, simultaneous inquiry of several inputs, relative shiftings etc. are realized with variables.

11.1 Variable types

Variables may contain different parameters. The variable type determines the kind of variable content. Identification of the variable type is given with the first letter of the variable name:

I	Integer variable; contains integer number (32 bit)
R	Real variable; contains real number with digits after the comma (64 bit)
V	Vector variable; contains shifting vector
P	Position variable; contains position in space
T	Tool variable; contains tool data
S	String variable; contains a character string

11.1.1 Integer variables

An integer variable contains a binary number with 32 bits. The representable numbers are in the range from -2 147 483 648 to +2 147 483 647.
The bit with the highest value (bit 31) is the sign bit, i.e., as long as this bit is 0, a positive number stands in the variable, if this bit becomes 1, there is a negative number.

If all bits, except bit 31, are set, then the number +2 147 483 647 stands in the variable. If only bit 31 is set and all others are 0, then the variable content corresponds to the decimal number -2 147 483 648. If all 32 bits are set, the variable contents is -1.

11.1.2 Real variables

A real variable consists of 64 bits. The representable numbers are in the range from -1.2 multiplied by 10 to the 38^{th} to +1.2 multiplied by 10 to the 38^{th} . The bit with the highest value (bit 63) also in this case is the sign bit. The remaining digits according to the IEEE format (say EI TRIPPEL I) are converted into a decimal number with digits after the comma.

11.1.3 Vector variable

This variable consists of 7 components. The first component (integer) contains the frame-number (coordinate system). The next three components (real) contain the shifting values in X-, Y- and Z-direction (unit mm). The components 5-7 (real) contain differential values for the orientation angles A, B and C (unit degree).

11.1.4 Position variable

This variable consists of 36 components in total composed of the following: Position type, frame, number of main and additional axes, coordinates of the TCP, tool data and positions of the axes (unit increments).

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.

11.3 Description of variables

The definition of a variable is only a reservation, i.e., the variable does not yet contain any valid values. First, it must be described (initialized) before it can be used.

11.3.1 Description of complete variables

The command "COPY" from the command group "Var" copies a source operand into a variable

e.g. **KOPIERE** Quelle: **100**, Ziel_Var: **I123**

After execution of this command the variable I123 contains the numeric value 10.

KOPIERE Quelle: **T**, Ziel_Var: **T1**

This command effects that the 16 components of the tool variable T are copied into the tool variable T1.

11.3.2 Description of individual components

If individual components of tool-, vector- or position variables are to be described, then the command "COYP_OFFS" from the command group "Var" must be programmed. This command also allows to copy individual components from tool vector or position variables.

5 parameters must be entered after selection of the command COPY_OFFS:

number of the component that shall be copied (source index)
name of the source variable (source)
number of the component of the destination variable (destination index)
name of the destination variable (destination)
number of components that shall be copied (number)
(including named component)

Examples:

COPY_OFFS Quellindex: **2,** Quelle: **V1,** Zielindex: **2,** Ziel: **V2,** Anzahl: **3**

The second, third and fourth component of the vector variable V1 are copied into the variable V2 .

COPY_OFFS Quellindex: **1,** Quelle: **100.0,** Zielindex: **12,** Ziel: **T1,** Anzahl: **1**

Hand length 100 mm is copied into the tool variable T1.

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)

13 Processing of vector variables

VEC_ADD
VEC_SUB
VEC_VALUE

The commands "VEC_ADD" resp. "VEC_SUB" effect that the individual components of two vector variables will be calculated with one another (by addition resp. by subtraction). The result of the executed operation is in the vector variable being named as second operand.

Example:

The vector variables VEC1 and VEC2 contain the following values:
(angles A,B and C are equal in both variables)

	VEC1	VEC2
displacement in X-direction	150	200
displacement in Y-direction	-30	100
displacement in Z-direction	10	30

After the sequence of commands

```
VEC_ADD     VEC1,VEC2
RELATIVE    VEC2
```

the following positions will be approached displaced in X-direction by 350 mm, in Y-direction by 70 mm and in Z-direction by 40 mm.

With the command "VEC_VALUE" the absolute length of a vector can be calculated and entered in a real variable.

Example:

The vector variable "VNAHT_1" contains the following components:

displacement in X-direction	80 mm
displacement in Y-direction	60 mm
displacement in Z-direction	0 mm

After execution of the command

```
VEC_VALUE    VNAHT_1,RLAENGE
```

the variable RLAENGE contains the value 100.0.

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
```

14 Logic Operations

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.

14.2 OR-linkage

OR <integer value, variable name>

The 32 bits of a variable are linked with the 32 bits of the 1st operand via the OR-function. For each bit of the variable the following is valid:
With an OR-operation with 1 the result is always a 1, independent of the bit status.
With an OR-operation with zero the value of the bit is taken over into the result. If a zero is in this bit, the result is a zero; if a 1 is in this bit, the result is a 1.

The command "OR" is selected in the 1st menu of instruction set "Log" with function key F2. After input of the 1st operand (the number resulting from the 32 individual bits) and a comma, the variable has to be named that shall be linked with the 1st operand.

14.3 EXCL_OR-linkage

EXCL_OR <integer value, variable name>

The 32 bits of a variable are linked with the 32 bits of the 1st operand via the EXCLUSIVE-OR-function. For the individual bits of the variable the following is valid:

With an EXCLUSIVE-OR-operation with 1 the result is zero when the value of the bit is a 1. The result becomes 1 when the value of the bit is a zero.

With an EXCLUSIVE-OR-operation with zero the result becomes 1 when there is a 1 in the bit; the result becomes zero when the value of the bit is a zero.

Briefly summarized - the result is 1 when both bits being linked via EXCLUSIVE-OR have various states; it is zero when both bits have the same value (both bits 0 or both bits 1).

14.4 Inversion of variable contents

INVERT <variable name>

The 32 bits of an integer variable are inverted, i. e., bits the content of which is zero become 1 and bits the content of which is 1 become zero. It must be observed that the highest-value bit (2**31) serves as sign bit. As long as this bit is zero, the variable content is interpreted as positive number, when it is 1, the content is interpreted as negative number. "INVERT" must not be mixed up with the command "NEG" from instruction set "Math".

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.

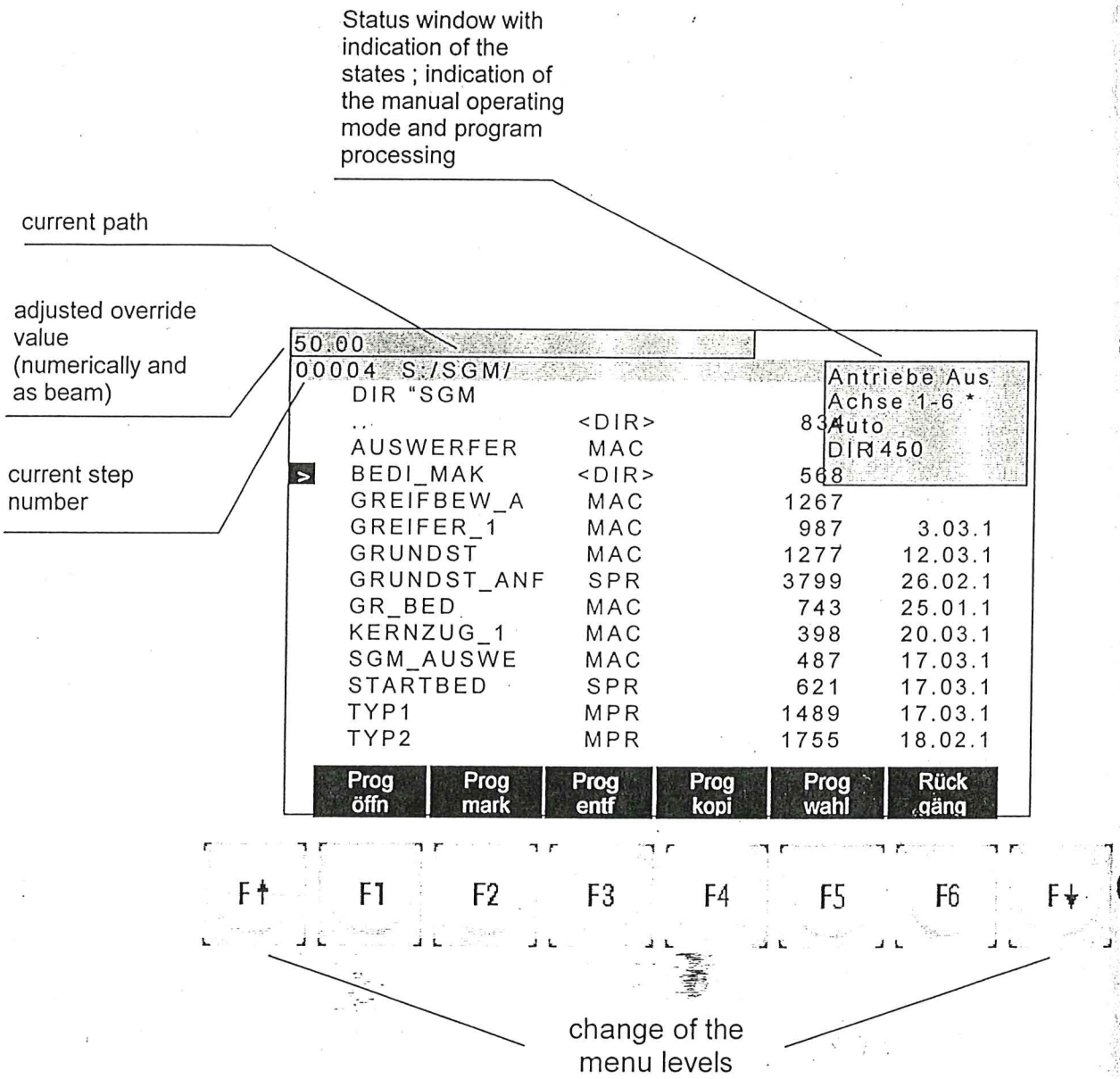
SHIFT_R <number of digits, variable name>

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.

Operation in DIR - Mode Key "Dir"

Display of the PHG DIR-Mode



Description of the indication in the display

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 drive and the directory are indicated the inputs of which are indicated in the lines 3 to 18 . S: is the standard drive of the control (S-RAM), A: means the disk drive.

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

A directory is a program type. The first step of a directory is named DIR followed by the name, the last step is always named END. The directories form a tree structure. Based on the master directory DIR "/" generation of further sub-directories is possible. The names are freely selectable and may have a length of up to 20 characters. The maximum allowed nesting depth is 6.
The second step of a sub-directory is always named ".." <DIR>. It represents the superimposed directory; i.e. this entry is not available in the master directory.

Programs and directories are listed in alphabetical order in a directory (numbers before letters). The program type is indicated after the name:

MPR	main program
SPR	sub-program
MAC	macro
PAL	pallet program
PLC	PLC - program
<DIR>	sub-directory

The properties of the programs are indicated after the program types:

program length [Byte]
date of generation or last modification
time of generation or last modification
activated program protection (see below)
remark that global definitions exist (see below)

In order to read the rest of a line running out of the window to the right hand side, 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 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".

Selection of program or directory

Selection of another directory or a program is possible from the directory by setting the cursor on the desired entry. The change is executed by operation of the key "ENTER" . If the cursor stands on step 2 of a sub-directory (".." <DIR>), then change is made into the superimposed directory. If a directory is selected, then the DIR - mode is kept. If jump into a program is made, the mode switches over to "EDIT".

For movement of the cursor in the current directory 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 directory ("END").

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

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**

Prog copy

Copying of programs. Directories cannot be copied!
If no marking exists, the following message is given:

Acknowledge the source program:

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

new program name (optionally with
drive and path):

The new program name must be entered. For entry without indication of path the new program is copied into the current directory. After operation of the key ENTER the type of the new program must be selected. Type DIR is not admitted!

If programs are marked, the following message is given:

destination path without program name:

The directory (with complete indication of path) must be entered, into which the marked programs are to be copied.
If the programs are to be stored **on disk,** then only **A:** must be entered! (At the time being it is impossible to copy into subdirectories on the disk!)

Prog sel

Selection of program by input of the program or directory name

If the program to be selected is not in the current directory, the path must be indicated. Directories are indicated by "/" (slash).

If only the slash is entered, direct change to the master directory is made.

Undo

No function yet at the time being.

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!

Prog ren

Renaming programs and directories

If no marking exists, the following message is given:

Acknowledge the source program:

In the input line below the path and name of the program / directory on which the cursor stands is indicated. By input of another name (perhaps indication of path) a random other program / directory can be renamed. After operation of the key ENTER the following message is given:

new program name (optionally with
disk and path):

The new program resp. directory name must be entered. For input without indication of path the new program / directory is displaced into the current directory. After operation of the key ENTER the type of the new program must be selected.

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

destination path without program name:

Enter the directory (with complete indication of path) into which the marked programs / directories shall be displaced.

Third menu

Flsh read Not for general use!

Flsh frmt Not for general use!

Operation in RUN - Mode Key "Run"

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.

First menu

Test 1

Single step mode macro calls are treated like a system command. (program execution by the interpreter)

Input signals must be simulated. This applies for the commands: WAIT_BIT #EINGANG (INPUT)... and TEST_BIT #EINGANG (INPUT)....
Binary user outputs are not switched
Approximation functions are not active

Test 2

Sequential step mode macro calls are treated like a system command. (program execution by the interpreter)

User inputs and user outputs are treated like in the automatic mode
Approximation functions are active

Test 3

Single step mode macro calls are treated like a system command. (program execution by the interpreter)

User inputs and user outputs are treated like in the automatic mode
Approximation functions are active

Test 4

Sequential step mode macro calls are treated like a system command. (program execution by the interpreter)

Input signals must be simulated. This applies for the commands: WAIT_BIT #EINGANG (INPUT)... and TEST_BIT #EINGANG (INPUT)....
Binary user outputs are not switched
Approximation functions are not active

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.

Second menu

Cal/ Sync

When the robot became asynchronous (corresponding message in the status window), the menu item "SYNCHRONIZATION" must be selected after selection of this option. The window then faded in shows the asynchronous axes in binary code. This display can be edited; axes that shall be synchronized, are represented with a 1, the other axes with a 0. After having pressed the key ENTER the following message is displayed

synchronization finished

After synchronization the reference points of the synchronized axes must be approached via the function Ref Pos (F3) , and the reference markings must be checked!

The menu item "Cal/Sync" (setting) is used for determination of the resolver offset values. This operation must be made with the first start-up, after replacement of the motor, after replacement of the resolver and after replacement of the gear unit.

Start PLC

Start of the integrated PLC

When a PLC program was written, or when modifications were made in an existing program, then the internal PLC must be started anew. That PLC program is started the name of which is copied in the subprogram

S:/$SYSTEM/$SYSMAC/SYVARINI

into the system variable "_SSTARTPLC". When a new program is to be started, the COPY-command in the program "SYVARINI" must have been executed before!

This program is automatically started with switching on the control.

Ref Pos

Approaching the reference points.

All axes are simultaneously approaching their reference points on the shortest way. The traversing speed is constant, independent from the adjusted override (condition higher than zero). When the last axis has reached its reference position, the following message will be indicated:

reference position reached

Attention! Danger of collision!

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 Key "Coord"

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.

tXYZ mouse	Cartesian movement of the TCP with the adapted 6D-mouse. The TCP is not moved via the keyboard of the teach pendant (PHG), but via the connected 6D-mouse.
rABC mouse	Orientation change via the 6D-mouse
AXIS 7 - 12	When the robot is equipped with up to 6 additional axes, those can be moved into negative resp. positive direction via the movement keys "1-", "1+" to "6-", "6+" . The key pair "1" applies for axis 7, the key pair "2" for axis 8 etc. If the keys of several axes are simultaneously operated, the corresponding axes will also move at the same time.

Second menu

ACTI CAL1	activation of the mouse-calibration at the first adaptation point
ACTI CAL2	activation of the mouse-calibration at the second adaptation point
ACTI KPHG	activation of the mouse-calibration at the teach pendant (PHG)
RUN CAL1	calibrating the mouse at the first adaptation point
RUN CAL2	calibrating the mouse at the second adaptation point
RUN KPHG	calibrating the mouse at the teach pendant (PHG)

Operation in INFO - Mode Key "Info"

Operation in INFO-Mode

The INFO- Mode is activated with the white operating mode key "Info".
In INFO - Mode it is possible to call information about the robot, about variable contents and about the state of user inputs, user outputs and PLC markers .

An opened menu window can be closed with "Quit/Esc", or by selecting another menu item.

Selection of the possible operating functions 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.

First menu

Achs Istw — Display of the axes' positions of the robot. For rotation axes indication is given in degrees, for linear axes in millimeters.

Kart Istw — Display of the cartesian actual values of the TCP. The current coordinates X, Y and Z are displayed in millimeters and the orientation angles A,B and C in degrees.
The displayed data are calculated from the axes positions and the tool data activated last.

Inkr Istw — Indication of the axes' positions of the robot in increments. (resolution of the resolvers: 4096 increments per motor revolution)

Vari able — Indication of contents of global variables. It is also possible to indicate contents of local variables being defined in the current program (only from the EDIT - Mode).
After selection of the menu item the name of the variable to be checked must be entered. For integer variables the display format can be selected (decimal; hexadecimal; binary).

Err Enab — Error messages being constantly given by the system and thus blocking the display and operation can be suppressed. Suppression is canceled when the drives are switched on.

Stat Ums — Switching off or on the status window

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 key "Funct"

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

Prog prot

Modifying the protection of marked programs.
After selection of this function first the input of a code number is expected. Then, the various protection functions can be activated one after the other. With entering the number 1 the indicated operation is locked, by entering the digit 0 the indicated operation is released. The following operation possibilities can be locked:

loesch.	deletion of programs
ausf.	execution of programs
edit.	editing of program steps
strukt.	modification of the program structure
umben.	renaming of programs

After the last input the following message is displayed:

program protection modified.

If a wrong code number has been entered, or if no programs have been marked, the corresponding message will be given at the end of the input procedure.

Order rst

Abort of an archival storage. The current transmission of a file is finished. All files not yet transmitted are canceled in the order list and archival storage is finished.

Operation in CWIN - mode key "Windows"

Operation in CWIN-mode

The CWIN- mode is activated by the white operating mode key "Window".

Horizontal or vertical division of the display in two windows is possible. After executed division and operation of the key "Quit/Esc" the cursor can be moved from one window to the other with the key "Switch Win" . While a program is running in automatic mode in one window, another program for instance can be edited in the second window.

First menu

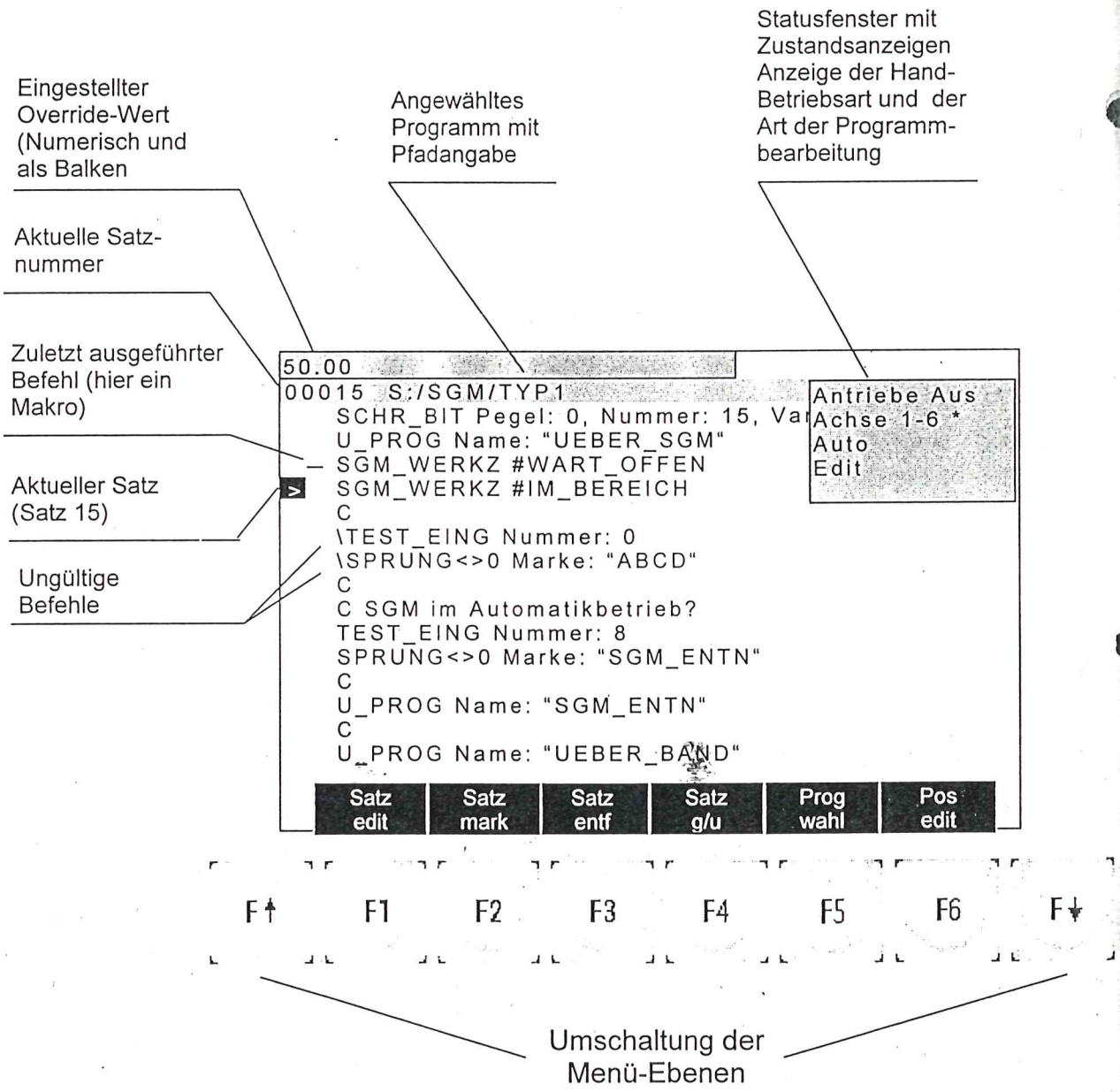
TIng Hztl Horizontal division (divi hori) of the display. For further operation the key "Quit/Esc" must be operated.

TIng Vrtk Vertical division (divi vert) of the display. For further operation the key "Quit/Esc" must be operated.

TIng aufh Cancel existing division (divi undo). For further operation the key "Quit/Esc" must be operated.

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

Indication in the display of the PHG EDIT-Mode



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".

Invalid step

An invalid step is marked with the sign "\" (backslash) before the command word. Invalid means that this instruction is not executed in automatic and test mode.

Instruction executed last

The sign "-" stands on the step that was completely executed as last instruction in automatic or test mode before the program was stopped.

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.

Prog sel

Program selection by entering the name of the program or the directory

When the program to be selected is not located in the current directory, the path must be indicated. Directories are indicated by "/" (slash).

If only the slash is entered, direct change into the master directory is executed.

Pos edit

Overwriting of a programmed position by the actual position of the robot.

The position is overwritten immediately and without inquiry!!

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.

Third menu

Find repl Search of a character string in the current program. Character strings can be replaced by other ones, when they are not part of command words, describers or system equates.

Code word Input of the code word for definition of the operator surface. Selection of 5 levels with various authorization for access can be selected. See table "authorization for access ROBOTstarV":

Edit mem Not for general use! Only fur customer service!

Undo Not yet implemented at the time being.

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

ROBOTstarV
Organization of the binary inputs and outputs

The communication between hardware and software takes place via two system variable arrays:

binary inputs: **_IBIN_IN[10]**

binary outputs: **_IBIN_OUT[10]**

Both arrays consist of 10 variables with 4 Bytes each.
Every Bit of these variables represents an input or an output respectively.

The inputs and outputs are addressed via Byte- and Bit-numbers. Consequently 40 Bytes can be addressed. The **Bytes** are numbered from **0 to 39.**
The **Bits** number **0 to 7** are available in each Byte.

The following commands serve for direct access to the inputs and outputs:

SCHR_BIT	group of commands Log
TESTE_BIT	group of commands Contr
WARTE_BIT	group of commands Contr

The first parameter to be entered is a system equate:

#AUSGANG	access to a user output
#EINGANG	access to a user input

Addressing is made in two of the following parameters:

Byte:	Byte number of the input or output (0 to 39)
Bit_Nr:	Bit number of the a. m. Byte (0 to 7)

Example: Setting an output:

SCHR_BIT #AUSGANG, level: 1, Byte: 5, Bit_Nr: 3

With this command the output would be set which is represented in the variable _IBIN_OUT[2] in Bit 11.

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

DOCUMENTATION

REIS GMBH & CO MASCHINENFABRIK OBERNBURG

GENERAL

Title: Path Control

Author:	Wetzel
Date:	01.07.98
Control version:	RSV
Software revision:	01_00

Date: 19.06.98

Release: Som

TABLE OF CONTENTS

1 Parameters of the path control

1.1 INTRODUCTION

The path control of the ROBOTstarV allows configuration of traversing ways with many parameters. The central traversing parameters such as movement mode, path speed or activation of the approximation mode are adjusted by means of user commands. Other parameters being used less frequently are programmable in system variables.

As a matter of principle, all **operations on system variables and the user commands are self-locking,** thus being valid for all following programmed points or paths. **Exceptions** are just the command **BAHN_ZEIT** (path time) and programming of the orientation weight in the system variable **_ORI_WEIGHT**.

In an initialization file all **path parameters are pre-allocated with values. Working is already possible with this pre-allocation.**
Finally, for recognition of limit cases, e.g. paths which don't have to move any distance, but only a readjustment of orientation, and for the spline mathematics there are further parameters which are stored in the machine data and which must only be changed under instruction.

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).

1.2.4 The PTP speed

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

Used with: **interpolation modes #PTP**

Definition of the speed in #PTP mode. The percentage refers to the max. axes speeds of the robot. (machine data RAXES_NOM_SPEED).

1.2.5 The PTP acceleration

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

Used with: **interpolation modes #PTP**

Definition of the acceleration in #PTP mode. The percentage refers to the max. axis acceleration of the robot. (machine data RAXES_NOM_ACCEL)

1.2.6 The traversing time on a CP-path

Command: **BAHN_ZEIT** [s]:<value>

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

Definition of the traversing time along a CP-path (modes: #LINEAR, #ZIRK_MIN_ORI, #ZIRK_BAHN_ORI and #SPLINE).

This command is not self-locking and has priority towards a programmed path speed. It **can** be used instead of the path speed command BAHN_GESCHW and it **must** be used if a path has been programmed without distance, but with orientation readjustment; because in this case a path speed is without importance.

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.

1.2.9 The path approximation distance

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

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

This parameter defines the distance to the programmed corner point of a CP-path. A **distance programming has priority** towards the radius programming.

The command can be switched off by BAHN_DISTANZ = 0.0, which also corresponds to the pre-allocation (→ chapter 1.5). Then an approximation path is calculated by means of BAHN_RADIUS.

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.

1.3.3 Speed factor of the circular auxiliary point orientation

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

Explication of name: **V** stands for speed, **CIRCHP** for circular auxiliary point and **O** for orientation.

Used with: **BEWEG_ART #ZIRK_MIN_ORI or #ZIRK_BAHN_ORI.**

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

The a. m. copy action effects a speed approximation for the circular auxiliary point orientation. The system variable represents a percentage factor defining the orientation speed in the auxiliary point completely analog to IV_FLYBY_O.

1.3.4 Curvature parameters of the spline interpolation for the course of the position path

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

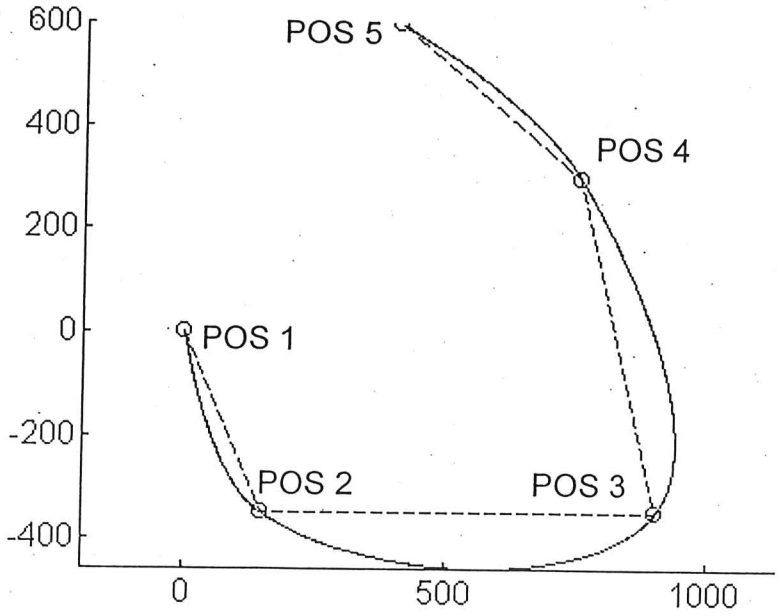
Explication of name: **SPL** stands for spline, **TENS** for tension

Used with: **BEWEG_ART #SPLINE.**

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

The variable is only used with programmed #SPLINE interpolation mode. With this factor the course of the path between two programmed points can be changed.

The name is explained by the comparison of the path with the curvature of an elastic material. The lower the tension is at a programmed position, the stronger the path approximates the linear path, the bigger will thus be the discontinuity at the position. The higher the tension is, the more the path strikes out, so that the discontinuity at the position becomes smaller (→ example plots).

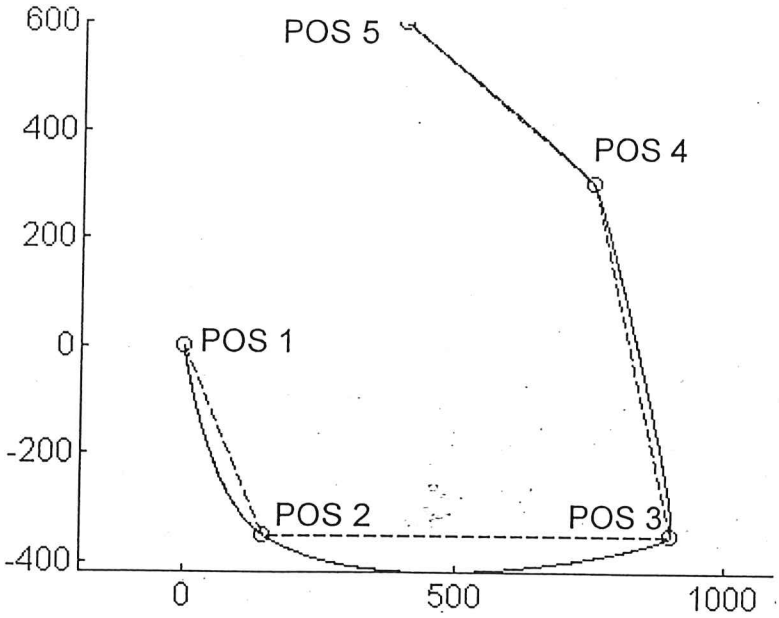


Teach Procedure:

1. POSITION 1

2. BEWEG_ART: #SPLINE

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



Teach Procedure:

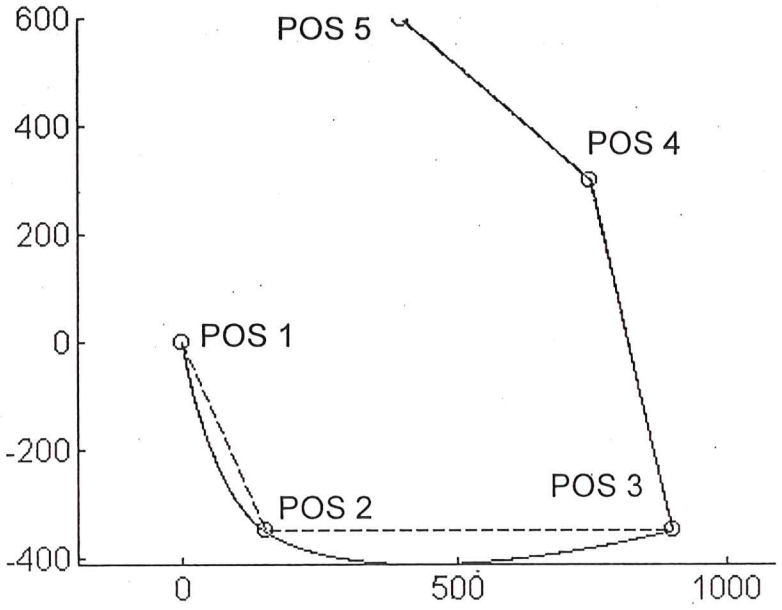
1. POSITION 1

2. BEWEG_ART: #SPLINE

3. POSITION 2

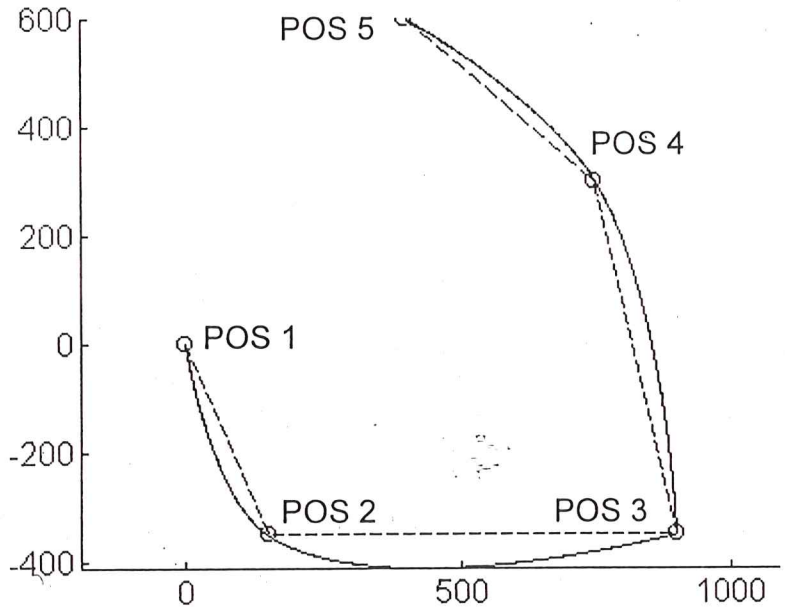
4. _ISPL_TENS = 20
5. POSITION 3
6. POSITION 4

7. _ISPL_TENS = 100
8. POSITION 5



Teach Procedure:

1. POSITION 1
2. BEWEG_ART: #SPLINE
3. POSITION 2
4. _ISPL_TENS = 0
5. POSITION 3
6. POSITION 4
7. _ISPL_TENS = 100
8. POSITION 5



Teach Procedure:

1. POSITION 1
2. BEWEG_ART: #SPLINE
3. POSITION 2
4. _ISPL_TENS = 0
5. POSITION 3
6. _ISPL_TENS = 100
7. POSITION 4
8. POSITION 5