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-

pendent from approximation actions and is used for shortening an orientation distance. It is **not self-locking**.

1.3.7 Weighting of the orientation of the circular auxiliary point

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

Explication of name: **ORI** stands for orientation, **CIRCHP** for circular auxiliary point.

Used with: **interpolation modes #ZIRK_MIN_ORI and #ZIRK_BAHN_ORI**

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

The factor indicates a measure in how far the orientation of a programmed circular auxiliary point shall be considered for interpolation. Accordingly, 100% means that the programmed orientation is passed through. A programming of 0% leaves the orientation of the auxiliary point disregarded for the circular interpolations.

Like with _IORI_WEIGHT, an average orientation is calculated by means of start point and end point of the circle.

Like most of the parameters, this one is also self-locking for all following circular auxiliary points.

1.3.8 The brake time for programming of a BAHN_ZEIT (path time)

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

Explication of name: **ACC** stands for acceleration, **TIME** for a period of time.

Used with: **paths without distance, paths with programmed BAHN_ZEIT (path time)**

Range of values: 0[%] - 50[%]

The percentage factor defines the brake time on a CP-path and refers to the programmed BAHN_ZEIT (path time).

For example, _IACC_TIME = 20% means that 20% of the BAHN_ZEIT (path time) each are used for acceleration and deceleration respectively, and the remaining 60% for the movement with constant speed. With maximum adjustment _IACC_TIME = 50% there is no constant movement phase (triangular profile).

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

1.4.5 RSMOOTH_WEIGHT

Significance: dimension for the smoothness for reduction of the acceleration for synchronization of speed profiles.

Range of values: $0.0 \leq$ RSMOOTH_WEIGHT ≤ 1.0

Default: 0.5

1.4.6 RSYNC_MIN_PHASE

Significance: minimum phase duration for synchronization of speed profiles proportionate to the total duration.

Range of values: $0.0 \leq$ RSYNC_MIN_PHASE ≤ 0.5

Default: 0.1

1.4.7 RTIME_EPS

Significance: minimum path duration for the detection of path zero steps in sec.

Range of values: RTIME_EPS > 0.0

Default: 0.001

1.4.8 RS_POS_EPS

Significance: minimum distance of programmed points in mm for determination of paths without travel distance.

Range of values: RS_POS_EPS > 0.0

Default: 0.01

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

1.5 DEFAULT OF THE PARAMETERS

For all path parameters, a pre-allocation of the path parameters is made when the system is started, and that in the initialization file of the system variables S:\$SYSTEM\$SYSMAK\SYVARINI.MPR. In the first instance, working is possible with this default.

Allocation of the system variables to the path parameters is made as follows:

default system variable	*appropriate path parameter*	*value*
_IDEF_MOVMOD	command: BEWEG_ART	0 → #PTP
_IDEF_FLYFLG	command: ÜBERSCHL	0 → #AUS
_RDEF_VEL_CP	command: BAHN_GESCHW	10.0
_IDEF_ACC_CP	command: BAHN_BESCHL	50
_IDEF_VELPTP	command: PTP_GESCHW	20
_IDEF_ACCPTP	command: PTP_BESCHL	50
_RDEF_PATH_R	command: BAHN_RADIUS	1.0
_RDEF_PATH_D	command: BAHN_DISTANZ	0.0
_RDEF_PATH_T	command: BAHN_ZEIT	0.0
_IDEF_VFLY_S	variable: _IV_FLYBY_S	100
_IDEF_VFLY_O	variable: _IV_FLYBY_O	0
_IDEF_VCRC_O	variable: _IV_CIRCHP_O	100
_IDEF_SPLTEN	variable: _ISPL_TENS	100
_IDEF_ORITEN	variable: _IORI_TENS	100
_IDEF_ORIWGT	variable: _IORI_WEIGHT	100
_IDEF_ORICRC	variable: _IORI_CIRCHP	100
_IDEF_ACCTIM	variable: _IACC_TIME	25

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.

2.2 PROCEDURE FOR THE SPLINE PROGRAMMING

In the movement mode SPLINE there are many possibilities for modification of the course of the path. From this results a special strategy for generation of a robot program.

Based on the experience with programming of arbitrary curved screens which have to be processed with constant speed, the following guideline has proven to be promising:

1. **Extremely exact setting of positions, especially of orientations.**

2. **Switching on the approximation command.**

3. **Processing the movement in operating mode TEST1 or TEST4** for verification of the position choice, because the robot stops at each position here. Doing so, the path between the programmed positions is of no importance.
   → Adaptation of positions.

4. **Processing the movement in operating mode TEST1 or TEST4** for verification of the course of the path and the course of the orientation between the programmed positions.
   → Adaptation of the path parameters _ISPL_TENS,
   _IORI_TENS,
   _IORI_WEIGHT.

5. **Processing the movement in operating mode TEST2, TEST3 or AUTO** at low speed (via override) for verification of the path and orientation speed in the programmed points.
   → Adaptation of the path parameters _IV_FLYBY_S,
   _IV_FLYBY_O
   and perhaps the machine data RSYNC_WEIGHT.

6. Perhaps increase of the **filter variable _FILTER** for smoothing of nominal values for the axes.

In any case, the **filter** is the **last resort** for smoothing of movements!

The total of programming must be done with filter switched off or with the smallest filter, because otherwise the optimization of parameters is nearly impossible.

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

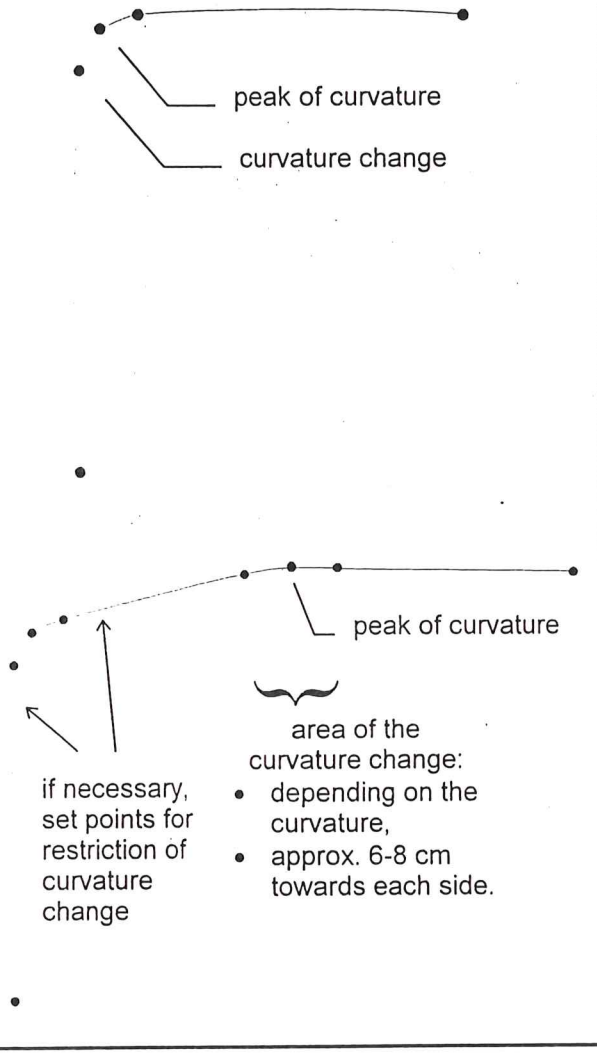
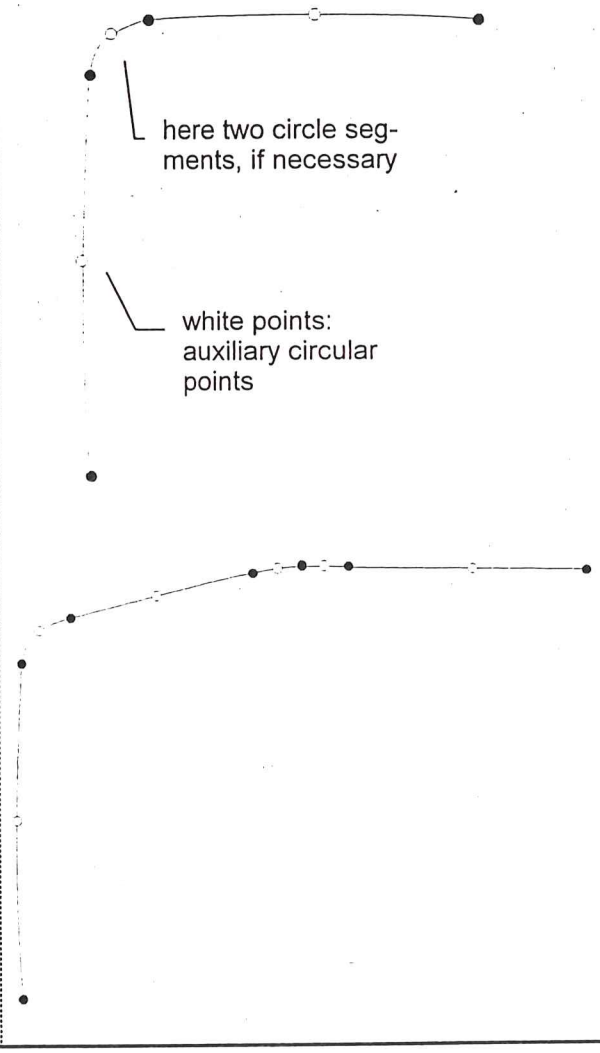
Points **should** be set

- 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.

SPLINE interpolation mode	ZIRK interpolation mode
	

2.4 OPTIMIZATION OF THE PATH PARAMETERS

In the movement mode SPLINE the **course for path and orientation** can be modified separately. **The speed profile for the course of path and orientation** can also be **modified separately**.

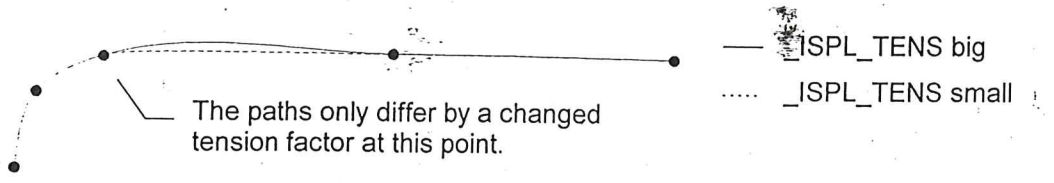
The adjustment of the user commands BAHN_GESCHW and BAHN_BESCHL is immediately clear and won't be mentioned any more in the following.

The **default of the path parameters** generates with **a modification** a course of path which is suitable for many applications: For simulation of the RSIV path behavior the system variable _IV_FLYBY_O was initialized with 0%, for processing of free form surfaces, however, stopping of the orientation in the programmed points is not desirable.

Therefore, either the initialization of _IV_FLYBY_O must be changed by the system variable _IDEF_VFLYO in the system file S:\$SYSTEM\$SYSMAK\SYVARINI or **_IV_FLYBY_O** has to be set **to 100%** at the beginning of the user program.

The following table gives possibilities for solution of some problems.

Problem	Solution
The path **strikes out too far** before/after a position before it swivels into the direction of the previous/following point.	The tension factor for the way is too high, the path must be torn nearer to the straight connection: → Reduce **_ISPL_TENS** before the positions(perhaps to 50%), then reset to 100%.



The **robot head strikes out far** between the positions. The orientation path thus moves too far between the positions.	The tension factor for the orientation is too high. → Reduce **_IORI_TENS** before the positions (perhaps to 20% or even 0%), then increase it again. Perhaps a global decrease of _IORI_TENS will be necessary, in this case, however, not to 0%.
The robot approaches orientations which are not really interesting: This problem occurs most frequently in case of **points set tight**. This may reveal itself by a **striking out of the robot head** or by great differences in the orientation speed of two adjacent paths.	Switching off orientations: → **_IORI_WEIGHT = 0** The command is not self-locking, it therefore only has an effect on the following position. Besides this, **two directly adjacent orientations must not be switched off**.
In TEST2, TEST3 or AUTO movement the robot **stops at each position, although approximation** is activated.	The robot shall pass through the programmed points without speed reduction: → **_IV_FLYBY_S = 100.** This corresponds to the default.
In TEST2, TEST3 or AUTO movement the **orientation** stops at each point, **although approximation** is active. In case of points set tight to each other this will result in jerks at those points.	→ Set **_IV_FLYBY_O = 100**.

Despite the a. m. modification, the robot **jerks** in TEST2, TEST3 or AUTO movement at positions where paths with small and big orientation ways follow one after the other.	The jerks result from **too high orientation accelerations.** Deceleration resp. acceleration of the orientation takes place within extremely short time. The length of this phase in the orientation path corresponds to the one of the path speed profile, because the machine data RSYNC_WEIGHT is set to 100%. → Reduce **RSYNC_WEIGHT** to 50 or (if the profiles may be completely independent from each other) to 0.

DOCUMENTATION

REIS GmbH & Co Obernburg

GENERAL

Title: Integrated PLC

Authors : May, Elter

Date : 19.11.98

Control version : RSV

Software revision : 02_01

RELEASE:

Date: 20.11.98

Signature: *Som*

TABLE OF CONTENTS

1 General Description of Capacity

1.1 HARDWARE CONDITIONS

The integrated PLC is running on a CPU board together with the controller software. The calculating capacity of the CPU board is divided between the controller software and the integrated PLC. A preponderance for PLC and controller software can be allocated by presetting a power stage for the PLC.

The power stage defines the number of PLC-commands per IPO-cycle. The PLC-interpreter therefore only processes the defined number of commands per cycle.
Default: 20
Machine data: IPLC_LOAD

A too high value may lead to a system error 1052.

1.2 CAPACITY

- The PLC programs are handled like robot programs, i.e. they are entered, tested and started via the teach pendant.

- Total of commands like Siemens-PLC S5/100 U, but:

  - Without the possibility to transfer parameters to subprograms
  - Different mnemonics, but same functionality
  - No data modules present (not necessary, because there are sufficient markers)
  - The total extent of languages is permitted in subprograms
  - Max. nesting depth for subprograms: 16
  - There are no start-up modules
    (for S5: organization modules OB20 .. OB22)
    Realization is possible via the initialization marker and a corresponding main program structure.
  - No distinction between PB (program module) and FB (function module).
  - Integrated PLC has functionality like FB.
  - Functions "AND" and "OR" have same priority (with S5 "OR" has higher priority).
  - Parentheses have to be set accordingly.
  - Commands for multiplication and division are included in the basic command set
  - No networks programmable (partially can be solved via subprograms)
  - Includes XOR-function

- Cycle time:

The capacity of the PLC depends on the configuration of the robot control, e.g. number of additional axes, CPU calculating capacity, position regulating cycle, duty of the control software (LIN, CIRC, PTP operation) and of the selected PLC power stage.

- The PLC cycle time may be monitored by a watchdog.
  The watchdog time is adjusted in the system variable _ITIME_PLC.

- Due to the increased cycle time, the integrated PLC shouldn't be used for time-critical controls, like for instance path switch-off.
- It has to be observed that the reaction time to state changes at the inputs or in the markers corresponds to the double cycle time.

- The PLC shares the user inputs and outputs with the robot controller.

- In the standard configuration PLC and robot controller can access to the outputs at the same time, the PLC having the higher priority.
- However, the outputs may also be allocated individually, either only to the robot control or only to the PLC.
- The number of inputs/outputs is defined by the number of connected I/O-modules, but max. 40 modules with 8 inputs each and 40 modules with 8 outputs each.
- The inputs/outputs can be addressed individually (bit-addressing)
  I/O 0.1 - I/O 39.7
- The inputs/outputs can be addressed in arrays (Byte-addressing)
  I/O 0 - I/O 39
- There are 2048 marker bytes with 8 bits each. The marker arrays 0 to 511 of those are freely configurable.
- The marker arrays 512 to 2047 (system markers) are reserved for the exchange of system states between robot controller and PLC.
- Access to the marker arrays possible via Byte address or double word address.

Information in the system markers:

- Axes actual values A1 .. A24 for range monitoring of axes
- Status of the system inputs and system outputs (PLC has only reading access)
- POWER-ON marker bit (initialization marker)
- PLC-error bit (division by zero, multiplication overflow)
- etc. (see marker list)

– The marker arrays can be read and written both by the robot controller and the PLC. They serve for quick and direct communication between the robot controller and the PLC.

– The markers can be addressed individually (M 0.0 - M 2047.7) (bit-addressing)
  - or as 8-bit word (M 0 - M 2047) (Byte-addressing)
  - or as 32-bit word (M 0 - M 2044) (double word addressing)

– The markers are maintained after switch-off of the control.

– The timer programming differs from S5; the function, however, is identical.
  Example timer 10 sec:
  S5: L KT 100.1 (unit coded in the 2nd parameter)
  SE T1

  Reis: L 1000 (unit 0.01 sec)
  ST E,T1

– The timers are situated in an own data range

– 32 timers are programmable (32 bit-word per timer)

– The time basis for timers is 1/100 sec.

– After each PLC command, a comment can be annexed for better program documentation.

– There are two 32-bit accumulators switched in series.

– After switch-on of the controller the PLC is started automatically. The start program is freely selectable via the system variable _SSTARTPLC and the corresponding path via the system variable _SPLC_PATH.

– Naming of the programs is free (up to 128 characters)

– During program start, the PLC program is checked with regard to plausibility for the recognition of

- Recursive subprogram calls
- Exceeding of the nesting depth
- Incorrect parentheses
- Uncovered subprogram calls
- Uncovered branches

– Faulty programs won't be started.

– In case of inactive PLC-commands the additional information regarding status, VKE, accu1 and accu2 is represented in red color on the teach pendant display.

2 Operating Surface/Programming Unit

2.1 PROGRAM GENERATION

PLC programs can be handled like robot programs. For distinction they have the program identification 'PLC'. PLC programs can be selected, copied and renamed like robot programs.

After generation of a new PLC program, the step for beginning of the program and the end step are displayed:

```
PLC 'program name'
END
```

Attention: **When deleting PLC programs, just the source code of the program is deleted; if the deleted program is being treated in the PLC cycle at the time being, it won't be influenced by deletion!**

2.1.1 Entering commands

Program generation is supported by a menu system. The commands relevant for the PLC are filed in 7 menus with max. 6 commands which can be achieved by means of the PLC key.

Procedure:

Press the key PLC in a HAND sub-operating mode. The first menu with the PLC commands is displayed. The other menus are reached by pressing the cursor key(s).

The command is selected with the function key allocated to the menu item or with the Enter key. Then the parameters are entered with the alphanumeric keyboard of the portable teach pendant.

During the parameter input the syntax check is active. As long as a red bar is displayed in the input line, the syntax is not okay and the command cannot be programmed.

After pressing the key <ENTER> and provided that the input was correct with regard to the syntax, the command is inserted into the actual program after the cursor.

2.1.2 Edition of commands

Commands can be edited like with robot programs.

2.1.3 Deletion of commands

Commands can be deleted like with robot programs.

Since the PLC-interpreter is working with a copy of the PLC-programs, there are all the possibilities available to change the PLC-programs during the runtime. The changes, however, will only become effective after hand-over to the PLC-interpreter by "PLC start".

2.2 START PROGRAM

In order to start a PLC-program or to activate changes, the key RUN must be pressed and the second submenu has to be scrolled to. A menu is displayed where the command "Start PLC" will be selected. If no error occurs during compilation, the changes will be taken over into the cyclic treatment of the PLC.

Remarks on the directory structure in the RSV:
The RSV offers the possibility to store PLC-programs in an own subdirectory or e. g. to keep several versions of one application in own subdirectories.
The following applies as a matter of principle:
Search path + program name are formed of the contents of the system variables _SPLC_PATH and _SSTARTPLC. _SPLC_PATH must and _SSTARTPLC may contain a path indication.
The system variable _SPLC_PATH is predefined with "S:/PLC" and _SSTARTPLC is with blanks.

Examples: _SPLC_PATH	_SSTARTPLC	started PLC-program
S:/PLC	MAIN	S:/PLC/MAIN
S:/PLC	PROJEKT1/MAIN	S:/PLC/PROJEKT1/MAIN
S:/PLC	PROJEKT2/MAIN	S:/PLC/PROJEKT2/MAIN

A directory change is not possible with subprogram calls (command SU).

2.3 START-UP BEHAVIOR

When switching on the robot controller it is checked whether the name of a PLC-program is entered in the system variable _SSTARTPLC. If so, this program will be searched in the directory from _SPLC_PATH (see above) and if existing, it will be taken over into the cyclic program treatment. Practically this program should always exist in the system, all PLC_modules are called from this start program by corresponding subprogram commands.

_SPLC_PATH should contain an absolute path, because the control selects the main directory after switch-on and the PLC-programs are only started from the current directory if the path indication is missing.

When the control is powered-up the bit 4 in the system marker 1020 is set. With this bit it is now possible e.g. to execute initialization routines. Subsequently the bit must be reset via PLC-command (e. g. R M1020.4)

2.4 ADJUSTMENT OF THE MONITORING TIME

The monitoring time is defined in the system variable _ITIME_PLC. In order to adapt it, the corresponding step in the program SYVARINI must be changed.

The unit of the monitoring time is [1/100 sec.].

2.5 TEST PROGRAM

For testing the PLC programs, it is possible to fade in behind the PLC-command on the teach pendant display the operand status, the linkage result VKE and the contents of the two accumulators. For this purpose the source text of a just running PLC-program must be on the display and the test indication must be switched on. For switching on and off the TEST indication serves the menu item Test PLC from the second submenu under the PHG key Run.

marker M1021.14 reset:	TEST-indication is switched off
marker M1021.14 set:	TEST-indication is switched on

For active commands the additional information regarding status, VKE and accumulator are displayed in black color, for inactive commands in red.

Source text and just running PLC-program must coincide. The control deactivates the TEST-display automatically if there is no coincidence (e. g. after editing). Only after the coincidence has been established again by PLC start, the control will activate the TEST-indication again.

3 Command Set

3.1 OPERANDS

The operands are either bit operands or Byte operands (8 bits) or double word operands (32 bits).

The following operands are admissible for the Reis-PLC:

- inputs (can be addressed individually):	I 0.0 - I 39.7
- input Byte:	I 0 - I 39
- input double word:	I 0 - I 36
- outputs (can be addressed individually):	O 0.0 - O 39.7
- output Byte:	O 0 - O 39
- output double word:	O 0 - O 36

The number of max. admissible inputs and outputs of course also depends on the extension stage of the robot, i. e. on the number of connected CAN-I/O-modules:

- markers (can be addressed individually):	M 0.0 - M 2047.7
- marker Byte:	M 0 - M 2047
- marker double word:	M 0 - M 2044

Timers and counters are situated in a common data range. This data range consists of 32 long words (32 bits per long word). Each long word may be used either as timer or as counter. This requires a certain discipline in programming in order to avoid double assignments.

- timers	T 0 - T 31
- counters	Z 0 - Z 31

When switching off the robot, all marker, timer and counter contents are maintained.

Constants are entered either decimal or hexadecimal. Doing so, it has to be observed that an 'H' is annexed to hexadecimal numbers.

Example: correct: 12ACH or BA12CD33H
wrong: BA12CD33

3.2 LOGIC OPERATIONS

3.2.1 AND-operation/ANDNOT-operation

The following operations link the indicated parameter with the VKE.
The result is stored in the VKE.

command	significance		
U Ax.y	AND output	x=array no.	y=output no.
U Ex.y	AND input	x=array no.	y=input no.
U Mx.y	AND marker	x=marker no.	y=bit no.
U Tx	AND timer	x=marker no.	
U Zx	AND counter	x=marker no.	
UN Ax.y	ANDNOT output	x=array no.	y=output no.
UN Ex.y	ANDNOT input	x=array no.	y=input no.
UN Mx.y	ANDNOT marker	x=marker no.	y=bit no.
UN Tx	ANDNOT timer	x=marker no.	
UN Zx	ANDNOT counter	x=marker no.	
U ACCU	The content of ACCU1 and ACCU2 is linked with AND. (32 bit-operation). The result is stored in ACCU1.		

In order to summarize operations with AND, there is the command **"U ("**.
A closing parenthesis ')' belongs to this command.

3.2.2 OR-operation / ORNOT-operation

The following operations link the indicated parameter with the VKE. The result is stored in the VKE.

command	significance		
O Ax.y	OR output	x=array no.	y=output no.
O Ex.y	OR input	x=array no.	y=input no.
O Mx.y	OR marker	x=marker no.	y=bit no.
O Tx	OR timer	x=marker no.	
O Zx	OR counter	x=marker no.	
ON Ax.y	ORNOT output	x=array no.	y=output no.
ON Ex.y	ORNOT input	x=array no.	y=input no.
ON Mx.y	ORNOT marker	x=marker no.	y=bit no.
ON Tx	ORNOT timer	x=marker no.	
ON Zx	ORNOT counter	x=marker no.	
O ACCU	The content of ACCU1 and ACCU2 is linked with OR. (32 bit-operation) The result is stored in ACCU1.		

In order to summarize operations with OR, there is the command **"O("**.
A closing parenthesis ')' belongs to this command.

Attention: **The OR and the AND operation have the same priority. Therefore, parentheses have to be set in any case. There is no regulation that AND has priority over OR like for other PLC types!**

3.2.3 EXCLUSIVE-OR-operation / EXCLUSIVE-ORNOT-operation

The following operations link the indicated parameter with the VKE.
The result is stored in the VKE.

command	significance		
XO Ax.y	XOR output	x=array no.	y=output no.
XO Ex.y	XOR input	x=array no.	y=input no.
XO Mx.y	XOR marker	x=marker no.	y=bit no.
XO Tx	XOR timer	x=marker no.	
XO Zx	XOR counter	x=marker no.	
XON Ax.y	XORNOT output	x=array no.	y=output no.
XON Ex.y	XORNOT input	x=array no.	y=input no.
XON Mx.y	XORNOT marker	x=marker no.	y=bit no.
XON Tx	XORNOT timer	x=marker no.	
XON Zx	XORNOT counter	x=marker no.	
XO ACCU	The content of ACCU1 and ACCU2 is linked with EXCLUSIVE-OR. The result is stored in ACCU1.		(32 bit-operation)

In order to summarize operations with EXCLUSIVE-OR, there is the command "XO(".
A closing parenthesis ')' belongs to this command.

Due to implementation of parentheses, there is no more limitation with regard to parentheses nesting depth.

3.3 SETTING COMMANDS AND RESETTING COMMANDS

The following commands set the corresponding operand to "1", if the VKE is set.

command	significance		
S Ax.y	set output	x=array no.	y=output no.
S Ex.y	set input	x=array no.	y=input no.
S Mx.y	set marker	x=marker no.	y=bit no.
S Zx	Sets counter no. x to its initial value.		

The following commands set the corresponding operand to "0", if the VKE is set.

command	significance		
R Ax.y	reset output	x=array no.	y=output no.
R Ex.y	reset input	x=array no.	y=input no.
R Mx.y	rest marker	x=marker no.	y=bit no.
R Zx	Sets counter no. x to 0		
R Tx	Resets timer no. x.		

The following commands allocate the content of the VKE to the operand.

command	significance		
= Ax.y	output = VKE	x = array no.	y = output no.
= Ex.y	input = VKE	x = array no.	y = input no.
= Mx.y	marker = VKE	x = marker no.	y = bit no.

3.4 LOADING AND TRANSFER COMMANDS

The following commands load the operand into the ACCU1, the old content of ACCU1 is shifted into ACCU2. ACCU1 is loaded with a Byte and extended to 32 bits without sign.

command	significance	
L Ax	ACCU1 = output Byte	No.x
L Ex	ACCU1 = input Byte	No.x
L Mx	ACCU1 = marker Byte	No.x

ACCU1 is loaded with a double word

command	significance	
LD Ax	ACCU1 = output Byte	No.x to x+3
LD Ex	ACCU1 = input Byte	No.x to x+3
LD Mx	ACCU1 = marker Byte	No.x to x+3

The following commands copy the content of ACCU1 (bit 0 to bit 7) into the respective operand.

command	significance	
T Ax	output Byte	No.x = ACCU1
T Ex	input Byte	No.x = Accu1
T Mx	marker Byte	No.x = Accu1

The following commands copy the content of ACCU1 (bit 0 to bit 31) into the respective operand.

command	significance	
TD Ax	output Byte	No. x to x+3 = ACCU1
TD Ex	input Byte	No. x to x+3 = ACCU1
TD Mx	marker Byte	No. x to x+3 = ACCU1

3.5 TIMERS

The following timer types are implemented:

- pulse : I
- extended pulse : V
- with switch-on delay : E
- with switch-off delay : A

The timers are started with the start-timer command:

- ST <timertyp> , <timer>

e. g.: ST I , T 5
ST A , T 31

The runtime of the timer has to be loaded into ACCU1 beforehand; the indicated number has the unit [1/100 sec.] as a matter of principle.

Functioning of the timers corresponds to the definitions of STEP 5.

3.6 COUNTERS

Also the counters are working according to STEP 5.

The commands

ZV <counter> , 1
ZR <counter> , 1

serve for counting up and down the counter.

As additional function, also the accumulator can be counted up and down with variable increment resp. decrement:

ZV ACCU , <incr>
ZR ACCU , <decr>

<incr> and <decr> have to be in the range from 0 to 255.

3.7 COMPARISON COMMANDS

The comparison commands compare ACCU2 with ACCU1 and set the VKE to 1 if the comparison is true. Otherwise the VKE is set to 0.

- equal : =?
- not equal : ><
- greater or equal : >=
- greater : >
- less or equal : <=
- less : <

3.8 ACCUMULATOR COMMANDS

- **Shifting of the accumulator**
  With the commands

  SH R , <no.>
  SH L , <no.>

  the bits in ACCU1 are shifted by <no.> digits to the right or to the left. The digits becoming free in doing so will be filled up with zeros. The parameter <no.> has to be in the range between 0 and 31.

- **Rotation of the accumulator**
  With the commands

  RO R , <no.>
  RO L , <no.>

  the bits in ACCU1 are rotated by <no.> digits to the right or to the left. In this case, <no.> also has to be in the range between 0 and 31.

- **Addition of the accumulators**
  With the command "+" the content of ACCU2 is added to the content of ACCU1; the result is in ACCU1.

- **Subtraction of the accumulators**
  With the command "-" the content of ACCU1 is subtracted from the content of ACCU2; the result is in ACCU1.

- **Multiplication of the accumulators**
  With the command "MU" the content of Accu2 is multiplied by the content of Accu1. The result is in Accu1. If there is an overflow, the marker 1020.5 is set.

- **Division of the accumulators**
  With the command "DI" the content of ACCU2 is divided by the content of ACCU1. The result is stored in ACCU1, the rest of the division is stored in ACCU2.

  In case of overflow or division by zero the marker 1020.5 is set.

- **Formation of complement**
  With the commands

  K E
  K Z

  the complement of one or the complement of two is formed in ACCU1. The result is in ACCU1.

- **Exchange of the accumulator contents**
  For exchange of the contents of ACCU1 and ACCU2 there is the command "TA".

- **Conversion HEX <-> BCD**
  Conversion from HEX to BCD : HD
  Conversion from BCD to HEX : DH

  The conversion takes place for Accu1 respectively.

3.9 BRANCH COMMANDS

In order to execute absolute or VKE-dependent branches in a program, the commands "SP" and "M" are provided as follows:

- SP <destination> : absolute branch to the label <destination>
- SP <destination> , 1 : branch to the label <destination>with VKE = 1
- SP <destination> , 0 : branch to the label <destination> with VKE = 0
- M <destination> : agreement of label <destination>

The names of the destination labels are freely selectable, max. 128 characters are allowed.

3.10 SUBPROGRAM TECHNIQUE

PLC programs can be nested in one another up to a depth of max. 16. Each PLC program may be called by any other PLC program and vice versa. There is no restriction of the extent of language. In order to call a PLC program as subprogram, there are the commands

- SU <prog> : absolute branch into the program <prog>
- SU <prog> , 1 : branch into the program <prog> with VKE = 1
- SU <prog> , 0 : branch into the program <prog> with VKE = 0

At the end of the called program, the calling program is returned to.

All subprograms called with SU must be in the same directory, a directory change is not possible with SU.

3.11 COMMENTS

Comments may be entered in two manners:

- as own command in a line, introduced by ";" or
- after all commands except "SP", "SU" and "M", separated from the command by ";".

4 Communication with the robot control

4.1 SYSTEM VARIABLE _SPS[512]

For the robot programs the system variable _SPS[512] has been defined as an array with 512 elements. This array is identical with the marker bytes 0 to 2047 of the PLC-programs. Thus, it is possible to access to the PLC-markers from the robot program with the commands KOPIERE, SCHR_BIT, WARTE_BIT, TESTE and TESTE_BIT.

Example: KOPIERE xxx,_SPS[54] writes the value xxx into the PLC-marker bytes 212 to 215.

Attention: **In robot programs the array index is counted from 1 to max. In the PLC program counting starts with 0.**
In robot programs all integer variables have a length of 4 bytes (double word). This also applies for array elements and thus, for the system variable _SPS[...].
Therefore _SPS[54] corresponds to the marker bytes M212 to M215.
This conversion is omitted if the commands SCHR_BIT, TESTE_BIT, WARTE_BIT, SUCH_BIT and TESTE are used with the parameter #MERKER. In this case the marker Byte is indicated directly.
Example: SCHR_BIT #MERKER, Pegel:1, Byte-Nr:0, Bit-Nr:5

4.2 SYSTEM MARKERS OF THE PLC

The marker bytes 512 to 2047 are used as system markers to which also the robot controller accesses. These markers must not be arbitrarily used for own programs!

The significance of the individual marker bytes is explained in chapter 7. Summary of the marker allocation.

5 Treatment of Errors

5.1 EDITING ERRORS

Errors occurring during editing are marked by a red bar on the teach pendant display. Such errors are caused e. g. by exceeding the value ranges (Byte number or bit number too high/too low, wrong timer number etc.) or by wrong operand types or by incomplete parameter input.

5.2 TRANSLATION ERRORS

This group of errors occurs after having started the PLC; the running program isn't impaired by this. After elimination of the cause of the error the command for starting the PLC must be executed again.

Messages:

- **"Start program doesn't exist"**
  The system variable _SSTARTPLC doesn't contain a name or the entered program doesn't exist or the entered name is wrong. The system variable _SPLC_PATH contains a wrong path.

- **"Program memory is full"**
  The program memory is so full that compilation can't take place any more.

  Measure: Remove programs from the memory which are no longer needed.

- **"No PLC-command"**
  A robot command has been programmed in the PLC-program, therefore remove this command.

- **"Nesting depth too big"**
  The max. nesting depth for subprograms has been exceeded or there are recursive calls.

- **"Subprogram not found"**
  A subprogram addressed with "SU" doesn't exist under the addressed name.

- **"Incorrect parentheses"**
  Parentheses are missing or parentheses are logically incorrect.

- **"Branch destination not defined"**
  The corresponding M-command to a branch destination in a SP-command cannot be found.

- **"Multiple definition of label"**
  A used branch label is not clear, i. e. it has been used at least twice.

5.3 RUNTIME ERRORS

Runtime errors always cause the PLC to go into the stop status, the drives to be switched off and the outputs to be reset.

All runtime errors have the form: PLC error 700x.

The numbers have the following significance:

- **7000: Time error**
  Time monitoring has responded. Either an endless loop has been programmed or the adjusted monitoring time is too short for the size of the program.

- **7001: Ram overflow**
  After compilation it is tried to reserve protected memory for the PLC. If this isn't successful, the PLC has to be stopped.
  Measure: Remove programs from the memory which are no longer needed.

- **7003: Incorrect function code**
  The PLC interpreter finds a function code (command), which is not defined.
  Measure: Start program anew.

- **7004: Incorrect start identification**
  The next command to be executed by the PLC is destroyed.
  Measure: Start program anew.

5.4 USER-DEFINED CLEAR TEXT MESSAGES

The program PLCMSG_CNF in the directory S:/PLC serves for the output of messages and errors in clear text. This program must contain the texts as well as the bit-coded reaction of the RSV-controller. The following reactions are possible:

Bit number	Reaction
0 - 3 = 0	Message output, the robot is not stopped
0 - 3 = 3	Error message, the robot is stopped
0 - 3 = 7	Error message with switch-off of the drives
8 = 0	This message is always displayed
8 = 1	This message is only displayed with drives being switched on

If bit 0 - 3 = 0 the message is displayed in a green window (operator guidance). The activity of the robot thus isn't influenced. If bit 0 - 3 <> 0 the message is displayed in a red window (error message) and the program treatment will be stopped with or without switch-off of the drives.

The texts may contain max. four parameters and may consist of max. 256 characters per text line. The first parameter is taken from the marker double word M952, the second one from M956, the third one from M960 and the fourth parameter is taken from M964.

Possible control characters: \n for return/linefeed
\d for display of a parameter as decimal number.
\h for display of a parameter as hexadecimal number.
\b for display of a parameter as binary number.

Format of a text line: Mnemonic text number reaction "text with max. 256 characters"

The program PLCMSG_CNF is already prepared in such a way that only the texts and the reaction of the control need to be entered.

The communication between integrated PLC and RSV control for output of freely defined clear text messages is running over the marker double word M948. Via the highest value Bit #31 in this marker a distinction is made whether a freely defined clear text message (Bit #31 set) or a standard message PLC error <parameter> (Bit #31 reset) shall be displayed.

For output of a clear text message the Bits 0 ... 30 must contain the text number of the corresponding I-step from the program PLCMSG_CNF and Bit 31 must be set. For output of the standard message the Bits 0 ... 30 must contain the parameter and Bit 31 must be reset.

Since the PLC markers may also be described by the robot program, it is now possible to initiate error messages also from the user program.

Attention:
At each PLC cycle end (END step in the PLC main program reached) the PLC interpreter checks the content of the marker double word M948 and triggers the message output, if the content of M948 is not equal to zero. Then the PLC interpreter deletes the marker M948 in order to prevent a cyclic output; otherwise the message cannot be acknowledged.

Nevertheless, the programmer must also pay attention that the marker M948 won't be described cyclically in the PLC program, otherwise the message cannot be acknowledged, either.

The marker M1021.1 indicates to the PLC that the message had been acknowledged.

6 System variables and machine data

6.1 SYSTEM VARIABLES

_SSTARTPLC default: 16 blanks

Contains the name of the PLC program which shall be taken over into the cyclic program treatment automatically after switch-on of the control or with the command Start PLC (see chapter 2.2 and 2.3).

The program name in _SSTARTPLC is invalid if the first character is a blank. After the program name there must be a blank (string end identification).
A path indication prefixed to the program name is allowed.

_SPLC_PATH default: S:/PLC

Contains the path for the start program. The complete path with program name results from _SPLC_PATH + _SSTARTPLC.

_ITIME_PLC default: 30 (0,3 s)

Contains the max. permitted cycle time in the unit 1/100 s for the active PLC program (watchdog).

6.2 MACHINE DATA

IPLC_LOAD

format:	4 Bytes integer
default:	20
range of values:	0 to xx

Contains the number of steps being treated by the PLC-interpreter per IPO-cycle.
Example: runtime for a PLC-program with 100 commands with an IPO-cycle = 10ms and
IPLC_LOAD = 20 : 50ms

At JSPS_TIME = 0 the PLC is deactivated, i. e. the robot control contains the total calculating time.

This value is adjusted only once during start-up of the robot control and must only be changed after consultation with Reis company.

7 Appendix: Summary of the System Marker Allocation

The system markers are in the range M 512 to M 2047. The following markers are occupied at the time being:

MARKER	Function	described by:
512 - 899	reserved	
900 - 903	oscillation frequency in Hz/100	PLC / robot
904 - 907	oscillation amplitude in mm/100	PLC / robot
908 - 911	waiting time until arc on in 0,01 sec	robot
912 - 915	waiting time after arc on (preheating time in 0,01 sec.)	robot
916 - 919	error response time during welding in 0,01 sec.	robot
920 - 923	waiting time until arc off in 0,01 sec	robot
924 - 939	reserved	
940 - 943	vision system data	
944 - 947	error number of the ROBOTstar V	robot
948 - 951	Error code for the output of the user defined PLC-errors. Bit #31 reserved for distinction between freely defined clear text message or standard message "PLC-error".	PLC
952 - 955	parameter 1 for clear text message	PLC
956 - 959	parameter 2 for clear text message	PLC
960 - 963	parameter 3 for clear text message	PLC
964 - 967	parameter 4 for clear text message	PLC
968.0	request for a program change to the program from the system variable _SNMIPRG	PLC
968.1	Auto Start if status changes from 0 to 1 Active only in AUTO	PLC
968.2	Auto Stop if status is 1. Active only in AUTO	PLC
968.3	activate lamp test with marker = 1	PLC
968.4 ... 968.7	reserved	
969.0	welding on = 1 / off = 0	robot
969.1	behavior with welding off 0 = default, 1 = stop path until source off	robot
969.2	source working, arc on = 1	PLC
969.3	preliminary sensor correction active	robot
969.4.... 971.7	reserved	

972 - 987	reserved	
988 - 991	system date	robot
992 - 995	system time	robot
996 - 999	input (Bit 0 to 15), level (Bit 16 to 23 and status (Bit 31) if an input inquiry is active in the robot program.	robot
1000 - 1003	system inputs (_ISYS_IN[2])	robot
1004 - 1007	system outputs (ISYS_OUT[2])	robot
1008 - 1011	system inputs (PCIN/_ISYS_IN[1])	robot
1012 - 1015	system outputs (PCOUT/_ISYS_OUT[1])	robot
1020.3	stops the robot movement as long as Bit = 1, no interpreter abortion! Not implemented	PLC
1020.4	Power-On Bit	robot
1020.5	error marker for the commands MU and DI	PLC
1020.6	robot-synchronous marker	robot
1020.7	stops robot movement as long as Bit = 1 (not Auto Stop)	PLC
1021.0	control in error status marker 944 - 947 contains error code	robot
1021.1	The error code entered in the marker double word 944 and also 948 was displayed and acknowledged	robot
1021.2	System initializations after Power-On are ready.	robot
1021.3	plus key pressed	robot
1021.4	minus key pressed	robot
1021.5	robot moves	PLC
1021.6.	PLC-test display is switched on	PLC / robot
1022 - 1023	operating mode	robot
1024 - 1119	actual values of axes 1 - 24	robot
1120.0	content of markers 1124...1135 is valid	robot
1122 - 1123	station number of the current station with table coordinate system	robot
1124 -	cartesian TCP-coordinates X, Y, Z in the current	robot

1135	coordinate system in 0.01 mm	
1136 - 1147	orientation angles A, B, C of the cartesian position in 0.01 degrees	robot
1148 - 1243	preset of the nominal position with movement of axes by the markers 1440...1447 preset in increments not implemented	PLC
1244.- 1339	preset of the acceleration with movement of axes by the markers 1440...1447 preset in % (1..100) of the nominal acceleration not implemented	PLC
1340 - 1435	Override for robot axes with external hand movement. The movement speed is multiplied again by this override value. Unit: % Range of values: +1.0 to +100.0 in DW16 not implemented	PLC
1436 - 1439	Override of 0% ... 100% by which the system override is multiplied. Only in AUTO, TEST2 and TEST4 operation active. Thus, a sequence program can be run with reduced speed, controlled by the internal PLC.	PLC
1440 - 1447	Markers for control of externally movable axes Marker 1440 ...1443 positive marker 1444...1447 negative not implemented	PLC
1448 - 1451	Markers for switching-over "speed or position control" (bit-coded for each axis) with movement through marker 1440...1447 0 -> speed control 1 -> position control not implemented	PLC
1452 - 1455	Bit-coded "axis is in position" with positioning control through marker 1440...1447 not implemented	robot
1456 - 1459	Station switch-over for the multi axes' transformation 1456.0 -> Station 1 1456.1 -> Station 2 etc. 1 means active 0 means inactive The control via PLC must be activated in JGRUP-TAB not implemented	PLC
1460 - 1463	Station switch-over for freeing of station axes. Significance like 1456.x The control via PLC must be activated in JSTMASKE not implemented	PLC
1464 - 1467	length of the current path until now in 1/100 mm, copy of _RPATH_DIST[1]	robot
1468 -	residual length of the current path in 1/100 mm,	robot

1471	copy of _RPATH_DIST[2]	
1471 - 1475	duration of the current path until now in 1/100 sec, copy of _RTIME_DIST[1].	robot
1476 - 1479	residual duration of the current path in 1/100 sec, copy of _RTIME_DIST[2].	robot
1480 - 1871	reserved	
1871.7	Toggle-Bit for the handshake with an external PLC	PLC
1872 - 1919	Markers containing the analog input values. marker allocation M1872...M1873 = analog input 1 M1874...M1875 = analog input 2 : : M1918...M1919 = analog input 24 Format: 16-Bit-A/D-converter data with sign bit. Conversion: marker value * 10.0[Volt] / (2**15 - 1)	PLC / robot
1920 - 1967	Markers containing the analog output values. marker allocation: M1920...M1921 = analog output 1 M1922...M1923 = analog output 2 : : M1966...M1967 = analog output 24 Format: 16-Bit-A/D-converter data with sign bit. Conversion: marker value * 10.0[Volt] / (2**15 - 1)	PLC / robot
1968 - 2007	Markers containing the binary inputs M1968 = input 0... 7 M1969 = input 8... 15 M1970 = input 16... 23 M1971 = input 24... 31 M1972 = input 32... 39 : : M2007 = input 312...319	robot
2008 - 2047	Markers containing the binary outputs M2008 = output 0... 7 M2009 = output 8... 15 M2010 = output 16... 23 M2011 = output 24... 31 M2012 = output 32... 39 : : M2047 = output 312...319	robot (PLC?)

The following operating modes are possible:

Main operating mode	Sub-operating mode	Code
HAND	HAND/AXIS	0101
	HAND/CART	0102
	HAND/ORI	0103
TEST	TEST_1	0201
	TEST_2	0202
	TEST_3	0203
	TEST_4	0204
AUTO	AUTO	0300
SPEC MODE	HOME_POS	0801

DOCUMENTATION

REIS GMBH & CO MASCHINENFABRIK OBERNBURG

Operating instructions

Title: 6D-Mouse: Calibration and traversing

Author:	May
Date:	13.11.1998
Control version:	RSV
Software revision:	1.6

Date: 01.12.98
Release: *Elter*

1 Task description

With the 6D-mouse an intuitive movement of the robot should be possible. With the movement of the robot with the keyboard of the teach pendant (PHG) you can only drive or rotate along the axes of the coordinate system just adjusted. In this case you need to know the directions of the three axes in the corresponding system of coordinates. It makes it more difficult, that by changing of the site of the server a permanent "rethinking" concerning the axes of the coordinates is necessary.

When using the method with the 6D-mouse this restriction does not exist any more. Now the robot can be "pressed" or "pulled" in any direction you like. So the approach of positions by hand is more facilitated.

The 6D-mouse, however, only offers an alternative to moving of the robot by PHG-keys. **Nothing changed when entering space points in a user program, therefore contours are not stored.**

2 Operating instructions

2.1 Conditions

The functionality for calibration of the 6D-mouse and the movement with the 6D-mouse is already stored in the RSV-system software.
The adaptions only refer to the fixing of the 6D-mouse fixture on the tool and on the PHG. Also the control PHG must be fitted with a 6D-mouse interface which is on the PHG reverse in the form of a socket with 5 poles.

2.2 Calibration of the 6D-Mouse

2.2.1 System programs for calibration

For calibration of the 6D-Mouse there already exist the three programs „MOUSE_KAL1", „MOUSE_KAL2" and „MOUSE_PHG". These programs are in the list S:/$SYSTEM/$DATA and can be chosen with the PHG-Key „Coord". The menu points „Run Kal1", „Run Kal2" and „Run K_PHG" which are in the second menu are for this purpose.

Additionally the subdirectory S:/$SYSTEM/$DATA contains the program „MOUSE_DAT" with the three global variables „V_KAL1[6]", „V_KAL2[6]" and „V_KALPHG[6]. These variables contain the 6D-mouse-data read in of the calibration carried out last with the following allocation:

V_KAL...[1]	X-Robot	V_KAL...[1].x contains coordinate 1 of the 6D-mouse V_KAL...[1].y contains coordinate 2 of the 6D-mouse V_KAL...[1].z contains coordinate 3 of the 6D-mouse
V_KAL...[2]	Y-Robot	V_KAL...[2].x contains coordinate 1 of the 6D-mouse V_KAL...[2].y contains coordinate 2 of the 6D-mouse V_KAL...[2].z contains coordinate 3 of the 6D-mouse
V_KAL...[3]	Z-Robot	V_KAL...[3].x contains coordinate 1 of the 6D-mouse V_KAL...[3].y contains coordinate 2 of the 6D-mouse V_KAL...[3].z contains coordinate 3 of the 6D-mouse
V_KAL...[4]	Rotation around hand-X	V_KAL...[4].a contains coordinate 4 of the 6D-mouse V_KAL...[4].b contains coordinate 5 of the 6D-mouse V_KAL...[4].c contains coordinate 6 of the 6D-mouse
V_KAL...[5]	Rotation around hand-Y	V_KAL...[5].a contains coordinate 4 of the 6D-mouse V_KAL...[5].b contains coordinate 5 of the 6D-mouse V_KAL...[5].c contains coordinate 6 of the 6D-mouse
V_KAL...[6]	Rotation	V_KAL...[6].a contains coordinate 4 of the 6D-mouse

around	V_KAL...[6].b contains coordinate 5 of the 6D-mouse
hand-Z	V_KAL...[6].c contains coordinate 6 of the 6D-mouse

2.2.2 Calibration at the tool

For calibration of the 6D-mouse on tools there are two programs „MOUSE _KAL1" and „MOUSE_KAL2". The program „MOUSE_KAL2" has the same structure as „MAUS_KAL1" and is only then necessary, if a second 6D-mouse-fixture is fixed on the tool.

„MOUSE_KAL1" describes the variable „V_KAL1[6]" and „MOUSE_KAL2" describes the variable „V_KAL2[6]". Thus two different tool calibrations can be stored and activated if required (fixing of the mouse to fixture 1 or fixture 2).

The calibration programs are structured in this way that starting from the actual robot position one after the other a movement to hand-X-direction, to hand-Y-direction, to hand-Z-direction and a rotation around the hand-X-axis, around the hand-Y-axis and around the hand-Z-axis is carried out. The TCP covers with the linear movement a way of -15mm and with the rotations an angle change of -5 degree is passed.

Sequence:

- Adjust at the PHG with the key selector switch the operating mode AUTO-TEST.
- Select the calibrating program. (PHG-key „Coord" and selection of the corresponding calibration with „Run Kal1" or „Run Kal2" in the 2nd function key menu)
- Preset the operating mode TEST2 (PHG-key „Run").
- Fix the 6D-mouse on the tool.
- Adjust the override to about 30%.
- Switch on the drives with the permission key.
- Start the calibration program (press PHG-key START).
- Detain the 6D-mouse so long until the first movement is over. At the 6D-mouse an excursion must arise opposite to the movement direction. Attention: The data of the 6D-mouse are read in only after the end of the movement, therefore don't let go the 6D-mouse immediately.
- Bring back the 6D-mouse to zero position (let go shortly and detain again at once).
- Repeat the last two points for each direction of the movement.

2.2.3 Calibration at the PHG

The program "MAUS_PHG" serves for the calibration of the 6D-mouse at the PHG. This program describes the variable „V_KALPHG[6]" and is structured in this way that starting from the actual robot position one after the other a movement to world-X-direction, to world-Y-direction, to world-Z-direction and a rotation around the hand-X-axis, around the hand-Y-axis and around the hand-Z-axis is carried out. The TCP covers with linear movements a way of -50 mm and with the rotations a modification of angle of -10 degree is given.

Sequence:

- Adjust with the key selector switch the operating mode AUTO-TEST on the PHG. If you consider the safety prescriptions concerning the protection of persons the operating mode AUTO is also possible.
- Select the calibration program. (PHG-key „Coord" and selection of the corresponding calibration with „Run K_PHG" in the 2nd function key menu)
- Preset the operation mode TEST2 or AUTO (see first point/PHG-key „Run").
- Fix the 6D-mouse on the tool.
- Adjust the override to about 30%.
- Switch on the drives with the permission key or with the key „Antriebe ein" (drives on).
- Start the calibration program (press PHG-key START).
- No longer twist the PHG and follow-up with the 6D-mouse the movement of the TCP, that means turn the 6D-mouse in that direction in which the TCP moves. Attention: The data of the 6D-mouse are read in only after the end of the movement, therefore do not let go the 6D-mouse at once.
- Bring back the 6D-mouse into zero position while the robot drives back into initial position (let go shortly and detain again).
- Repeat the last two points for each direction of the movement.

2.3 Activation of the calibration

As already said, the results of the calibrations are in the variables „V_KAL1[6]" „V_KAL2[6]" and „V_KALPHG[6]". After having started a calibration program the corresponding calibration is activated automatically. As long as the contents of the vector variables are not destroyed it is enough to recall simply the calibration data once found out with the help of the PHG-key „Coord" and one of the menu items „Aktiv Kal1", „Aktiv Kal2" or „Aktiv K_PHG" in the second function key menu.

To each of these menu items belongs a macro which is carried out after having operated the corresponding key.

Menu item	Macro
Aktiv Kal1	S:/$SYSTEM/$SYSMAK/AKTI_KAL1
Aktiv Kal2	S:/$SYSTEM/$SYSMAK/AKTI_KAL2
Aktiv K_PHG	S:/$SYSTEM/$SYSMAK/AKTI_KPHG

In these programs the system command „MAUS_KAL IB" is carried out and the Bit 1 in the system variable „_VERFAHR[3]" is set or is deleted.

The system command „MAUS_KALIB V..." calculates from the data of the vector variable V... a transformation matrix for a translationary and for a rotary movement and stored these data in the system variables „_RMOUSE_T" and „_RMOUSE_R".

The Bit 1 in „_VERFAHR[3]" determines, whether with the procedure the 6D-mouse-data should be additionally transformed into the hand system of coordinates. This is necessary only, when the 6D-mouse is fixed on the tool and the system of coordinates must be followed up.

Bit 1 = 0: Transformation of 6D-mouse to world
Bit 1 = 1: Transformation of 6D-mouse to hand

2.4 Data back-up

As long as nothing is changed on the 6D-mouse fixtures and as long as the content of the user program memory is not disturbed, a single calibration is enough for each available 6D-mouse-fixture. By the buffering of the user program memory the calibration data are not lost even if you switch off the control.

Nevertheless it is recommended to store the program "S:/$SYSTEM/$DATA/MAUS_DAT" on disk. After a data loss it is possible in a simple way to restore the first calibration stand by reading in this program.

2.5 Movement with the 6D-mouse

The robot is to be handled with the 6D-mouse only in the movement modes „T_XYZ" (translationary) and „R_ABC" (rotatory).

An unwillingly change of the orientation angles or the TCP-coordinates, while movement is made with the 6D-mouse, can be prevented by blocking of the corresponding movement mode (translationary or rotatory). The two menu points „T_XYZ mouse" and „R_ABC mouse" underneath the PHG-key „Coord" are meant for this.

Otherwise the same conditions are valid like for the procedure by PHG-keys:
- The robot must be synchronous.
- The drives must be switched on.
- The movement mode „T_XYZ" and/or „R_ABC" must be chosen.

The robot is moved by simple excursion of the 6D-mouse. With the right calibration the robot moves in this direction in which it is guided. Hereby the speed of the robot is proportional to the amplitude and reaches with maximal excursion of the 6D-mouse and with an override of 100% the maximal possible hand movement speed of _RVEL_KART and _RVEL_ORI.

3 Error messages

„Calibration not possible“

This error message shows that for at least one movement direction no data are read in by the 6D-mouse. At least one array element of the vector variables mentioned in the command „MAUS_KALIB“ contains only zero-values.

Possible causes:

- The 6D-mouse interface is defective
- The 6D-mouse was not guided during the calibration
- The 6D-mouse was let go too early during the calibration

4 Machine Data

IMOUSE_MIN

Threshold value for the reading in of the 6D-mouse-data. Data which are smaller than IMOUSE_MIN (amount) are replaced by zero.

Value = 0 if -IMOUSE_MIN < 6D-mouse-coordinate < +IMOUSE_MIN

Default: IMOUSE_MIN = 30

IMOUSE_MAX[1] to IMOUSE_MAX[6]

Maximal values for the standardization of the 6D-mouse-data to the maximal hand movement speed. The resulting speed is limited in every case to the value of _RVEL_CART or _RVEL_ORI.

Default:
IMOUSE_MAX[1] = 520
IMOUSE_MAX[2] = 400
IMOUSE_MAX[3] = 440
IMOUSE_MAX[4] = 520
IMOUSE_MAX[5] = 400
IMOUSE_MAX[6] = 450

DOCUMENTATION

REIS GMBH & CO MASCHINENFABRIK

TRAVERSING MODULES

Title: **Transformation of user programs (Transformation function)**

Authors: May, Dr. Dresselhaus

Date: November 1998

Control: RSV

Software: 2.0

Date: 1.12.98
Release: Elter

TABLE OF CONTENTS

1. Description of task

The transformation function of the robot control enables the user to transform a generated and fixed programmed user program (AWP = UP) in a suitable manner.

Depending on the defined transformation parameters the following tasks can be solved with the transformation function:

1.) User programs can be taken for shifted and/or turned workpieces without having to correct teach-in of the position steps.

2.) User programs can be used on several installations with almost identical arrangement of the work envelope without any correction of teaching.

3.) User programs can also be used for workpieces being mirror-inverted to the original workpiece without correction teaching of positions.

4.) User programs can be used for similar workpieces having a defined ratio of size to the original workpiece.

5.) User programs for workpieces with plane, straight line or point symmetries can be also used for the mirror-inverted workpieces.

2. Definition of the user command TRAFO_6D

Transformation of user programs is executed with the user command TRAFO_6D. With this command a source program is transferrred into a destination program. The position steps of the source program are transformed depending on the transformation parameters; all other steps are copied into the destination program without any modifications.

It is possible to maintain the hand/tool orientation of the source program or to transform it.

For all transformations it has to be observed that the transformed positions are within the robot work envelope so that approach with robot is possible. With extreme shiftings or twistings of the coordinate systems there might occur transformation errors after transformation.

The user command TRAFO_6D executes the transformation of the position steps of the source program depending on the 5 command parameters.

The command is defined as follows:

TRAFO_6D	Ref_source	ref_q,
	Ref_destination	ref_z,
	Source	quelle,
	Destination	ziel,
	Control word	kontr_var

The significance of the command parameters is indicated in chapter 2.1. Chapters 2.2 to 2.6 contain the descriptions for the transformation procedures.

2.1 Definition of the command parameters

a) Program name of the program "reference-source"

This program contains the definitions for the coordinate system of the source program. Its structure depends on the mode of transformation and is described for the individual transformation modes.

b) Program name of the program "reference-destination"

This program contains the definitions for the coordinate system of the destination program. The structure depends on the transformation mode and is described for the individual transformation modes.

c) Program name of the source program

The source program contains the program to be transformed. The source program is arbitrarily structured. All position steps of the source program are transformed. Main and sub-programs can be transformed with the command TRAFO_6D.

d) Program name of the destination program

The destination program is generated by the command TRAFO_6D. If overwriting of the destination program is allowed, an already existing program with the same name will be deleted prior to this. If overwriting is not allowed, this will cause an error message in this case. The program type of the source program (main or sub-program) will be maintained.

If the source program contains global variable definitions, they are copied into the destination program and will be invalidated there, in order to avoid double definitions. If the names of the source and the destination program are identical, i.e. the source program itself is modified, the variable definitions remain valid.

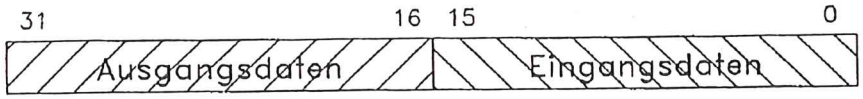
In the normal case the program names are entered during programming of the command as fixed character strings. Besides this, automatic generation of the names is possible. In this case, the program name is indicated by the name of the string variable (S...) instead of "prg_name".

e) Control variable

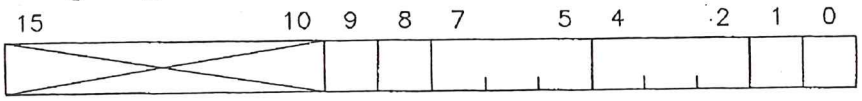
The control variable has to be of the integer type and consists of 4 bytes - 2 bytes each for data input and output. The structure of the control variable is shown in **ill. 1**.

The input data are entered in the variable prior to calling the command and will be kept even after processing of the command. The output data are entered in the output array of the variable with the transformation command. The error numbers entered in the output data array and their significance are summarized in the table of chapter 4.

Kontroll-Variable



Eingangsdaten



Bit 0: 0 evtl. vorhandenes Zielprogramm nicht überschreiben
1 evtl. vorhandenes Zielprogramm überschreiben

Bit 1: 0 Orientierung wird transformiert
1 Orientierung wird nicht transformiert

Bit 2-4: Transformationsart
000 Shiften
001 Strecken/Stauchen/Spiegeln
010 Shiften mit Spiegeln

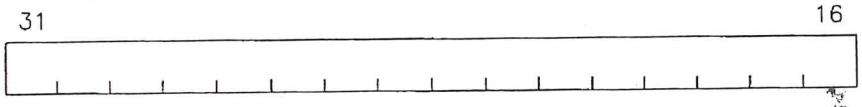
Bit 5-7: Transformationsparameter
000 Offset in kartes. Koordinaen } bei Transformationsart
001 Offset in Zylinderkoordinaten } Shiften

000 Bezugssystem Ebene
001 Bezugssystem Gerade } bei Transformationsart Strecken/Stauchen/Spiegeln
010 Bezugssystem Punkt

Bit 8: 0 keine Fehlerausgabe auf PHG
1 Fehlerausgabe auf PHG

Bit 9: 0 Auswertung aller Fehler der Vorzeichenberechnung
1 nur fatale Fehler der Vorzeichenberechnung werden ausgewertet

Ausgangsdaten



Bit 16-31: Fehlernummer

III. 1: Structure of the control variable

2.2 General 6D-transformation

Due to the general 6D-transformation it is possible to shift a user program taught with regard to a certain coordinate system into all 6 degrees of freedom (3 translatory and 3 rotatory ones).

Doing so, you assume for instance that a machining program has been taught for a workpiece which has a fixed position regarding a workpiece carrier. If the workpiece carrier was displaced within the robot work envelope or was transferred to another machine, usually all positions of the processing program must be taught again. This correction of teaching is avoided with the command TRAFO_6D.

2.2.1 Preset of shift by taught positions

For use of the 6D-transformation it is necessary to select three significant points of the workpiece carrier in the original position, to approach them with a defined tool (tip or similar) and to file them as positions in the program "reference-source". The same three points are also taught on the displaced carrier and stored in the program "reference-destination".

The original program (source program) is processed with the TRAFO_6D command and transferred into a destination program. By means of the destination program the workpiece is now machined in the shifted position **(ill. 2).**

When storing the positions in the reference programs, it has to be observed that all positions are defined in the same coordinate system. Either all positions have to be within the basic coordinate system (frame number = 0) or in a freely defined coordinate system (frame number >0).

2.2.2 Numerical preset of shift

Besides the source and destination coordinate system given by 3 reference points each, the shifting vector can also be given numerically.

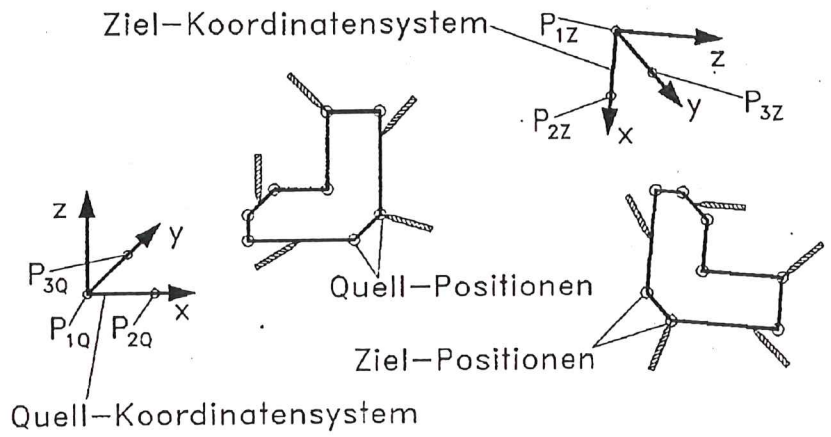
Depending on the transformation parameter being entered in the control variable (see ill.1), the offsets of the shift are stored in the components of a real array in the following manner.

The following values must be entered in the real array:

	cartesian coordinates	cylinder coordinates
1	delta x	delta rho
2	delty y	delta phi
3	delta z	delta z
4	delta A	delta A
5	delta B	delta B
6	delta C	delta C

For definition of the individual parameters please see ill. 8.
If a robot is equipped with additional axes, another array element has to be reserved for each additional axis; the offsets for adjustment of additional axes will be entered there (see chapter 2.7).

Structure of the reference programs for general 6D-transformation is defined in chapter. 3.1.



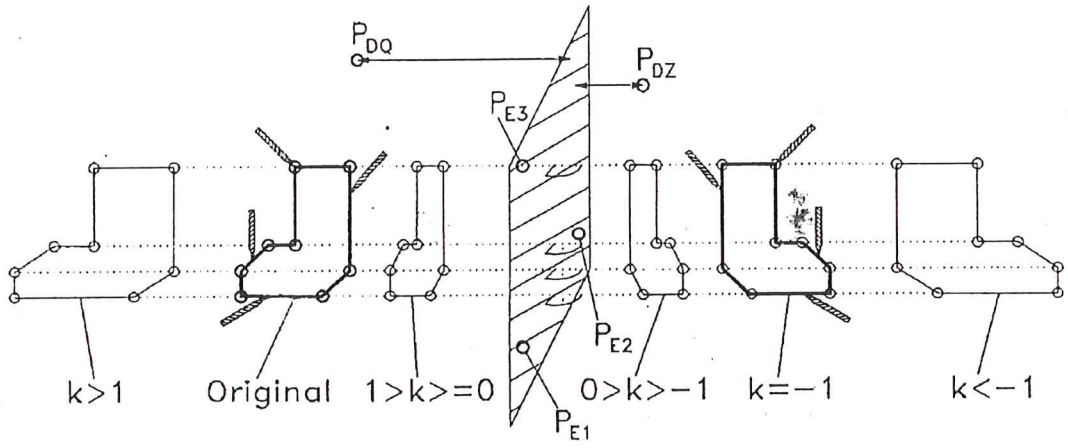
III. 2: General 6D-transformation

2.3 Expansion, shrinking and mirror inversion of user programs on a plane

By means of this transformation mode the user is enabled to reduce the programming efforts for workpieces with symmetries with regard to one plane. Furthermore, by means of the transformed user programs, without any correction of teaching it is possible to machine similar workpieces having a fixed ratio of size with regard to one plane or mirror-inverted workpieces having a plane symmetry.

The reference plane for the transformation (mirror plane) is given by three positions which are stored in the program "reference-source".

The size factor can be given in two ways: by numeric indication in a real variable or constant or by indicating two positions having a certain distance from the reference plane. The ratio of distance between the two positions indicates the size factor for the transformation **(III. 3).**



III. 3: Transformations regarding a plane

With the size factor k all positions of a source program with regard to the plane indicated in "reference-source" are expanded, shrinked and also mirror-inverted in case the k-factor has a negative sign.

Depending on the k factor, the following is possible:

$k > 1$	expansion
$1 > k \geq 0$	shrinking
$0 > k > -1$	shrinking and mirror inversion
$k = -1$	mirror inversion
$k < -1$	expansion and mirror inversion

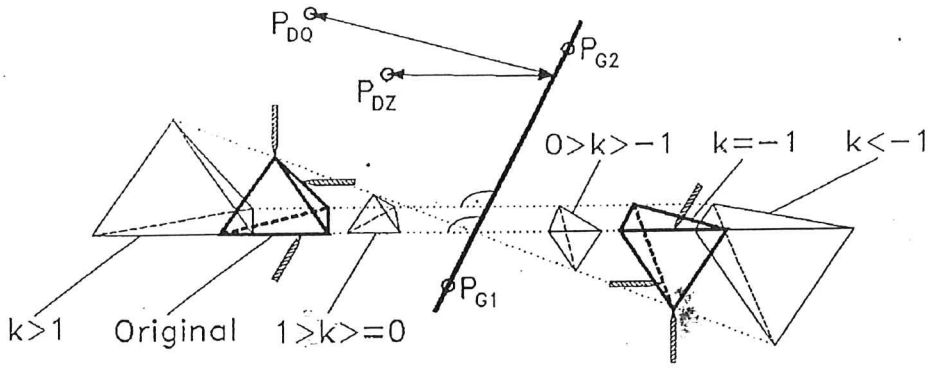
Structure of the reference programs is defined in chapter 3.2 .

2.4 Expansion, shrinking and mirror inversion of user programs on a straight line

This transformation mode can be used for workpieces having a symmetry with regard to a straight line (rotation symmetry).

Analogue to the plane symmetry there are the same possibilities for transformation of programs.

The reference straight line is given by two positions which are stored in the "reference-source" program. For indication of the size factor k also in this case numeric indication in a real variable or constant is possible, as well as the indication by two positions with defined distances to the reference straight line. The ratio of distance results in the size factor k **(III. 4).** The direction of the positions P_{DQ} and P_{DZ} is not taken into consideration when calculating k.



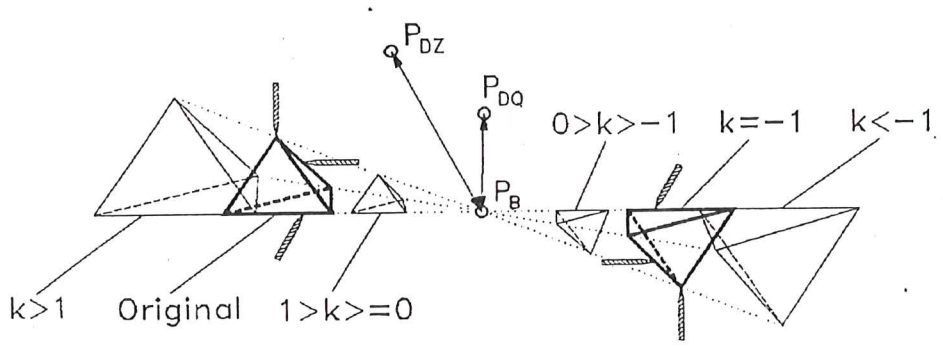
III. 4: Transformations regarding a straight line

Structure of reference programs for transformation of user programs regarding a straight line is defined in chapter 3.3 .

2.5 Expansion, shrinking and mirror inversion of user programs in a point

This transformation mode is provided for point-symmetric workpieces. User programs for machining of point-symmetric workpieces having a certain ratio of size or for mirror-inverted workpieces are transferred with this transformation mode **(III. 5).**

The reference point of the transformation is stored as position in "reference-source". The size factor k is given as real variable or constant analogue to the plane and straight line transformation or is calculated from the ratio of distance of two positions from the reference point. The direction of the positions PDQ and PDZ is not taken into consideration when calculating k.



III. 5: Transformations regarding a point

Structure of the reference programs for transformation of user programs regarding a point is defined in chapter 3.4 .

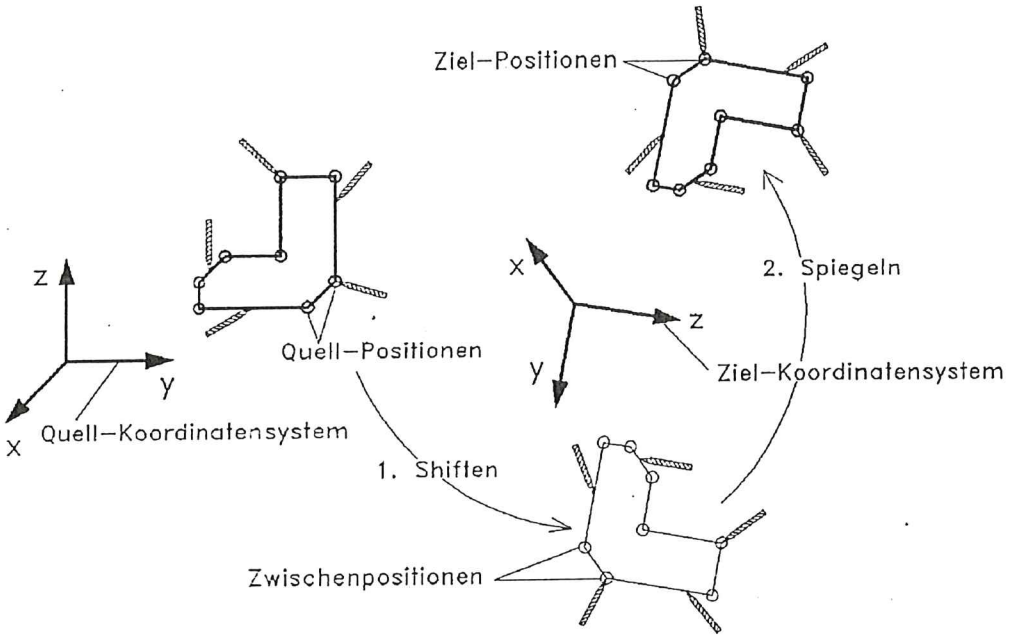
2.6 6D-transformation of user programs with simultaneous mirror inversion

The variant described here is especially suited for applications which require processing of mirror-inverted workpieces of same size and where exact determination of the mirror plane is difficult.

For preparation of the transformation only three significant points have to be selected at the original workpiece and are to be filed as positions in the program "reference-source". Furthermore, the corresponding points at the mirror-inverted workpiece must be stored as positions in the same order in the program "reference-destination".

Due to this transformation, the source program is shifted in all 6 dimensions with one step and then is mirror-inverted on the x-z plane of the destination coordinate system **(III. 6)**.

Indication of a size factor k is not possible for this transformation mode.



III. 6: 6D-transformation with mirror inversion

Structure of the reference programs is defined in chapter 3.5 .

2.7 Treatment of additional axes

Besides transformation of positions in the main axes system of the robot, also the additional axes of the robot are transformed with the command TRAFO_6D.

For robots being equipped with additional axes it has to be observed that during programming of the positions in the reference programs (position steps or position variables) the additional axes are exactly adjusted to each other in "reference-source" or "reference-destination".

For treatment of additional axes, there are two cases depending on the transformation mode.

In the normal case (without mirror inversion) the additional axes are adjusted by means of the presettings in the reference programs. In cases where source programs are mirror-inverted, the additional axes are mirror-inverted, too.

2.7.1 Transformation of additional axes without mirror inversion

For transformation of additional axes without mirror inversion, the differences of the increments of all additional axes are calculated.

Presetting for the adjustment is made in two manners:

a) If **two different reference programs** "reference-source" and "reference-destination" exist:

   The differences of the increments are calculated from the positions in "reference-source" and "reference-destination". The calculated differences are added to the incremental values of the corresponding additional axes of the positions of the source program.

b) If only **one reference program** exists:

   "reference-source" = "reference-destination"

   Adjustment of additional axes is made by numerical presettings; for rotary axes they are indicated in degrees, for linear axes they are indicated in millimeters.

   The additional axes are adjusted by the indicated angles (for rotary axes) or distances (for linear axes).

If the programs "reference-destination" and "reference-source" had been programmed at different stations, further measures will be required for transformation of the additional axes.

In this case, the kinematic order of the additional axes at both stations is checked (entries in the group table). If the table shows that different additional axes are active, the increments of the corresponding axes are calculated.

If for instance at station 1 (reference-source) axes 7, 8 and 9 are active and at station 2 (reference-destination) axes 7, 10 and 11, the increments of axes 8 and 10 as well as the increments of axes 9 and 11 are calculated with each other. The increments of axis 7 are calculated as described above.

2.7.2 Transformation of additional axes with mirror inversion

The transformation of additional axes with mirror inversion is executed if the following conditions are met:

- The reference programs "reference-source" and "reference-destination" are different,
- The frame numbers in "reference-source" and "reference-destination" are equal,
- The size factor "k" in transformation mode expansion/shrinking/mirror inversion is negative.

Treatment of additional axes results from the position steps of the reference programs.

1.) Additional axes having identical positions in "reference-source" and "reference-destination will not be transformed.

2.) Additional axes having different positions in "reference-source" and "reference-destination" will be mirror-inverted with regard to their reference position. This reference position results from the average value of the increments in the two reference programs.

2.8 Transformation of positions with different frame number

With the transformation function positions taught in the basic coordinate system (frame number = 0) as well as positions taught in a freely defined coordinate system (frame number > 0) can be transformed.

2.8.1 Conditions for transformation of different position types

1.) The reference programs "reference-source" and "reference-destination" must only contain positions with the same frame number.

2.) Frame numbers > 0 may vary between "reference-source" and "reference-destination". They have to coincide within the reference programs.

3.) The increments of the additional axes may vary between "reference-source" and "reference-destination". They have to coincide exactly within the reference programs.

2.8.2 Reference programs contain positions with frame number = 0

If the reference programs contain positions in the basic coordinate system, the source program may contain positions with random frame number. The frame number of the transformed positions is adopted from the corresponding position from the source program.

2.8.3 Reference programs contain positions with frame number > 0

If the reference programs contain positions in a freely defined coordinate system the source program must only contain positions with frame number = 0 or with same frame number like in the reference program.

The transformation allows:

- transformation of positions with frame number = 0 into positions with frame number = 0.

- transformation of positions with frame number > 0 into positions with that frame number the positions have in "reference-destination".

2.9 Conditions for transfer of programs to other machines

Cartesian coordinates are filed in user programs. The cartesian coordinate system for each robot is defined by measurement of the robot.

Due to wrong measurement of a robot the cartesian coordinate system differs from a real rectangular cartesian coordinate system. Thus, all programs taught with this robot contain faulty cartesian values. Since the taught (faulty) positions are always approached again in the same manner for the same robot, this error doesn't have an effect on programs which are not transferred to other machines.

If a program is transferred from one robot to another, the program will run faulty there, since now the different coordinate systems cause differences in positions.

In order to avoid such errors, already prior to programming of the original program the correct measurement of both robots has to be checked.

Correct measurement can be checked by a simple test. At the first machine, at least three points are taught at a random workpiece which should be as large as possible.

The workpiece and the program are transferred to the destination machine. The workpiece is placed in such way that the taught positions of the program coincide with those of the workpiece.

If this is not possible, both robots have a different measurement and the transformations are not possible in a correct way.

2.10 Remark concerning the used tool data

For all transformations executed with the TRAFO_6D command, tool data stored in the position steps of the source program are used.

If a position step doesn't contain any valid tool data, the current valid tool data are used for the transformation being active in the user program where the command TRAFO_6D is processed.

Positions without valid tool data are, for instance, positions generated by offline programming.

If a transformation mode is selected where positions are mirror-inverted, correct orientation of the tool in direction of the tool angle must be observed when mounting the tool.

This can be tested e.g. by moving in hand operation in the mode HAND/CARTESIAN. If the robot then won't move towards the tool direction during movement in U-direction (movement key +A/-A), this causes faulty orientations for mirror-inverted positions.

3. Structure of the reference programs

The reference programs of the command TRAFO_6D serve for definition of coordinate or reference systems for the different transformation modes. The format of the reference programs has to be exactly kept in accordance with the definitions given in chapters 3.1 to 3.5.

The reference programs "reference-source" and "reference-destination" can be selected identical or not identical. The structure of the programs is different for both variants.

3.1 General 6D-transformation

a) Reference programs not identical

The transformation parameters are given by 6 positions which determine the source and the destination coordinate system.

- program "reference-source"

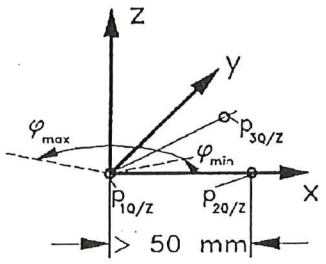
MP/SP	REFQ	or	MP/SP	REFQ
TOOL	T		TOOL	T
POSITION	P1Q		VAR_POS	P1Q
POSITION	P2Q		VAR_POS	P2Q
POSITION	P3Q		VAR_POS	P3Q
END			END	

- program "reference-destination"

MP/SP	REFZ	or	MP/SP	REFZ
TOOL	T		TOOL	T
POSITION	P1Z		VAR_POS	P1Z
POSITION	P2Z		VAR_POS	P2Z
POSITION	P3Z		VAR_POS	P3Z
END			END	

The three positions form a coordinate system each. P1Q or P1Z indicate the coordinate origin. P1Q and P2Q or P1Z and P2Z form the x-axis of the coordinate system. P3Q and P3Z are in the x-y-plane of the coordinate system **(Ill. 7)**.

To increase the accuracy a minimum distance of 50 mm is required for the positions; the angle has to be between 30 degrees and 150 degrees. If these conditions are disregarded, an error message will be displayed.



III. 7: Definition of the coordinate systems

b) reference programs identical

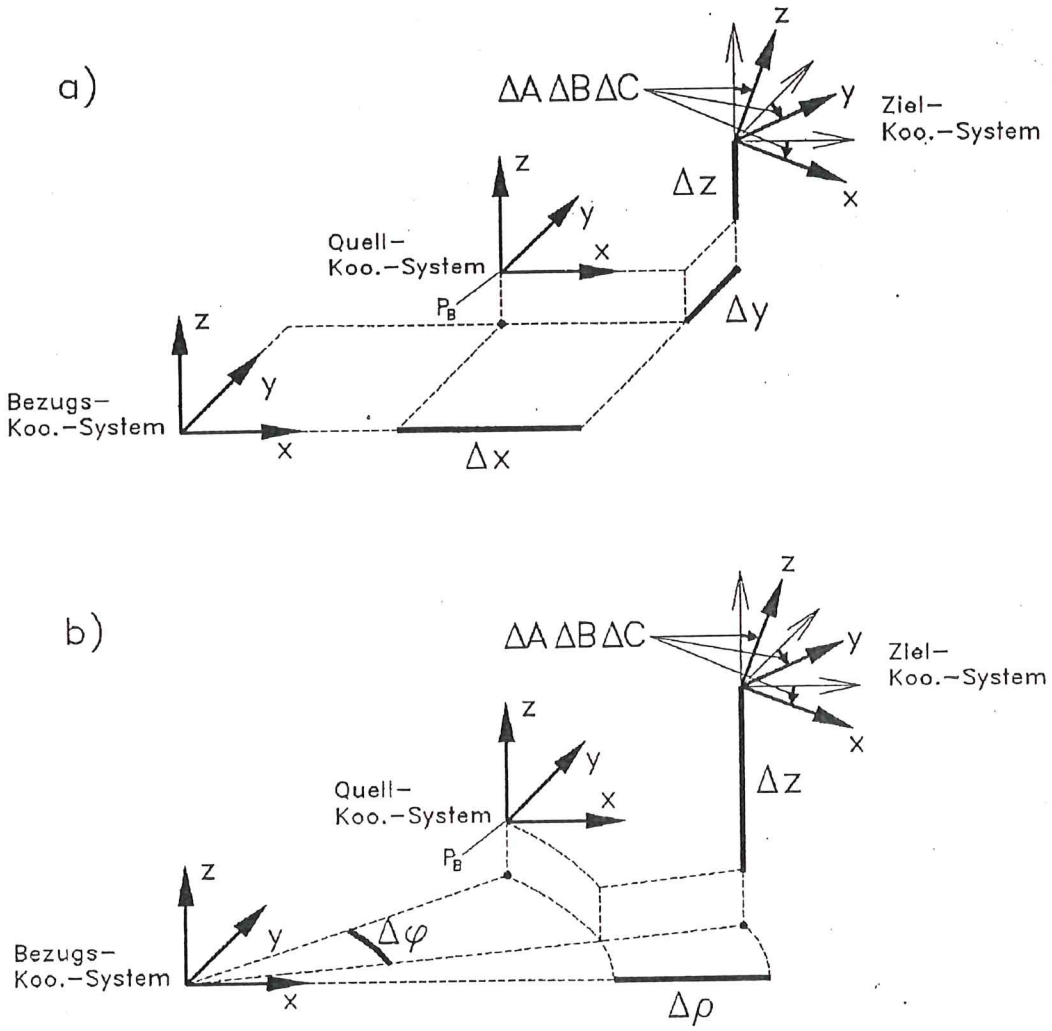
The transformation parameters are given by a position and a real type array.

- program "reference-source" = "reference-destination"

MP/SP	REFQ	or	MP/SP	REFQ
VAR	RARR[6]		VAR	RARR[6]
TOOL	T		TOOL	T
POSITION	PB		VAR_POS	PB
END			END	

The position PB is the reference point for the transformation.
In the real array RARR, the shifts and torsions are entered (see ill. 8). If shifting in cartesian coordinates is selected, delta x, delty y and delta z will be stored in the first three elements; with shifting in cylinder coordinates, delta rho, delta phi and delta z will be stored in the first three elements.

If there are additional axes, another array element has to be reserved for each additional axis. For an installation with 3 additional axes, consequently VAR RARR[9] has to be defined. The elements 7 to 9 in the above example contain the change of the additional axes 7 to 9 in degrees (for rotary axes) or mm (for linear axes).



Ill. 8: 6D-transformation with reference point and offset vector
a) offset indicated in cartesian coordinates
b) offset indicated in cylinder coordinates

3.2 Expansion, shrinking and mirror inversion on a plane

a) reference programs not identical

The size factor k is given by two positions.

- program "reference-source"

MP/SP	REFQ	or	MP/SP	REFQ
TOOL	T		TOOL	T
POSITION	PDQ		VAR_POS	PDQ
POSITION	PE1		VAR_POS	PE1
POSITION	PE2		VAR_POS	PE2
POSITION	PE3		VAR_POS	PE3
END			END	

- program "reference-destination"

MP/SP	REFZ	or	MP/SP	REFZ
TOOL	T		TOOL	T
POSITION	PDZ		VAR_POS	PDZ
END			END	

The size factor k is calculated with the ratio of the distances of the positions PDQ and PDZ from the reference plane.

k = distance (PDZ) / distance (PDQ)

If PDQ and PDZ are situated on the same side of the plane, k is set positive, if they are situated on different sides of the plane, k is set negative (mirror inversion!). The distances of PDQ and PDZ from the reference plane have to be 50 mm at least.

The reference plane is built by the three positions PE1, PE2 and PE3.

To increase the accuracy, just like for the general 6D-transformation, there is required a distance of 50 mm between the positions and an angle between 30 and 150 degrees.

If also additional axes shall be transformed during mirror inversion ($k < 0$), this is only possible within one station.

Presetting for the additional axes' transformation results from the position steps in the reference programs (see chapter 2.7.2).

b) reference programs identical

The size factor k is given numerically in a real variable or real constant.

- program "reference-source" = "reference-destination"

MP/SP	REFQ	or	MP/SP	REFQ
CONST/VAR	RVAR		CONST/VAR	RVAR
TOOL	T		TOOL	T
POSITION	PE1		VAR_POS	PE1
POSITION	PE2		VAR_POS	PE2
POSITION	PE3		VAR_POS	PE3
END			END	

The size factor k is indicated by RVAR. The reference plane is built by the three positions PE1, PE2 and PE3.

To increase the accuracy, just like for the general 6D-transformation, there is required a distance of 50 mm between the positions and an angle between 30 and 150 degrees.

If also additional axes shall be adjusted during transformation, another definition of the real variable is required.

Instead of the definition CONST/VAR RVAR, the definition must now be VAR RARR[n+1], "n" being the number of additional axes. Accordingly, for three additional axes the definition must be VAR RARR [4]. In the elements 2 to 4 in the above example, the additional axes' adjustments of axes 7 to 9 have to be entered. For rotary axes indication is made in degrees, for linear axes in mm.

3.3 Expansion, shrinking and mirror inversion on a straight line

a) reference programs not identical

The size factor k is given by two positions.

- program "reference-source"

MP/SP	REFQ	or	MP/SP	REFQ
TOOL	T		TOOL	T
POSITION	PDQ		VAR_POS	PDQ
POSITION	PG1		VAR_POS	PG1
POSITION	PG2		VAR_POS	PG2
END			END	

- program "reference-destination"

MP/SP	REFZ	or	MP/SP	REFZ
TOOL	T		TOOL	T
POSITION	PDZ		VAR_POS	PDZ
END			END	

The size factor k is calculated with the ratio of the distances of the positions PDQ and PDZ from the reference straight line.

k = distance (PDZ) / distance (PDQ)

Factor k is always positive for variant a); mirror inversion of programs is not possible. The distances of PDQ and PDZ from the reference straight line have to be at least 50 mm.

The reference straight line runs through the positions PG1 and PG2. To increase the accuracy, PG1 and PG2 must have a minimum distance of 50 mm.

Mirror inversion of additional axes is not possible.

b) reference programs identical

The size factor k is given numerically in a real variable or real constant.

- program "reference-source" = "reference-destination"

MP/SP	REFQ	or	MP/SP	REFQ
VAR/CONST	RVAR		VAR/CONST	RVAR
TOOL	T		TOOL	T
POSITION	PG1		VAR_POS	PG1
POSITION	PG2		VAR_POS	PG2
END			END	

The size factor k is indicated by RVAR. The reference plane is built by the three positions PG1 and PG2. In order to increase accuracy, a minimum distance of 50 mm between PG1 and PG 2 is required.

If additional axes are to be adjusted the reference program must be modified (see last paragraph in chapter 3.2, section b).

3.4 Expansion, shrinking and mirror inversion in a point

a) reference programs not identical

The size factor k is given by two positions.

- program "reference-source"

MP/SP	REFQ	or	MP/SP	REFQ
TOOL	T		TOOL	T
POSITION	PDQ		VAR_POS	PDQ
POSITION	PB		VAR_POS	PB
END			END	

- program "reference-destination"

MP/SP	REFZ	or	MP/SP	REFZ
TOOL	T		TOOL	T
POSITION	PDZ		VAR_POS	PDZ
END			END	

The size factor k is calculated with the ratio of the distances of the positions PDQ and PDZ from the reference point.

k = distance (PDZ) / distance (PDQ)

Factor k is always positive for variant a); mirror inversion of programs is not possible. The distance from PDQ and PDZ from the reference point has to be at least 50 mm. The reference point for transformation is given by the position PB.

Mirror inversion of additional axes is not possible.

b) reference programs identical

The size factor k is given numerically in a real variable or real constant.

- program "reference-source" = "reference-destination"

MP/SP	REFQ	or	MP/SP	REFQ
VAR/CONST	RVAR		VAR/CONST	RVAR
TOOL	T		TOOL	T
POSITION	PB		VAR_POS	PB
END			END	

RVAR indicates the size factor k. The reference point for the transformation is defined by the position PB.

If additional axes shall be adjusted, the reference program has to be changed (see last paragraph in chapter 3.2, section b).

3.5 6D-transformation with simultaneous mirror inversion

The structure of the reference programs for this transformation mode is identical to the one valid for general 6D-transformation. Only variant a) (reference programs not identical) is allowed.

- program "reference-source"

MP/SP	REFQ	or	MP/SP	REFQ
TOOL	T		TOOL	T
POSITION	P1Q		VAR_POS	P1Q
POSITION	P2Q		VAR_POS	P2Q
POSITION	P3Q		VAR_POS	P3Q
END			END	

- program "reference-destination"

MP/SP	REFZ	or	MP/SP	REFZ
TOOL	T		TOOL	T
POSITION	P1Z		VAR_POS	P1Z
POSITION	P2Z		VAR_POS	P2Z
POSITION	P3Z		VAR_POS	P3Z
END			END	

The structure of the source coordinate system is analog to the procedure described in 3.1 under a).

For building the destination coordinate system the position P3Z first is mirror-inverted in the x-z-plane of the destination coordinate system and then the destination coordinate system is formed by the positions P1Z, P2Z and the mirror-inverted position P3Z.

If additional axes shall also be transformed during mirror inversion, this is only possible within one station.

Presetting for additional axes' transformation results from the position steps in the reference programs (see chapter 2.7.2).

4. Table of error messages

When processing the command TRAFO_6D, two different modes of error messages are generated: errors given by the interpreter during processing of the command (programming errors) and errors being sent by the transformation function (transformation error).

a) programming error

description	remedy
"program in the stack"	check parameter of TRAFO_6D; rename destination program

b) Transformation error

Transformation errors are entered into the output array of the control register and can be evaluated after processing of the command TRAFO_6D.

Besides this, it is possible to indicate the error number on the PHG (teach pendant). For this purpose, bit 8 of the control word has to be set to "1" **(Ill. 1).** In this case the running program is aborted and the error number is indicated on the PHG.

The emitted error message is as follows:

TRAFO_6D-error: nnn

For the errors marked with (*) in the following table, after acknowledgement of the error message, the step content indication in addition is set to the currently processed step of the source program.

The significance of the error numbers is shown in the following table:

error no.	description	remedy
1	The program "reference-source" was not found	Generate "reference-source" program; check the parameters of the TRAFO-6D command
2	"reference-source" contains invalid step	Validate all steps in "reference-source"
3	Number of positions in "reference-source" is wrong	Check syntax of the "reference-source" program; check in the control variable, whether correct transformation mode was selected
4	Positions in "reference-source" don't build a coordinate system	Check positions in reference-source" (distances, angles)
5	Increments of the additional axes in "reference-source" are not identical	Identical adjustment of increments of the additional axes
6	"reference-source" has wrong format	Check syntax of the "reference-source" depending on the transformation mode
7	"reference-source" contains wrong step	Check syntax of the "reference-source" depending on the transformation mode
8	Definition of a variable/constant in "reference-source" not found	Program variable/constant definition
9	Variable in "reference-source" not initialized	Describe the variable with a valid value
10	Distance of the positions in "reference-source" is too small	Check positions; adjust distance > 50 mm
11	Angle between positions in "reference-source" is too small	Check positions; adjust angle > 30 degrees

error no.	description	remedy
12	Number of additional axes in the machine data (IAXES_DESCR) doesn't coincide with the one in the position step in "reference-source"	Modify machine data; teach positions again
13	"reference-source" contains positions with wrong frame	Teach all positions in "reference-source" and "reference-destination" in the same coordinate system
14	position in "reference-source" contains wrong frame number	Teach all positions in "reference-source" with frame number > 0 at the same station
15	Error with rotary table-world-transformation in "reference-source"	Check positions in "reference-source"; check calibration of the additional axes; check station number
50	real variable in "reference-source" not defined as array	Correct the definition of the real variable
51	real array in "reference-source" has wrong number of elements	Correct the definition of the real array. Adjust number of elements acc. to definition depending on the transformation mode.
101	The program "reference-destination" was not found	Generate program "reference-destination"; check parameters of the command TRAFO_6D
102	"reference-destination" contains invalid step	Validate all steps in "reference-destination"
103	Number of positions in "reference-destination" is wrong	Check syntax of the program "reference-destination"; check in control variable whether correct transformation mode was selected
104	Positions in "reference-destination" don't build a coordinate system	Check positions in "reference-destination" (distances, angles)

error no.	description	remedy
105	Increments of the additional axes in "reference-destination" are not identical	Identical adjustment of the increments of the additional axes
106	"reference-destination" has wrong format	Check the syntax of "reference-destination" depending on the transformation mode
107	"reference-destination" contains wrong step	Check the syntax of "reference-destination" depending on the transformation mode
108	Definition of a variable/constant in "reference-destination" not found	Program variable/ constant definition
109	variable in "reference-destination" not initialized	Describe the variable with a valid value
110	Distance of the positions in "reference-destination" is too small	Check positions; adjust distance > 50 mm
111	Angle between positions in "reference-destination" is too small	Check positions; adjust angle to > 30 degrees
112	Number of additional axes in the machine data (IAXES_DESCR) doesn't coincide with that in the position step in "reference-destination"	Modify machine data; teach positions again
113	"reference-destination" contains positions with wrong frame	Teach all positions in "reference-source" and "reference-destination" in the same coordinate system
114	position in "reference-destination" contains wrong frame number	Teach all positions in "reference-destination" with frame number > 0 at the same station

error no.	description	remedy
115	Error with the rotary table-world-transformation in "reference-destination"	Check positions in "reference-destination"; check calibration of the additional axes; check station number
150	"reference-destination" has the same name as "reference-source"	Check name of "reference-destination"; check transformation mode
151	"reference-destination" has not the same name as "reference-source"	Check name of "reference-destination"; check transformation mode/parameters
201	Source program not found	Generate source program; Check parameter of the TRAFO_6D command
202 (*)	Number of additional axes in the machine data (IAXES_DESCR) and of a position in the source program don't coincide	Modify machine data; teach position again
203 (*)	Error with rotary table-world-transformation in the source program	Check positions in the source program; check calibration of additional axes; check station number
204 (*)	Error with world-rotary table-transformation in the source program	Check positions in the source program; check calibration of additional axes; check station number
205 (*)	Position in the source program contains wrong frame number	Check positions in the source program; check station number of the reference programs
206 (*)	Source program contains VAR_POS step	Generate source program in such a way that no VAR_POS steps are contained.
301	Destination program exists already	Modify control-variable of the command TRAFO_6D (permit overwriting); Change name of the destination program
302	No memory for destination program	Check user program memory; back-up programs and delete if required

401	Control variable contains invalid value	Check value of the control variable and enter valid value acc. to definition

error no.	description	remedy
402	Control variable contains invalid transformation mode	Check value of the control variable and enter valid value acc. to definition
403	Control variable contains invalid transformation parameter	Check value of the control variable and enter valid value acc. to definition
501	Internal error during copying of the source program	Back-up current user memory and data; Contact Reis
502	Internal error during transformation	Back-up current user memory and data; Contact Reis
503	Internal error when reading the position data	Back-up current user memory and data; Contact Reis
504 (*)	Internal error during determination of the axes' signs in the source program	Check position steps of the source program
505 (*)	Internal error during determination of the axes' signs in the destination program	Check position steps in the source program; check transformation mode and parameters; check source and destination coordinate systems
506 (*)	Internal error during determination of the axes' signs in the destination program	Check position steps in the source program; check transformation mode and parameters; check source and destination coordinate systems
507	Internal error concerning axes' factors of the additional axes	Check machine data; correct adjustment of axes' factors of the additional axes

DOCUMENTATION

REIS GMBH & CO MASCHINENFABRIK OBERNBURG

PROGRAMMING

Title: **COPY-command**

Author:	Wenzel
Date:	04.12.98
Control version:	RSV
Software version:	2.3

Date: 07.01.99
Release: Elter