Antriebs- und Steuerungstechnik

# **BAPS2 Brief description**







# rho 3 BAPS2 Brief description

1070 073 035-105 (94.12) GB



Reg. Nr. 16149-03

© 1993

by Robert Bosch GmbH, All rights reserved, including applications for protective rights. Reproduction or handing over to third parties are subject to our written permission.

Discretionary charge 10.- DM

# 1. BAPS2

GENERAL	2
PROGRAM STRUCTURE	3
COMPILER STATEMENTS	4
PROGRAM DECLARATIONS	6
SUBROUTINE DECLARATIONS	7
STANDARD CONSTANTS	8
DATA TYPES	9
POINT DECLARATIONS	10
GLOBAL VARIABLES	11
ARRAY DECLARATIONS	12
I/O ARRAY DECLARATIONS	13
CHANNEL DECLARATIONS	14
PROGRAM FLOW CONTROL	16
LOGIC – ARITHMETIC OPERATIONS	18
VALUE ASSIGNMENTS	19
STANDARD FUNCTIONS	20
MOVEMENT INSTRUCTIONS	26
ACTUAL POSITION	28
MEASUREMENT POSITION	29
SPEEDS	30
V / A / D FACTORS	32
SLOPE MODES	33
BELT SYNCHRONIZATION	34
WORKING AREA LIMITS	35
WRITE / READ INTERFACES	36
FILES – I/O (READ)	38
FILES – I/O (WRITE)	39
PARALLEL PROCESSES	40
SPECIAL FUNCTIONS	41
TOOL – DAT	47
FIXED FILES	48
2. BAPSPIC	
BAPSPIC-SYNTAX	49
3. INDEX	53

# GENERAL

This manual provides a summary and brief description of the language instructions available in **BAPS2** and **BAPSPIC**.

Please refer to the **BAPS2** programming instructions for an in-depth description of the individual instructions of **BAPS2**.

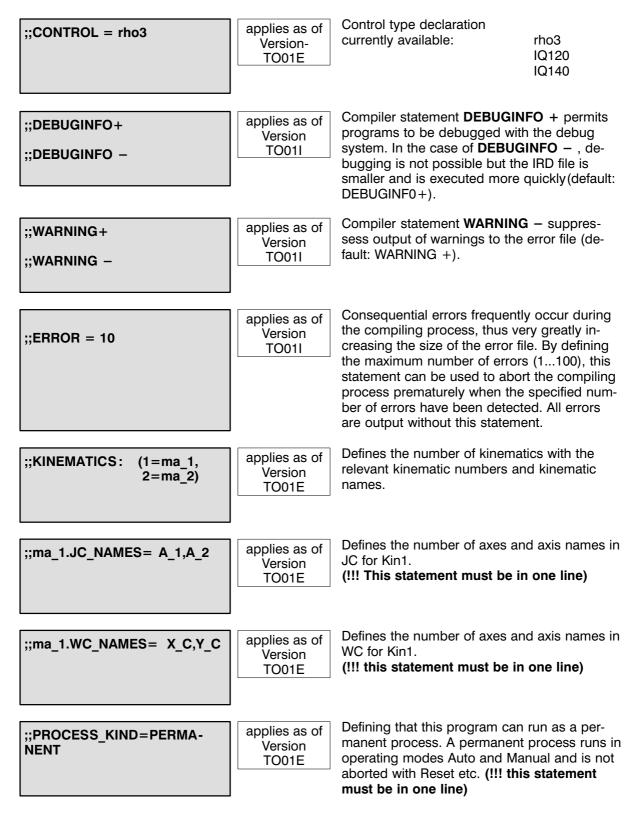
# **PROGRAM STRUCTURE**

;;CONTROL = RHO3	Control type declaration
;;KINEMATICS: ( 1 = ma_1 , 2 = ma_2 )	Number of kinematics with kinematic names (as defined in MPP).
;;ma_1.JC_NAMES = A_1,A_2,A_3,A_4 ;;ma_1.WC_NAMES = X_K,Y_K,Z_K,A_K	Number of axes and axis names in JC for Kin 1. Number of axes and axis names inWC for Kin 1
;;ma_2.JC_NAMES = B_1,B_2,B_3 ;;ma_2.WC_NAMES = X_B,Y_B,Z_B	Number of axes and axis names in JC for Kin 2. Number of axes and axis names inWC for Kin 2
PROGRAM example	Program name max. 8–character
;;KINEMATICS = ma_1	All movement instructions and point entries without kinematic entry relate to the kinematic ma_1 ( default kinematic).
EXTERNAL:maon, maoff INPUT: 1 = partthere, 2 = slide_forwar	Declarations for external subroutines or processes. Declarations for inputs/outputs
OUTPUT: $1 = belt on$	
SPC_FCT: 1 = path_io (VALUE INTEGER:	Declaration of the special functions
. )	
	Order of other variable declarations is arbi- trary, but they must be in front of <b>BEGIN</b> .
ma_1.POINT : pal_corner1,pal_corner2 ma_2.JC_POINT : @stroke_cent	Declaring points for different kinematics.
BEGIN	Main program begin
MOVE ma_2 TO @stroke_cent sub_example	Subroutine call
PROGRAM_END	Main program end
SUBROUTINE sub_example INPUT: 4 = le_shi REAL: counter	Subroutine identification (max. 12-character) Declaration part for the subroutine
BEGIN	Subroutine begin
belt_on = 1	
SUB_END	Subroutine end

3

# **COMPILER STATEMENTS**

Except the instructions **KINEMATICS**, **INCLUDE** and **INT** all compiler instructions must be declared before the program name. Lines containing compiler instructions should have no other characters following the statement.





;;DRIVE\_TYPE

;;INCLUDE exdat

;;KINEMATICS = ma\_2

(sr\_1.A\_1 = SM, sr\_2.A\_1 = SM\_CAN)

applies as of Version TO01E	Changes the default kinematic( the default kinematic is always no. 1 unless otherwise defined in this declaration.
	(!!! this statement must be in one line)
applies as of Version "ECO"	The drive type is defined. The statement is skipped by the BAPS 2 com- piler. It only affects the BAPS-ECO compiler
	for the BOSCH IQ120 control.
applies as of Version TO01E	The compiler includes file EXDAT.QLL at this point and generates the IRD code for it. EXDAT.QLL may have no other INCLUDE

;;INT = PTP	applies as of	Interpolation mode presetting
;;INT = LINEAR ;;INT = CIRCULAR	Version TO01E	The interpolation mode can be defined by the compiler statement in a program. The default is PTP.
;;ma_2.INT = PTP ;;ma_2.INT = LINEAR ;;ma_1.INT = CIRCULAR	applies as of Version TO01E	Ditto with kinematic assignment
;;SER_IO_STOP-	applies as of Version TO02F	The compiler statement can be used to pre- vent a user program being aborted if an inter- face error occurs.
;;SER_IO_STOP+	applies as of Version TO02F	The compiler statement indicates that a user program is to be aborted if an interface error occurs. The default is SER_IO_STOP+.

statements.

5

### **PROGRAM DECLARATIONS**

PROGRAM dana	applies as of Version TO01E	MAIN PROGRAM DECLARATION The program name may have a maximum length of 8 characters. The first character must be a letter. The only special character permitted is the underscore character "_".
BEGIN · PROGRAM_END	applies as of Version TO01E	MAIN PROGRAM BEGIN-END The declaration part is followed by the start of the main program with instruction BEGIN. It ends with the keyword "PROGRAM_END".
BEGIN HALT PROGRAM_END	applies as of Version TO01E	MAIN PROGRAM-END BY HALT HALT is optional and may occur at several points in the main program. Program execution is stopped (e.g. in the case of decisions IFTHEN).
EXTERNAL: subrout_name subrout_name	applies as of Version TO01E	EXTERNAL SUBROUTINE CALLS The names of the external subroutines must be declared before the keyword "BEGIN".
EXTERNAL: subrout_name (VALUE INTEGER : sp,ze, VALUE REAL : de_z) subrout_name (3,2,-50)	applies as of Version TO01E	CALLING EXTERNAL SUBROUTINES with parameter transfer.
PROGRAM subrout_name (VALUE INTEGER: sp,ze VALUE REAL: de_z)	applies as of Version TO01E	PROGRAM DECLARATION with parameter transfer. Can be called as an external subroutine.

# SUBROUTINE DECLARATIONS

SUBROUTINE subrout_name BEGIN SUB_END	applies as of Version TO01E	SUBROUTINE DECLARATION The subroutine name may have a maximum length of 12 characters. The first character must be a letter. The only special character permitted is the underscore "_".
SUBROUTINE subrout_name (VALUE REAL:aw) BEGIN SUB_END	applies as of Version TO01E	SUBROUTINE DECLARATION with parameter- list. If variable 'aw' is modified in subroutine SUBROUTINE_NAME, variable 'oh' is un- changed after the return (see below).
SUBROUTINE subrout_name (REAL:aw) BEGIN SUB_END	applies as of Version TO01E	SUBROUTINE DECLARATION with parameter- list. If variable 'aw' is changed in subroutine SUBROUTINE_NAME, variable 'oh' has the same value after the return (see below).
subroutine name	applies as of Version TO01E	SUBROUTINE CALL A subroutine call can be performed from the main program. A subroutine, in turn, can call another subroutine.
subroutine name (oh)	applies as of Version TO01E	SUBROUTINE CALL with parameter transfer.
RETURN	applies as of Version TO01E	SUBROUTINE-END This can be used optionally if you wish to quit the subroutine at various points.

# **STANDARD CONSTANTS**

WRITE PHG,CLS	applies as of Version TO02F	CLEAR SCREEN CLS You can clear the PHG display by output of "CLS".
IF VERSION < 2.31 THEN WRITE 'Old compiler version'	applies as of Version TO02F	VERSION You can scan the version number of the com- piler in the BAPS program. The constant is of the type REAL.

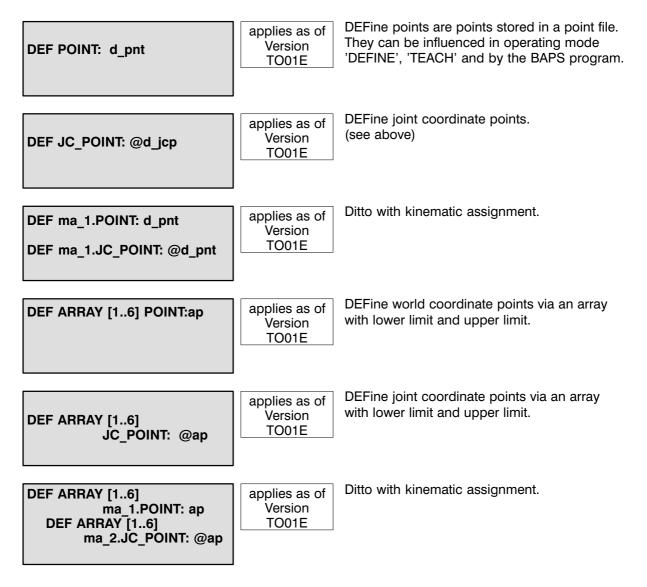


**DATA TYPES** 

# DATA TYPES

POINT: p_name JC_POINT: @p_name	applies as of Version TO01E	Points in world coordinate representation. Points in joint coordinate representation.
ma_1.POINT: kp_name ma_2.JC_POINT: @kp_name	applies as of Version TO01E	Ditto with kinematic assignment.
INTEGER: i_var REAL: r_var	applies as of Version TO01E	Integer values (whole numbers) Decimal (real) floating point
BINARY: b_var	applies as of Version TO01E	Binary variables or I/O signals. There are two states (logical 0 or logical 1).
CHAR: char TEXT: message	applies as of Version TO01E	ASCII characters in accordance with DIN 66003 Character string, e.g. 'SCHOOL'
FILE: danadat	applies as of Version TO01E	This is where the names of the files with exten- sion DAT are declared. Values can be read from or written to DAT files. (See FILES I/O.)
SEMAPHORE: sema_var	applies as of Version TO01E	Variables of type SEMAPHORE are required for the EXCLUSIVE statement.

# POINT DECLARATIONS





### **GLOBAL VARIABLES**

Global variables permit simple data exchange between several independent BAPS2 user programs (processes). The underlying idea is that of combining programs operating with the same global variables to form one program group. One program can export data within this group while the other programs of this group may only import these data. Any number of such program groups can be created on the control (restricted only by the available memory space).

Example program

;Exporting program **PROGRAM exp\_var** 

;global data PUBLIC DEF POINT: start\_pos PUBLIC INTEGER: index PUBLIC SEMAPHORE: writeprotect

**REAL: mvalue** 

#### BEGIN

;BAPS statements EXCLUSIVE writeprotect start\_pos =POS index=5 EXCLUSIVE\_END

;... further statements

PROGRAM END

;Importing program PROGRAM imp\_var

;global data EXTERNAL exp\_var: start\_pos EXTERNAL exp\_var: index EXTERNAL exp\_var: writeprotect

;local data POINT: end\_pos

BEGIN ;BAPS statements EXCLUSIVE writeprotect RPT index TIMES

MOVE TO start\_pos MOVE TO end\_pos RPT\_END EXCLUSIVE\_END

PROGRAM\_END

applies as of Version TO02F

Declaration part in the exporting program:

Global variables are declared by adopting the reserved word PUBLIC in the type declaration of these variables. The data range for these variables is reserved in the IRD file. Teach points (e.g. identified by the keyword DEF) are stored in the PNT file.

Declaration part in the **importing program**:

The variables can be accessed by other BAPS programs if the variables are declared with EX-TERNAL and the name of the exporting program.

#### Restrictions

The following restrictions must be taken into consideration when using global data:

1. The exporting program of a program group must have been compiled before the importing programs of this group (because of the type checking of the compiler), i.e. the user must ensure that the importing programs are 'more recent' than the exporting program. If this is not the case, an error message or a warning is issued at the start of the program.

2. In a program, it is not possible to import and export data simultaneously.

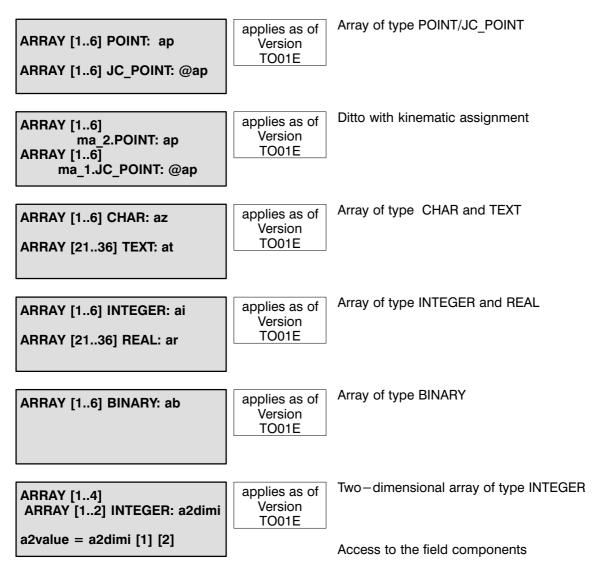
3. Data may be imported only from one program.

4. Global data may be exported or imported only in the declaration part of the main program, and not in the subroutines.

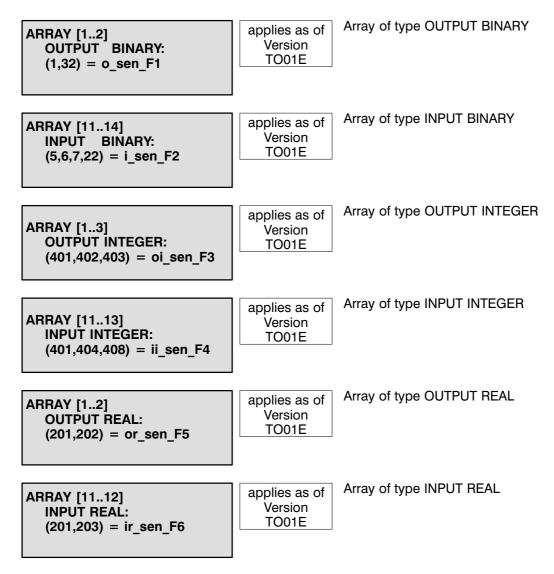
5. Access to the variables must be prevented by use of the EXCLUSIVE statement before interruption in order to guarantee data consistency. The consistency of simple standard types such as BINARY, INTEGER, REAL, CHAR is provided by the operating system.

# **ARRAY DECLARATIONS**

Arrays are data structures consisting of a fixed number of elements of the same type. The lower and upper limits are specified in square brackets. The data type is specified after the square brackets. The array limits are monitored for the following range when declaring arrays of any element type: [-8388608...8388607].



### I/O ARRAY DECLARATIONS



I/O arrays may be declared only in one dimension.

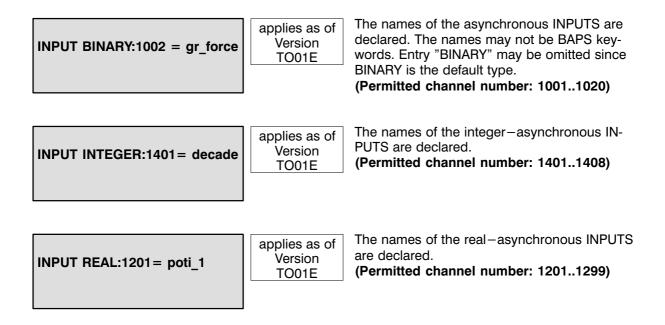
### **CHANNEL DECLARATIONS**

INPUT BINARY: 2 = gr_force OUTPUT BINARY: 9 = magnet	applies as of Version TO01E	The names of the USER INPUTS/OUTPUTS are declared. The names may not be BAPS keywords. Entry " <b>BINARY</b> " may be omitted since BINARY is the default type. (Permitted channel number: 1120, 6xy with high speed inputs)
		Structure of the high speed input number: <b>6 x y</b> Number of the high speed input 0 = Probe input 19 = high speed inputs Number of the Servoboard (13) characteristic for probe and high speed inputs
INPUT INTEGER: 401 = decade OUTPUT INTEGER: 403 = lcd_display	applies as of Version TO01E	The names of the INTEGER USER INPUTS/ OUTPUTS are declared. These are output on the interface with a width of 8 BITS (signifi- cance $0-255$ ). (Permitted channel number: 401408)
INPUT REAL: 201 = poti_1 OUTPUT REAL:203 = v_pres- sure	applies as of Version TO01E	The names of the REAL USER INPUTS/OUT- PUTS are declared. (Permitted channel number: 201299)
BELT: 501 = belt1_ma1 ma_2.BELT: 502 = belt2_ma2	applies as of Version TO01E	The names of the belts are declared. (Permitted channel number: 501508)



#### Asynchronous inputs:

The declaration of asynchronous inputs is made by adding an offset of **1000** to the channel number.



15

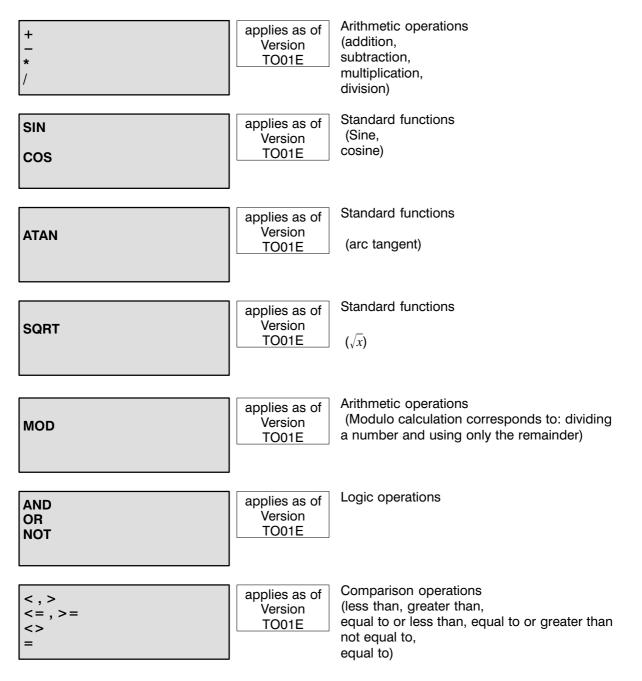
### **PROGRAM FLOW CONTROL**

WAIT 5.5	applies as of Version TO01E	Program execution is stopped for the specified period. The constant (5.5) indicates the time in seconds.
REAL: dwell_time dwell_time = 5.5 WAIT dwell_time	applies as of Version TO01E	A variable may also be used instead of a con- stant. This variable must have been declared beforehand as a REAL or INTEGER variable. This variable must be assigned a value before it is used.
INPUT: 4 = part_there WAIT UNTIL part_there = 1	applies as of Version TO01E	WAIT for a condition to occur. (Only scanning inputs is permitted in the condition.)
INPUT: 5 = sig WAIT UNTIL sig = 0 MAX_TIME = 5.6 ERROR statement1	applies as of Version TO01E	WAIT for a condition to occur with a time limit. If the time has elapsed, the instruction ERROR statement1 is executed. If the condition is ful- filled in the max. time, the error is skipped and the next BAPS instruction is executed.
PAUSE	applies as of Version TO01E	Program execution is stopped. The program is continued with the interfacesignal "RC-START"
REPEAT 5 TIMES REPEAT_END	applies as of Version TO01E	A fixed number of program runs is executed between the two instructions.
INTEGER: number number = 5 REPEAT number TIMES REPEAT_END	applies as of Version TO01E	A fixed number of program runs is executed between the two instructions.
IF condition THEN statement1 ELSE statement2	applies as of Version TO01E	CONDITIONAL STATEMENT If the condition is fulfilled, the control executes statement 1. If the condition is not fulfilled, statement 2 is executed (after ELSE).



IF in1 = 1 THEN BEGIN out36 = 0 out01 = 1 END ELSE BEGIN END	applies as of Version TO01E	BEGIN END compound Several statements can be combined by means of compound statements at points at which only one single statement may occur.
JUMP label_1 label_1:	applies as of Version TO01E	It is possible to jump to labels within main pro- grams or subroutines. Forward and backward jumps are possible.
INTEGER: magnitude,offset BEGIN magnitude=0 offset=0 READ magnitude CASE magnitude EQUAL 9,10 : offset=0 EQUAL 11,12 : offset=1 EQUAL 13 : offset=2 DEFAULT offset=5 CASE_END PROGRAM_END	applies as of Version TO03B	Using the branch statement "CASE", it is possible to implement a selection from several alternatives. Nested "IF – THEN " scans can thus be avoided and the program execution time can be shortened.

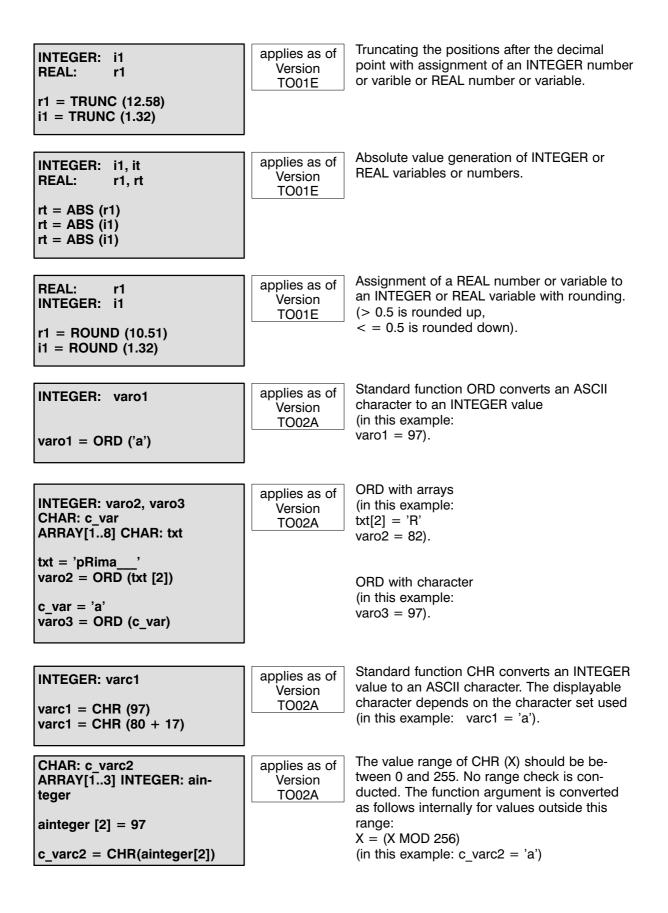
### LOGIC – ARITHMETIC OPERATIONS



# VALUE ASSIGNMENTS

variable = expression	applies as of Version TO01E	An assignment assigns a new value to a vari- able. This value must have a type compatible with the variable type.
p1 = (0,0,-50.123,0) @p1 = @(0,0,-19.4,0)	applies as of Version TO01E	POSITION ASSIGNMENT. Complete value assignment of a world and joint coordinate point. The number of compo- nents is equal to the number of axes of the kinematic.
REAL: z_value z_value = -50.0 p1 = (0,0,z_value,0)	applies as of Version TO01E	Value assignment of a point via variables.
z_axis.Z_C = 31.2 p1 = (0,0,z_axis.Z_C,0)	applies as of Version TO01E	AXIS ASSIGNMENT The axis name is used for the component val- ue assignment.
INTEGER: i i = 0 i = i+1	applies as of Version TO01E	The value assignment can be used for all data types (e.g. REAL, INTEGER, etc.) and for all standard variables (e.g. V_PTP, A, VFACTOR, AFACTOR etc.).
CONST: yellow = 0, white = 1, blue = 4, red = 3 INTEGER : color	applies as of Version TO03B	CONSTANTS. Constants are defined before the variables of a program.

# **STANDARD FUNCTIONS**





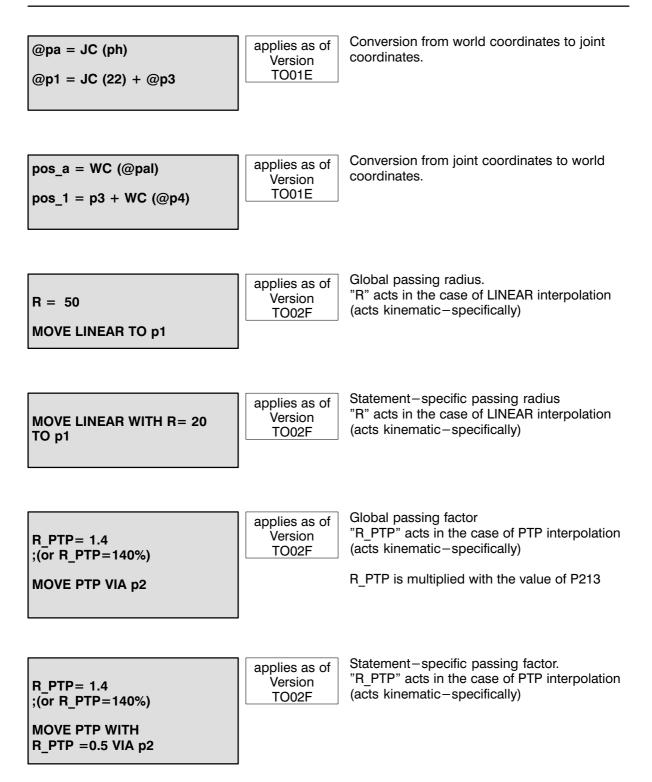
INTEGER: int_number INTEGER: index_ca INTEGER: length_char INTEGER: error_rue ARRAY[110] CHAR: ascii_array	applies as of Version TO02A	an in <u>Note</u> The f blank If an cess, and t	ormat is right justified with leading
int_number = -1234 index_ca = 1 length_char = 5		ascii_	- RESULT ACKNOWLEGMENT array = Result array of Type ARRAY [] char (The destination area is initialized with blanks before conversion)
INT_ASC (int_number, ascii_array, index_ca, length_char, error_rue)		index  lengt	<ul> <li>PRESET</li> <li>ca = Start index in the character array</li> <li>PRESET</li> <li>h_char = Maximum number of charac ters reserved for the number to be converted</li> <li>RESULT ACKNOWLEDGMENT</li> <li>rue = Error number output</li> </ul>
		0	No errors
error_rue <> 0 THEN WRITE 'error in INTEGER-ASCII conver- sion'		-1	Start index outside array limits
ELSE WRITE ascii_array		-2	End index (start index + length) outside array limits
		-3	Reserved length too small
		-4	Range transgression (value too big)
		-5	Array length < 0
		-6	Array length = 0

# **STANDARD FUNCTIONS**

BOSCH Believe Automation

CHAR: ascii_array ascii_array = 'AaBc' index_ca = 1 length_char = 5 ASC_INT (ascii_array, int_number, index_ca, length_char,	applies as of Version TO02A       The standard procedure ASC_INT converts an array of characters to an integer. The procedure reads in characters as of the start position until: <ul> <li>a character which is not a digit is detected</li> <li>the maximum number of characters has been read</li> <li>the end of the character array is reached</li> </ul> <li>The end of the character array is reached</li> <li>The procedure reads in characters has been read</li> <li>the end of the character array is reached</li> <li>The end of the character array of type ARRAY [] char</li> <li>The end end end end end end end end end en</li>		array = array of type ARRAY [] char to be converted - RESULT ACKNOWLEDGMENT er_number = converted number (only type INTEGER permitted) PRESET
error_rue)	Г	0	No errors
IF	-	-1	Start index outside the array limits
error_rue <> 0 THEN WRITE 'error in ASCII-INTEGER conver		-2	End index (start index + length) lies outside the array limits
sion' ELSE		-3	
WRITE int_number		-4	Range transgression (value too high)
		-5	Array length < 0
		-6	Array length = 0
		-7	Character string does not start with a number or sign





If R = 0 or R\_PTP = 0 is programmed, Spatial passing movement is deactivated.

23

# **STANDARD FUNCTIONS**



PROGRAM io_test	applies as of Version	Standard function CONDITION permits: 1a. The status of an interface to be deter-
EXTERNAL: procname INTEGER: index,number, status FILE: fina CONST: textconst = 'I85.7' txbyconst = 'I13' BEGIN number=0 READ V24_1, index IF CONDITION(V24_1) < 0 THEN WRITE 'Read error' status =		mined.V24_1 V24_4,PHG,TTY are supportedFunction value:0no errors-1Interface not found-2Interface blocked-3Timeout-4Parity error-5Overrun error-6Framing error-7Interface error-8Wrong string length-9Protocol error-10Illegal REAL value-11Point not defined1b. The status of a file to be determined.CONDITION (filename) checks, if the specified
CONDITION(fina)		file exists. <u>Function value:</u> 0 specified file does not exist 1 specified file exists
status= CONDITION(procname)	applies as of Version TO04J	<ul> <li>2. To get the status of processes.</li> <li>The name of the process must be declared with EXTERNAL 'processname'.</li> <li>Function value: <ul> <li>-1</li> <li>process does not exist</li> <li>process is holding</li> <li>process is running</li> <li>process is running</li> <li>process is ready</li> </ul> </li> <li>11 1 sub process of main pro- <ul> <li>cess procname is active</li> <li>12 2 subprocesses are active</li> <li>13 3 sub process are active</li> <li>10+n</li> <li>n sub proc. active (n=190)</li> <li>&gt;100</li> <li>process error <ul> <li>error code=funcvalue -100</li> </ul> </li> </ul></li></ul>
status= CONDITION('10.0') status= CONDITION('10') status= CONDITION(textconst) status= CONDITION(txbyconst) PROGRAM_END	applies as of Version TO04J	<ul> <li>3. To get the status of system signals (inputs and outputs). There are two ways possible:</li> <li>1. The address of the signal to be test can be used directly as parameter. Example: 'Ox.y', 'lx.y'</li> <li>2. The address of the signal is a CONST (of type text). Same for an input byte. Function value:</li> <li>0/1 status of a signal "0" or "1"</li> <li>0255 status of a byte</li> <li>-1 invalid signal group</li> <li>-2 invalid bit address</li> <li>-3 invalid character</li> <li>-4 invalid signal address</li> </ul>



PROGRAM progname FILE:filevar TEXT:filename,one_line, channel_var	applies as of Version TO04J	The BAPS command ASSIGN makes it pos- sible to change the names of files or to change the assignment of standard channels (V24_1V24_4, SER_1SER_4, PHG) at the runtime of a BAPS program.
BEGIN WRITE PHG,CLS WRITE PHG,'file name:' READ PHG, filename ASSIGNfilevar,filename		Clears display of PHG Read <i>filename</i> from the PHG Assign the contents of <i>filename to filevar</i> . Fol- lowing file I/O uses the file specified by <i>filevar</i> .
READ_BEGIN filevar READ filevar,one_line WRITE one_line CLOSE filevar		Open file for read Read a line from the file and writes it on the PHG, then close the file.
channel_var = 'V24_2.' ASSIGN PHG, channel_var WRITE PHG, one_line ASSIGN PHG,'PHG.'		<b>note:</b> when ASSIGN is used with files, it is neces- sary to close the files before using <b>ASSIGN</b> with these files
		assigns channel PHG to the content of chan- nel_var (V24_2), writes <i>one_line</i> to channel V24_2 and assigns PHG back to channel PHG
PROGRAM_END		<b>note:</b> it is important to write the character '.' as li- mitation of channel names. If you omit the point '.', assignment is done to a filename (ex- tension DAT), e.g. PHG.DAT

### **MOVEMENT INSTRUCTIONS**

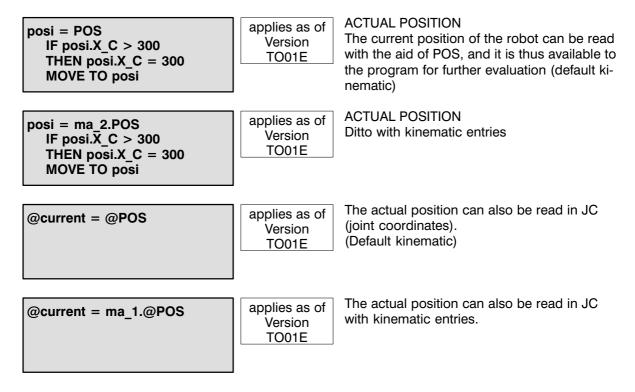
REF_PNT (3) REF_PNT (1,2,3,)	applies as of Version TO01E	REFERENCE POINT TRAVEL Those axes that are to travel simultaneously to the reference point are specified in brackets. This should be the first command used in the INIT program. The default kinematic is moved.
REF_PNT ma_1 (3) REF_PNT ma_2 (1,2,3,)	applies as of Version TO01E	REFERENCE POINT TRAVEL with kinematic entry
MOVE fetchpos MOVE @fetchpos	applies as of Version TO01E	MOVE ABSOLUTE This movement instruction traverses the robot in world or joint coordinates (default kinemat- ic).
MOVE ma_2 fetchpos MOVE ma_1 @fetchpos	applies as of Version TO01E	MOVE ABSOLUTE with kinematic entry.
MOVE fetch + up MOVE @fetch + @up	applies as of Version TO01E	As for Move absolute, but the position is com- puted beforehand and, only then, the system moves.
MOVE ma_2 fetch + up MOVE ma_1 @fetch + @up	applies as of Version TO01E	Ditto with kinematic entry.
MOVE_REL down MOVE_REL @down	applies as of Version TO01E	MOVE INCREMENTAL The robot can be advanced by a distance (in mm) or an angular distance (in degrees) from the current position (default kinematic).
MOVE_REL ma_2 down MOVE_REL ma_2 @down	applies as of Version TO01E	Ditto with kinematic entries
MOVE_REL down + distance_z MOVE_REL @down + @distance_z	applies as of Version TO01E	As for Move incremental, but the position is computed beforehand, and only then, the sys- tem moves. (default kinematic)



MOVE_REL ma_2 down + distance_z MOVE_REL ma_2 @down + @distance_z	applies as of Version TO01E	Ditto with kinematic entries
MOVE TO pos1 MOVE_REL EXACT stre	applies as of Version TO01E	Move to points EXACT (with brief axis stop).
MOVE p1,p2,p3	applies as of Version TO01E	Move EXACT to several points (with brief axis stop).
MOVE VIA @palpos MOVE_REL APPROX home	applies as of Version TO01E	Move to points WITHOUT EXACT HALT.
MOVE UNTIL force = 1 TO hole_5	applies as of Version TO01E	ABORT MOVEMENT Movement occurs in direction hole_5 until hole_5 is reached or until condition (FORCE=1) is fulfilled.
MOVE UNTIL force = 1 ERROR pause TO hole_5	applies as of Version TO01E	ABORT MOVEMENT Movement occurs in direction hole_5 until the condition (FORCE=1) is fulfilled. If the destina- tion position is reached and the condition is not fulfilled, the instruction after ERROR is ex- ecuted
MOVE PTP @pos_door MOVE_REL PTP palpos	applies as of Version TO01E	MOVE POINT-TO-POINT in joint or world coordinates. The control moves synchronously PTP, i.e. all axes start and finish at the programmed point at the same instant.
MOVE LINEAR TO a_z MOVE_REL LINEAR APPROX up	applies as of Version TO01E	LINEAR INTERPOLATION The points are connected by an exact straight line (path), if the working area permits.
MOVE CIRCULAR (hi_put,end_put) MOVE_REL CIRCULAR (@pos1,@pos2)	applies as of Version TO01E	CIRCULAR INTERPOLATION Circular interpolation permits movement in circular arcs. One pair of points is always re- quired so that the control can compute the circular arc size.

BOSCH Flexible Automation

### **ACTUAL POSITION**





# **MEASUREMENT POSITION**

@p1 = @MPOS

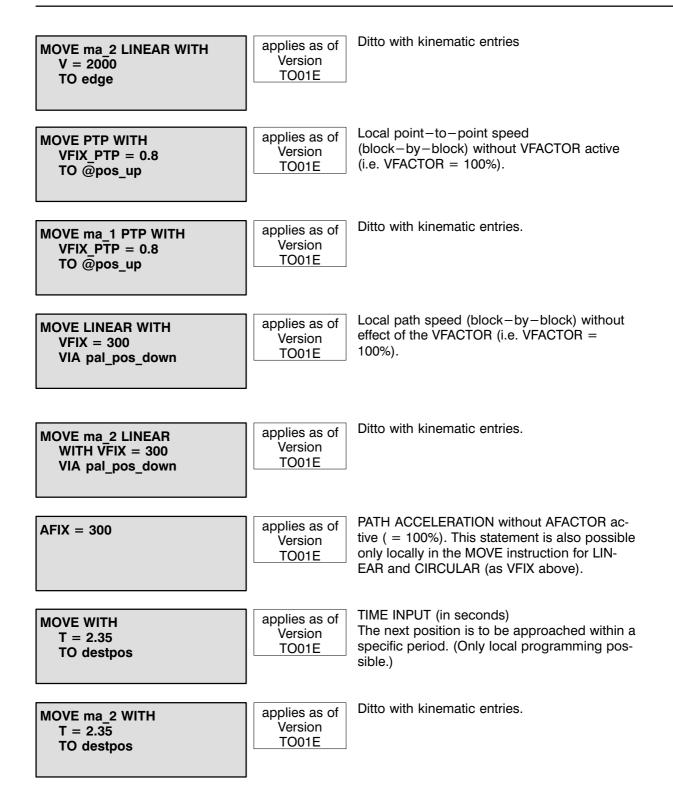
@p1 = ma\_1.@MPOS p1 = WC(ma\_1.@MPOS) applies as of Version TO02A The component @MPOS contains the measurement position which can be determined with the high-speed inputs on the servo card with high accuracy. @MPOS acts only in the joint coordinate system but may be converted with the standard function WC. **SPEEDS** 

# SPEEDS

V_PTP = 80% V_PTP = 0.8	applies as of Version TO01E	POINT-TO-POINT SPEED The global PTP speed is written in one block on its own. It acts until it is overwritten by another speed (acts on the default kinematic).
ma_1.V_PTP = 80% ma_2.V_PTP = 0.8	applies as of Version TO01E	Ditto with kinematic entries. Acts only on the specified kinematic
V = 800	applies as of Version TO01E	PATH SPEED The global path speed acts on linear and cir- cular movements of the default kinematic.
ma_1.V = 800	applies as of Version TO01E	Ditto with kinematic entries
A = 765	applies as of Version TO01E	PATH ACCELERATION The path acceleration acts only on linear and circular movements and not on PTP move- ments.
ma_1.A = 765	applies as of Version TO01E	Ditto with kinematic entries
MOVE PTP WITH V_PTP = 30% TO home	applies as of Version TO01E	LOCAL PTP SPEED The local PTP speed acts only in a MOVE (or Move_REL.) block. After this the global speed becomes active again.
MOVE ma_2 PTP WITH V_PTP = 30% TO home	applies as of Version TO01E	Ditto with kinematic entries
MOVE LINEAR WITH V = 2000 TO edge	applies as of Version TO01E	LOCAL PATH SPEED The local path speed acts only in this block. It must be entered in mm/s.

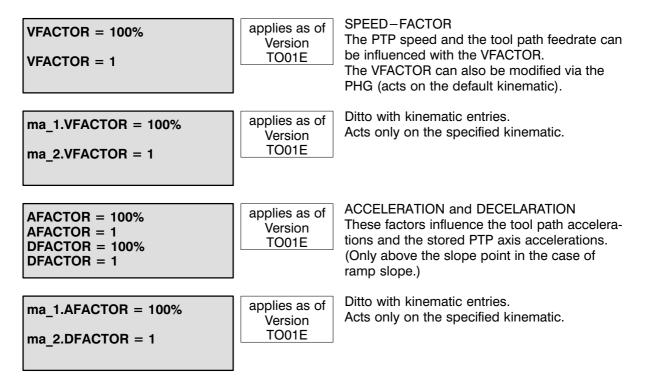


SPEEDS



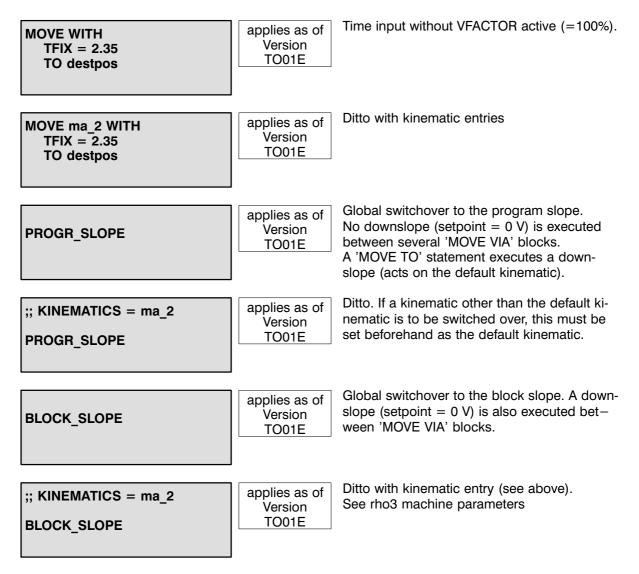


# V / A / D FACTORS





# SLOPE MODES



### **BELT SYNCHRONIZATION**

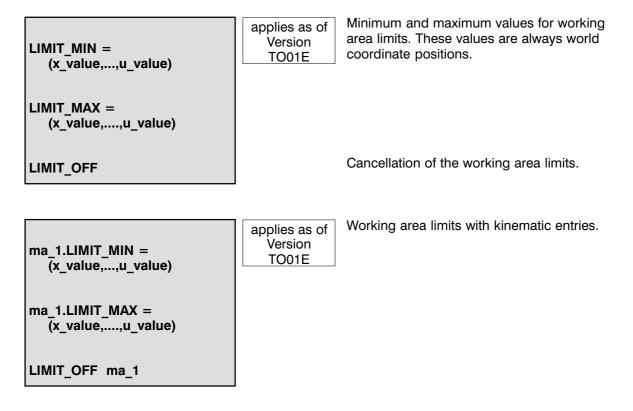
The synchronization statements ensure that the controlled machine assumes the correct position and orientation with respect to the belt. The belt may move forwards and backwards, change its speed or stop as required.

The belt must be a "straight line". This line may be arbitrarily positioned in space.

ma_1.BELT: 501 = belt_kin1	applies as of Version TO01E	DECLARATION OF THE BELT VARIABLES. Several belts can be declared for a kinematic. The belt names must be different. The same belt can be used for several kine- matics.
SYNC belt_kin1 >= 10.0 SYNC BELT, light = 1	applies as of Version TO01E	Belt counter is reset. Reseting may be dependent upon the current value of the belt variables themselves or upon a condition (see SPC_FCT 28).
SYNCHRON ma_1 belt_kin1	applies as of Version TO01E	Instruction SYNCHRON activates belt synchro- nization. From this point on, the programmed movements are synchronized with the belt as regards position and orientation. The kinemat- ic entry is optional.
SYNCHRON_END ma_1 belt_kin1	applies as of Version TO01E	Instruction SYNCHRON_END deactivates belt synchronization. The kinematic entry is option- al; if it is omitted, the default kinematic is used.

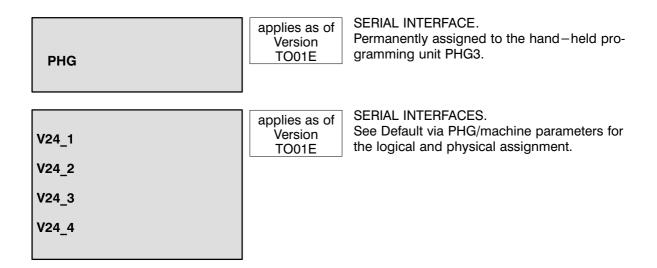


#### WORKING AREA LIMITS



#### WRITE / READ INTERFACES

The WRITE/READ operations in BAPS2 are performed on so-called logical devices to which specific transmission protocols are assigned by default and by machine parameters. Assignment to a physical interface is also performed via machine parameters. This permits flexible use of the interfaces on the control.



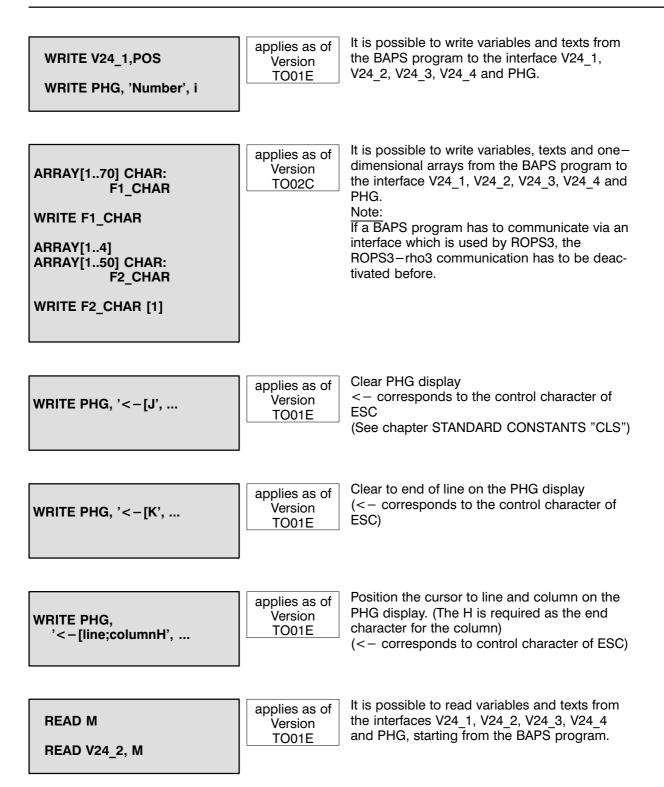
Various communication protocols are available for communication. These can be selected via machine parameter (default) or via mode 9.1 using the PHG.

P No.	Protocol structure	Read Echo
1 a 1 b	<data> followed by <cr><lf> <data> followed by<cr> oder <lf></lf></cr></data></lf></cr></data>	yes
2	<data></data>	yes
3	<soh><stx><data><etx> followed by <soh><stx><cr><lf><etx></etx></lf></cr></stx></soh></etx></data></stx></soh>	no
4	<soh><stx><data><etx></etx></data></stx></soh>	no
5	<data></data>	no
6	PHG – Protocol	yes
7	rho1 / rho2 compatible with P.No. 3	no
8	Data protection layer ot the Siemens-Pro- tocol 3964/R	no

1a = Data - Input

1b = Data-Output



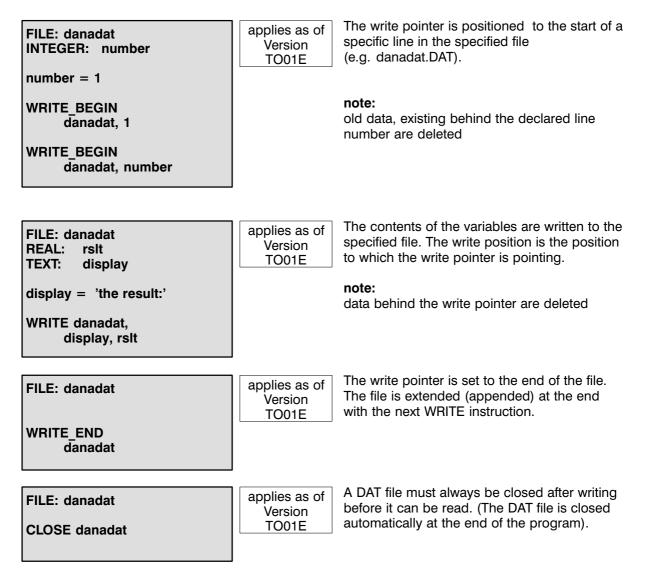


37

### FILES – I/O (READ)

FILE: fina	applies as of Version TO01E	Type declaration for ASCII files which are to permit read and/or write access within a BAPS program. (E.g. in this case, access to FINA.DAT)
FILE: fina READ_BEGIN fina, 5	applies as of Version TO01E	Positioning the READ pointer to the start of a defined line of the given file (in this case, the fifth line in file FINA.DAT). The pointer is positioned to line 1 if the line entry is omitted.
FILE: fina INTEGER: otto READ fina, otto	applies as of Version TO01E	The first value is read from the file (in this case: FINA.DAT) as of the positioned line, and is stored into the variable (e.g. OTTO). The invisible READ pointer is advanced by one position.
FILE: fina IF END_OF_FILE (fina) THEN JUMP finished finished:	applies as of Version TO01E	This function permits interrogation of whether the file end has been reached (e.g. FINA.DAT).
BNR_FILE: fina	applies as of Version TO05G	Type declaration of BINARY-files which are to permit read and/or write access within a BAPS program. (E.g. in this case, access to FINA.BIN)

### FILES – I/O (WRITE)



#### PARALLEL PROCESSES

PARALLEL PROCESSES (EXTERNAL)  EXTERNAL: gripper	applies as of Version TO01E(no synchronized end with par- allel processes)ToonalExternal program with name must be defined	
	in the declaration part.	
START gripper START gripper PRIO=100	The main program (HP) and external program (in this case: GRIPPER) are run in parallel.	
· STOP gripper	A priority scale from 100 (highest) to 150 (low- est) can be assigned.	
•	Parallel program Gripper is stopped and only the main program continues to run.	
PARALLEL PROCESSES (INTERNAL)	(synchronized end in the case of parallel pro- cesses)	
 PARALLEL MOVE pal_pos_1 I = 5	1st process	
ALSO MOVE ma_2 TO pal_corner_I	2nd process	
ALSO	3rd process run simultaneously.	
PARALLEL_END	When all processes are finished, the system continues in the program after PARAL-LEL_END.	
MOVE TO home		
SEMAPHORE:sema_var applies as of Version	If it is intended to access exclusively to com- mon resources (e.g. printers), this can be	
EXCLUSIVE TO01E	achieved with the Exclusive statement. The semaphore variable must have been declared beforehand in the declaration part and acts	
EXCLUSIVE_END	globally in the operating system, i.e. an inter- lock function can be performed by all pro- grams.	
	If another parallel-running process is using the semaphore, the second process remains in "wait" state until the semaphore is released again. EXCLUSIVE – uses the semaphore EXCLUSIVE_END – releases the semaphore If the parallel running process is not using the semaphore, the interlock has no effect.	



#### **SPECIAL FUNCTIONS**

SPC_FCT:	applies as of Version TO01E	Functions to which no BAPS keywords are assigned are declared as SPECIAL FUNC- TIONS. All special functions are listed in order of number below.
1 = eal (VALUE INTEGER: eal_no VALUE INTEGER: kin_no VALUE INTEGER: coordno VALUE REAL: outp_pos VALUE REAL: para_outp VALUE INTEGER: rate time)	applies as of Version TO01E	Exact-position switching of digital signals on the path with rate time. Up to 8 channels can be used for the function.
2 = ppa (VALUE INTEGER: ppa_no VALUE INTEGER: kin_no VALUE INTEGER: coord_no VALUE REAL: outp_pos VALUE REAL: param VALUE INTEGER: rate)	applies as of Version TO01E	Exact-position output of process parameters output with rate time. SPC2 includes 8 functions for output of 8 bit- wide outputs (value range 0255) on the inter- face to the control or any decimal value via a free analog value output on the servo card.
3 = mach_pos (VALUE INTEGER : kin_no kin_name.JC_POINT : @p_name)	applies as of Version TO03D	Special function 3 sets the internal machine position to the values specified in the point variables @p_name.
3 = mach_pos (VALUE INTEGER : kin_no JC_POINT : @p_name)		If no kinematic is entered when declaring the special function, the point relates to the de-fault kinematic.
		The special function must be declared sepa- rately for each kinematic to which it is in- tended to assign machine positions.

#### **IMPORTANT :**

The kinematic selected with the variable **kin\_no** and kinematic defined in the declaration must be the same.

The operating system of the rho3 control does **not** check for correct assignment.

#### CAUTION ! (Special function 3) :

Note that if machine positions are activated due to incorrect programming, the controller will no longer 'know' where the machine actually is.



4= command (VALUE INTEGER: com **TEXT:** src,dest **INTEGER:** status)

= command (VALUE INTEGER: com TEXT: src,dest INTEGER: status)			nction permits to use rho3 sys- n a BAPS2 program. name of special function for use in BAPS selects the system function 1 = COMPILE (QLL-file) 2 = COPY (any file) 3 = DELETE (any file) 4 = START (user process)
		src,dest =	5= STOP (active user pro- cess) parameters for selected sys- tem function
		status +	result of the function as an integer value:
		ror in file name if -3 = file exten -4 = error in C -5 = file not cfile is no $-6 = erro if CC-7 = compiler>0 = numberPILE or e1051: us1057: file1073: pr1079: sum1085: inv1092: se1112: seis$	-
7 = mirroring (VALUE BINARY: X_INV Y_INV Z INV)	applies as of Version TO01E	Axes 1 and 2 a	are mirrored in this example.
irror, (1,1,0)			
OVE @(50,120,-25.21)			
		This appoid fu	notion normite the required
l = belt_mode (VALUE INTEGER:belt_no VALUE INTEGER:mode_belt)	applies as of Version TO03D	mode of belt sy The following s rently impleme mode_belt = 1	nction permits the required ynchronization to be selected. synchronization modes are cur- nted: Belt synchronization with aversing of the kinematic

17 = mirroring

mirror, (1,1,0)

21 = belt\_mode

MOVE @(50,120,-25.21)



Flexible Automation

23 = time_date (INTEGER: hours, minutes, day, month, year)	applies as of Version TO01E	Access to the time and date of the system clock in the rho3.
24 = sys_time (INTEGER:start_time)	applies as of Version TO01E	Access to an internal system counter (real-time counter).
27 = wc_mainarea (INTEGER: kin_no)	applies as of Version TO01E	Coordinate transformation may lead to excess rotation of the world orientation angles in WC. Special function 27 serves to eliminate this excess rotation.
28 = set_belt (VALUE INTEGER: belt_no VALUE REAL: reset_value)	applies as of Version TO03D	Calling special function 28 assigns the value of variable reset_value to the internal belt reset value for the corresponding belt The internal belt counter is set to the reset value with signal RESET BELT COUNTER.
29 = pnt_store_on (VALUE INTEGER: kin_no VALUE INTEGER: ms_no VALUE REAL: distance)		Calling special function 29 activates storing of internal "commanded positions". Stores positions as WC and JC points. kin_no = number of selected kinematic ms_no = number of measuring system (axis, belt or analogous input) which is related to the pa- rameter 'distance' distance = stores actual "commanded position" when distance is reached
		note: Storing of position after every 'distance' will be done on default (option byte $36 = "0"$ , but it's also possible to store positions at every trans- formation clock by setting the option byte $36$ = "1".
30 = pnt_store_off (VALUE INTEGER: kin_no INTEGER: no_points)		Calling special function 30 terminates storing of "commanded positions". The stored values are saved until the next call of special function 29 with the same kine- matic. kin_no = number of selected kinematic no_points = returns the quantity of stored points since call of special function 29.



31 = pnt_st_read (VALUE INTEGER: kin_no VALUE INTEGER: index JC-POINT: @position POINT: position)	Calling special function 31 makes it possible to read the values stored by special function 29 in a ring buffer. kin_no = number of selected kinematic index = ring buffer index of selected value @position = stored position in JC position = stored position in WC <u>note:</u> To get valid positions it's necessary to call up SPC_FCT 29 and 30 before calling SPC_FCT 31. The size of the ring buffer is fixed to 50 points independent of the number of axes.
(VALUE INTEGER: kin no	Calling special function 41 "Asynchronous speed preselection" permits the modification of the Vfactor of a kinematics during move- ment. kin_no = number of selected kinematics poti_value = asynchronous Vfactor
(VALUE INTEGER: kin no	Ties as of persion D05GOpposite to the normal "MOVE UNTIL"-state- ment the movement by this special function is not directly aborted if the condition is true (probe input). The distance in parameter <i>m_distance</i> is moved before the movement is ended.kin_no=number of selected kinematics m_distance =distance =distance to move after active probe inputnote: This modified "MOVE UNTIL"-statement acts only with probe inputs. The base of this inputs is "700".



Flexible Automation

43 = probe_on (VALUE INTEGER: kin_no VALUE INTEGER: flank INPUT: channel_no)	applies as of Version TO05G	Special function 43 permits to store the actual position by a minimum reaction time of 0,01 ms. The measured value is stored in the stan- dard varaiable "MPOS". The parameter <i>flank</i> is used to select the trigger condition (starting the measurement). The <i>channel_no</i> indentifies, if a measurement has occured. kin_no = number of selected kinematics flank = positive flank = 0 negative flank = 1 channel_no = measurement occurs = 1 no measurement = 0 <u>note:</u> The probe input acts only with incremental measuring systems. The base of the probe inputs is "600".
44 = probe_off (VALUE INTEGER: kin_no INPUT: channel_no)	applies as of Version TO05G	Calling special function 44 disables an active special function 43. kin_no = number of selected kinematics channel_no = input number of probe input
45 = move_file (VALUE INTEGER: kin_no BNR_FILE: curve_x VALUE INTEGER: base JC_POINT:@mod_fiag VALUE ARRAY[16] INTEGER: reserve)	applies as of Version TO05G	<pre>Calling SPC_FCT 45 "MOVE_FILE" a curve chart stored in a binary file is moved by read- ing the actual values in this file. The output takes place in the interpolation raster or posi- tioning control raster of the servo board clock. The issued values are position values. Kin_no = number of selected kinematics curve_x = name of the BNR_FILE base = selects the output of the position values.         1 = interpolation raster         2 = positioning control raster         3 = position modulo calculation at the end of the         block. The value stored in machine pa-         rameter P311 is used as modulo value.         3 = positioning system         3 = positioning</pre>



#### 46 = block\_prep (VALUE INTEGER: kin\_no VALUE INTEGER: number)

applies as of Version TO05G Calling special function 46 "reduced block preparation with MOVE-statements" permits to reduce the internal block preparation. The use of this special function is recommended in connection with "MOVE\_FILE" and "Asynchronous Inputs". Especially with applications of asynchronous inputs it could be necessary to reduce the

block preparation to ensure faster recognition of the inputs.

kin\_no = number of selected kinematics

number = This value is an upper limit of prepared blocks of a kinematics. The range of this parameter is between 1..11. Values < 1 are set internal to 1, values > 11 are set to 11.

#### TOOL – DAT

Coordinate transformation for determining the TOOL center point (TCP) is adjusted in two parts in order to permit different grippers/tools to be used during a processing run.

- a) Transformation as far as the flange. The robot-specific data are defined via the machine parameters.
- b) Transformation as of the flange The individual gripper parameters are stored in the file. Three translations (G\_X, G\_Y, G\_Z) and three rotations (G\_D1, G\_D2, G\_D3), each referring to the flange, are used for general definition.

TOOL.DAT						
gripper_l	= 10	2.5	5	1	2	3
gripper_r	= 0	50.0	1.3	5.4	0	0
gripper_1	= -20	0	120	5	0	6
off	= 0	0	0	0	0	0

The order of the individual coordinates (first  $G_X$ , then  $G_Y$ , ... and last,  $G_D3$ ) must always be observed. Zeroes must be set explicitly for values omitted. The gripper name must start in the first column and may have a maximum length of 12 characters.

TOOL ma_1 gripper_I	A gripper is selected in the BAPS program via command TOOL. e.g.: The gripper coordinate system "grip- per_1" from the file is offset.
------------------------	---

TOOL ma_1 off	applies as of Version TO01E	There is no TOOL_END. However, this can obtained by a gripper coordinate system padded with zeroes. (see example).
------------------	-----------------------------------	---

#### **FIXED FILES**

EXPROG.DAT 00 = init 01 = pal1 FF = basicpos	EXTERNAL PROGRAM SELECTION. The interface input numbers are assigned to the program names (IRD) in file EXPROG.DAT.
MZA.DAT 01 = 'slide forward' 99 = 'end mza'	MACHINE STATUS DISPLAY. The interface input numbers are assigned to the texts in file MZA.DAT. These selected texts can be displayed via the PHG in mode 7, 12, 1/2.
TEXTE.DAT 00 = 'text1' FF = 'text256'	SELECTION OF TEXTS VIA THE INTERFACE. The interface input numbers are assigned to the texts in file TEXTE.DAT.
$\begin{tabular}{lllllllllllllllllllllllllllllllllll$	TOOL COORDINATE description The order of the individual coordinates must always be observed (zeros must be set explic- itly for values omitted). The gripper name must start in the first column and may have a maxi- mum length of 12 characters. (See Tool.)

#### **BAPSPIC-SYNTAX**

#### 1. Declaration part

All variables used in a BAPSPIC – program may have a maximum length of 12 characters. The first character must be a letter. As special character only the underline "\_" is permitted. Upper – and lower letters are equivalent.

Compiler instructuions like **CONTROL** and **VERSION** must be declared before **PROGRAM**. Lines containing compiler instructions should have no other characters following the statement.

;; CONTROL = PIC250	COMPILER INSTRUCTION "PIC250". Indicates that the current program is a PIC-pro- gram (file extension "QLS").
;;VERSION = 200	COMPILER INSTRUCTION "VERSION". Indicates which complier version is used compi- ling this PIC-program (from ROPS3-Version W2C the new BAPSPIC-compiler version 2.00 is available). The default setting is compiler version 1.00
PROGRAM pic_io	PROGRAM NAME. The program name may have a maximum length of 8 characters. The name should agree with the file name, otherwise a warning is repor- ted during compilation of this file.
;;INCLUDE pic_sym	COMPILER INSTRUCTION "INCLUDE". Like in QLL-programs there is the possibility to include other files in the program. This allows to create the symbol file in a separate file which can be included in the PIC-program by the <i>INCLUDE</i> statement. From compiler version 2.00 the file extension is of your own choice. If there is no extension indi- cated, the compiler uses "QLS".

49

PROGRAM pic_io	Declaration of IN- and OUTPUTS. All inputs and outputs used in the PIC-program must be declared.
INPUT: 0 = permiss_phg,	The symbol names are separated by a comma. After the last symbol name in a group, the comma is omitted (see example).
767 = user_24_di	
OUTPUT: 16 = emerg_n_rci,	
743 = user_24_do	

PROGRAM pic_io BINARY: 0 = c0_decrem_m,	Declaration of REMANENT and NON-REMA- NENT MARKER. All marker used in a PIC-program must be de- clared. Remanent marker are declared as "BINARY", temporary marker as "TEMP BINARY". The symbol names are separated by a comma.
191 = c31_reset_m	After the last symbol name in a group, the comma is omitted (see example).
TEMP BINARY: marker_1, marker_2, marker_3	

#### 2. Program part

PROGRAM pic_io BEGIN ;instructions PROGRAM_END	INSTRUCTION PART. From compiler version 2.00 the syntactical con- struction of a BAPSPIC – program is adapted to the BAPS2–Syntax. This means that the in- struction part is now separated from the decla- ration part by the keyword "BEGIN". The end of the program is indicated by the key- word "PROGRAM_END".
feede_aa_rci = feeden_aa_di cyclrun_do = proc_act_rco OR permpr_a_rco user_10_do = feedhd_n_di AND contr_aa_di	ASSIGNMENTS. With the "="-character the value of the variable or expression right of the sign of equality is assi- gned to the variable on the left. The keyword "OR" permits a logic OR-combi- nation of several variables ( <i>var1</i> OR <i>var2</i> OR <i>var3</i> ). The keyword 'AND" permits a logic AND-com- bination of several variables ( <i>var1</i> AND <i>var2</i> AND <i>var3</i> ).
IFauto_mn_di	IF-THEN INSTRUCTION. The IF-THEN instruction permits the build of conditional instructions. This means the instruc-

The IF–THEN instruction permits the build of conditional instructions. This means the instruction is executed if the condition is true.

The keyword "NOT" permits the examination of a variable or expression of logic 0.

IF NOT auto_mn_dI THEN
auto_mn_rci = 0 ELSE
user_1_do = 1

THEN

THEN

auto\_mn\_rci = 0

IF NOTuser 1 di

 $user_1_do = 1$ 

IF-THEN-ELSE INSTRUCTION. This instruction permits the execution of several instructions depending on the input condition.

IF NOT auto_mn_di THEN BEGIN
auto mn rci = 0
user_2_do = 1
END
ELSE BEGIN
user_2_do = 1
user 3 do = 0
END

#### COMPOUND INSTRUCTION.

The compound instruction is used to clamp several instructions.

Thereby it is possible to execute several instructions where normally only one instruction is permitted. feede\_aa\_rci = feeeden\_aa\_di AND (marker\_1 OR marker\_2) OR marker\_3 BUILDING COMPONENT EXPRESSIONS. Using the parentheses "(" and ")" permits the construction of one expression of several variables.

IF NOT auto\_mn\_di THEN JUMP automatic

. automatic: IF marker\_3 THEN user\_11\_do = 1 JUMP INSTRUCTION.

The jump instruction permits jumps to several positions in the PIC-program. The instruction consists of a jump instruction (JUMP) and a destination (*mark:*).

#### INDEX

# **Symbols**

@, 19, 26 @MPOS, 29 @POS, 28 +, 18, 26 -, 18 \*, 18 /, 18 =, 18 <, 18 <, 18 <=, 18 <, 18 <, 18 >, 18 >, 18

# Α

A, 30 ABS, 20 AFACTOR, 32 AFIX, 31 ALSO, 40 AND, 18, 51 APPROX, 27 ARRAY, 12, 13, 37 ARRAY BINARY, 12, 13 ARRAY CHAR, 12 ARRAY INPUT INTEGER, 13 ARRAY INPUT REAL, 13 ARRAY INTEGER, 12 ARRAY JC\_POINT, 12 ARRAY OUTPUT, 13 ARRAY OUTPUT INTEGER, 13 ARRAY OUTPUT REAL, 13 ARRAY POINT, 12 ARRAY REAL, 12 ARRAY TEXT, 12

ARRAY,Two-dimensional, 12 ASC\_ARRAY, 22 ASSIGN, 25 Assignment, 51 Asynchronous inputs, 15 ATAN, 18 AXIS ASSIGNMENT, 19

# В

BAPSPIC Compiler instruction CONTROL, 49 INCLUDE, 49 VERSION, 49 BAPSPIC Declaration part, 49 BAPSPIC instruction part, 51 BAPSPIC Program part, 51 BEGIN, 6, 7, 17 BELT, 14, 34 BINARY, 9 BLOCK\_SLOPE, 33 BNR\_FILE, 38

# С

CASE, 17 CASE\_END, 17 CHANNEL NUMBERS, 14 CHAR, 9, 37 CHR, 20 CIRCULAR, 27 CLOSE, 39 CLS, 8 COMPOUND Instruction, 51 CONDITION, 24 CONST, 19 CONTROL, 4 COS, 18

## D

DEBUGINFO, 4 DEF, 10

# INDEX

BOSCH

DEF ARRAY JC\_POINT , 10 DEF ARRAY POINT, 10 DEF JC\_POINT, 10 DEF POINT, 10 DEFAULT, 17 DFACTOR, 32 DRIVE\_TYPE, 5

## Ε

ELSE, 16, 51 END, 17 END\_OF\_FILE, 38 EQUAL, 17 ERROR, 4, 16, 27 EXACT, 27 EXCLUSIVE, 11, 40 EXCLUSIVE\_END, 11 EXPROG.DAT, 48 EXTERNAL, 6, 11, 40

## F

FILE, 9, 38

### Η

HALT, 6

## 

IF, 16, 51 IF NOT, 51 IF THEN ELSE, 16 INCLUDE, 5 INPUT BINARY, 14 INPUT REAL, 14 INPUTER INTEGER, 14 INT, 5 INT\_ASC, 21 INTEGER, 9, 19

### J

JC, 23 JC\_NAMES, 4 JC\_POINT, 9 JUMP, 17, 52

# Κ

KINEMATICS, 4

# L

LIMIT\_MAX, 35 LIMIT\_MIN, 35 LIMIT\_OFF, 35 LINEAR, 27

# Μ

MAX\_TIME, 16 MOD, 18 MOVE, 26 MOVE\_REL, 26 MZA.DAT, 48

### Ν

NOT, 18

## 0

OR, 18, 51 ORD, 20 OUTPUT BINARY, 14 OUTPUT INTEGER, 14 OUTPUT REAL, 14

## Ρ

PARALLEL, 40 PARALLEL\_END, 40 PAUSE, 16 PERMANENT, 4

# BOSCH

PHG, 37 POINT, 9 POS, 28, 37 PRIO, 40 PROCESS\_KIND, 4 PROGR\_SLOPE, 33 PROGRAM, 6 PROGRAM\_END, 6 PTP, 27 PUBLIC, 11

# R

R, 23 R\_PTP, 23 READ, 37 READ\_BEGIN, 38 REAL, 9, 19 REF\_PNT, 26 REPEAT, 16 REPEAT\_END, 16 RETURN, 7 ROUND, 20

# S

SEMAPHORE, 9, 11 SER\_IO\_STOP, 5 SERIAL INTERFACES, 36 SIN, 18 SPC FCT, 41 1 = exact - position switching, 4117 = mirroring, 422 = exact - position parameter output, 4121 = Belt mode selection, 4223 = System time and date, 43 24 = System counter, 4327 = wc mainarea, 4328 = set belt counter, 4329 = store command position "on", 43 3 = set machine position, 4130 = store command position "off", 43 31 = read stored command positions, 44 4 = command, 42

41 = asynchronous speed preselection, 44 42 = asynchronous "MOVE UNTIL", 44 43 = fast measurement on, 45 44 = fast measurement off, 45 45 = MOVE\_FILE, 45 46 = reduced block preparation, 46 SQRT, 18 START, 40 STOP, 40 SUB\_END, 7 SUBROUTINE, 7 SYNC, 34 SYNCHRON, 34 SYNCHRON\_END, 34

## Т

```
T, 31
TEXT, 9
TEXTE.DAT, 48
TFIX, 33
THEN, 16, 51
TIMES, 16
TO, 27
TOOL, 47
TRUNC, 20
```

### U

UNTIL, 27

### V

V, 30
V\_PTP, 30
V24\_1, 37
VALUE, 6
VERSION, 8
VFACTOR, 32
VFIX, 31
VFIX\_PTP, 31
VIA, 27

#### W

WAIT, 16



WAIT UNTIL, 16 WAIT UNTIL MAX\_TIME, 16 WARNING, 4 WC, 23 WC\_NAMES, 4 WITH, 30 WRITE, 37 WRITE\_BEGIN, 39 WRITE\_END, 39

#### 56

# **Bosch-Automationstechnik**

Robert Bosch GmbH Geschäftsbereich Automationstechnik Industriehydraulik Postfach 30 02 40 D-70442 Stuttgart Telefax (07 11) 8 11-18 57

Robert Bosch GmbH Geschäftsbereich Automationstechnik Fahrzeughydraulik Postfach 30 02 40 D-70442 Stuttgart Telefax (07 11) 8 11-17 98

Robert Bosch GmbH Geschäftsbereich Automationstechnik Pneumatik Postfach 30 02 40 D-70442 Stuttgart Telefax (07 11) 8 11-89 17 Robert Bosch GmbH Geschäftsbereich Automationstechnik Montagetechnik Postfach 30 02 07 D-70442 Stuttgart Telefax (07 11) 8 11-77 12

Robert Bosch GmbH Geschäftsbereich Automationstechnik Antriebs- und Steuerungstechnik Postfach 11 62 D-64701 Erbach Telefax (0 60 62) 78-4 28

Robert Bosch GmbH Geschäftsbereich Automationstechnik Schraub- und Einpreßsysteme Postfach 11 61 D-71534 Murrhardt Telefax (0 71 92) 22-1 81

Robert Bosch GmbH Geschäftsbereich Automationstechnik Entgrattechnik Postfach 30 02 07 D-70442 Stuttgart Telefax (07 11) 8 11-34 75

Technische Änderungen vorbehalten

Ihr Ansprechpartner





Robert Bosch GmbH Geschäftsbereich Automationstechnik Antriebs- und Steuerungstechnik Postfach 11 62 D-64701 Erbach Telefax (0 60 62) 78-4 28