1. Description of task

The COPY command is used for inscription of variables.

2. Operating manual

2.1 COPYING VARIABLES

The command belongs to the command group **Var** and can be found in the first menu.

Command syntax:

COPY	Source:	,	Variable:
	<name of a variable>	,	<name of a variable>
	< constant >	,	<name of a variable>

All variable types, even arrays, are allowed for source and destination. For variables with several components - those are the variables of the type **T**ool, **V**ector, **F**rame and **P**osition - source and destination must be of the same type. For the variables of the **I**nteger or **R**eal type a type conversion is executed. This also applies for system variables if they are also of the type integer or real.

The data of the source must be valid, the data of the destination will be validated after copying. For arrays only the array size addressed with the index must be valid, resp. only this array size will be validated.

Examples for correctly programmed COPY-commands:

COPY	2.5	,	R-Var
COPY	8	,	R-Var[Index] (type conversion)
COPY	I-Var	,	I-Var
COPY	I-Var	,	R-Var[Index] (type conversion)
COPY	R-Var[Index]	,	I-Var[Index] (type conversion)
COPY	T-Var	,	T-Var
COPY	V-Var	,	V-Var[Index]
COPY	P-Var[Index]	,	P-Var[Index]
COPY	_I-VAR	,	I-Var (system variable)
COPY	_I-VAR	,	R-Var[Index] (type conversion)
COPY	_I-VAR	,	_I-VAR (system variable)
		,	

Example for incorrectly programmed COPY-commands:

COPY	2.5	,	V-Var
COPY	P-Var	,	T-Var

2.2 COPYING STRUCTURAL ELEMENTS

In the ROBOTstar V it is possible to copy variables with different components (structures) component by component. In the ROBOTstar V the variable types **T**ool, **V**ector and **P**osition are released for structure access. The same syntax is valid as described under item 2.1.

As separator for the structure access there was defined the „ . „ .
The structure element names in the ROBOTstar V are predefined and exact description of those is given below.
There is no distinction made between capitalization.
Copying of individual components with arrays of structure variables is **not yet possible** in the version RSV 2.3 .

Access to a structure element therefore must **always** be as follows:

<name of the variable>.<name of the structure element>

separator

2.2.1 STRUCTURE ACCESS WITH VECTORS

A variable type **Vector** is defined as follows in the ROBOTstar V:

1.	**frame number**
2.	**X-shifting**
3.	**Y-shifting**
4.	**Z-shifting**
5.	**A-orientation offset**
6.	**B-orientation offset**
7.	**C-orientation offset**
8. ... 31.	**articulation angle offsets for axis 1 to axis 24**

From this there are derived the following structure element names for copying in components:

As an example there are copied digital constants into a **Vector variable**:

COPY	Source:0	,	Variable:**V**-Var.**Frame**
COPY	Source:0	,	Variable:**V**-Var.**X**
COPY	Source:0	,	Variable:**V**-Var.**Y**
COPY	Source:0	,	Variable:**V**-Var.**Z**
COPY	Source:0	,	Variable:**V**-Var.**A**
COPY	Source:0	,	Variable:**V**-Var.**B**
COPY	Source:0	,	Variable:**V**-Var.**C**
COPY	Source:0	,	Variable:**V**-Var.**A1**
COPY	Source:0	,	Variable:**V**-Var.**A2**
...			
COPY	Source:0	,	Variable:**V**-Var.**A24**

The following structure copying actions are executed with vectors in the ROBOTstar V:

```
COPY    Source:<real-number>                    ,  Variable:V-Var.<structure element>
COPY    Source:<integer-number>                 ,  Variable:V-Var.<structure element>
COPY    Source:R-Var                            ,  Variable:V-Var.<structure element>
COPY    Source:I-Var                            ,  Variable:V-Var.<structure element>
COPY    Source:V-Var.<structure element>        ,  Variable:I-VAR
COPY    Source:V-Var.<structure element>        ,  Variable:R-Var
COPY    Source:V-Var.<structure element>        ,  Variable:V-Var.<structure element>
```

2.2.2 STRUCTURE ACCESS WITH POSITIONS

A variable type **Position** is defined as follows in the ROBOTstar V:

1.	number of the main axes
2.	number of the additional axes
3.	position type
4.	frame number
5.	X-shifting
6.	Y-shifting
7.	Z-shifting
8.	A-orientation offset
9.	B-orientation offset
10.	C-orientation offset
11. ... 34.	articulation angle offsets for axis 1 to axis 24

The following structure element names are derived from this for copying in components:

As an example number constants are copied into a **Position variable**:

```
COPY    Source:0    ,  Variable:P-Var.AA    /* Additional Axes */
COPY    Source:0    ,  Variable:P-Var.MA    /* Main Axes */
COPY    Source:0    ,  Variable:P-Var.PT    /* PosTyp */
COPY    Source:0    ,  Variable:P-Var.Frame
COPY    Source:0    ,  Variable:P-Var.X
COPY    Source:0    ,  Variable:P-Var.Y
COPY    Source:0    ,  Variable:P-Var.Z
COPY    Source:0    ,  Variable:P-Var.A
COPY    Source:0    ,  Variable:P-Var.B
COPY    Source:0    ,  Variable:P-Var.C
COPY    Source:0    ,  Variable:P-Var.A1
COPY    Source:0    ,  Variable:P-Var.A2
...
COPY    Source:0    ,  Variable:P-Var.A24
```

The following structure copying actions with positions are executed in the ROBOTstar V:

COPY	Source:**<real-number>**	,	Variable:**P-Var.<structure element>**
COPY	Source:**<integer-number>**	,	Variable:**P-Var.<structure element>**
COPY	Source:**R-Var**	,	Variable:**P-Var.<structure element>**
COPY	Source:**I-Var**	,	Variable:**P-Var.<structure element>**
COPY	Source:**P-Var.<structure element>**	,	Variable:**I-Var**
COPY	Source:**P-Var.<structure element>**	,	Variable:**R-Var**
COPY	Source:**P-Var.<structure element>**	,	Variable:**P-Var.<structure element>**

2.2.3 STRUCTURE ACCESS WITH TOOLS

A variable type **Tool** is defined in the ROBOTstar V as follows:

1. **X-shifting**
2. **Y-shifting**
3. **Z-shifting**
4. **rotation around the x-axis of the tool frame**
5. **rotation around the y-axis of the tool frame**
6. **rotation around the z-axis of the tool frame**

The following structure element names are derived from this for copying in components:

As an example number constants are copied into a **Tool variable**:

COPY	Source:0	,	Variable:T-Var.**X**
COPY	Source:0	,	Variable:T-Var.**Y**
COPY	Source:0	,	Variable:T-Var.**Z**
COPY	Source:0	,	Variable:T-Var.**RX**
COPY	Source:0	,	Variable:T-Var.**RY**
COPY	Source:0	,	Variable:T-Var.**RZ**

The following structure copying actions with tools are executed in the ROBOTstar V:

COPY	Source:**<real-number>**	,	Variable:**T-Var.<structure element>**
COPY	Source:**<integer-number>**	,	Variable:**T-Var.<structure element>**
COPY	Source:**R-Var**	,	Variable:**T-Var.<structure element>**
COPY	Source:**I-Var**	,	Variable:**T-Var.<structure element>**
COPY	Source:**T-Var.<structure element>**	,	Variable:**I-Var**
COPY	Source:**T-Var.<structure element>**	,	Variable:**R-Var**
COPY	Source:**T-Var.<structure element>**	,	Variable:**T-Var.<structure element>**

2.2.4 STRUCTURE ACCESS BETWEEN VECTORS AND POSITION VARIABLES

ROBOTstar V proved that it might be useful to exchange data between vector variables and position variables.

The following structure copying actions between vector variables and position variables are supported in the ROBOTstar V:

COPY	Source:**V**-Var.**<structure element>**	,	Variable:**P**-Var.**<structure element>**
COPY	Source:**P**-Var.**<structure element>**	,	Variable:**V**-Var.**<structure element>**

However, copying is only possible among identical "element groups". For this purpose the following element groups have to be observed:

element group „translation“:	structure elements **X, Y and Z**
element group „orientation“:	structure elements **RX, RY and RZ**
element group „art. angle“:	structure elements **A1 to A24**
element group „frame“:	structure element **frame**

It is only possible to exchange elements of **one** element group (e.g. translation) between position variables and vector variables.

Examples:

Possible combinations:

COPY	Source:**P**-Var.**x**	,	Variable:**V**-Var.**z**
COPY	Source:**P**-Var.**Frame**	,	Variable:**V**-Var.**Frame**
COPY	Source:**V**-Var.**A7**	,	Variable:**P**-Var.**A9**

Combinations which are **not** supported:

COPY	Source:**V**-Var.**Frame**	,	Variable:**P**-Var.**A9**
COPY	Source:**V**-Var.**z**	,	Variable:**P**-Var.**rx**

3. Error messages

The following error messages may occur in conjunction with the COPY-command:

Definition not found!

The variable indicated in the COPY-command was not yet defined or the definition of a LOC_CONST resp. LOC_VAR could not be found during nested search of local definitions.

Definition is an array!

This case occurs if one of the operands is a variable without array indication, the variable definition itself, however, is an array.

```
VAR    I_ARRAY[10]
COPY 5 , I_ARRAY
```

Syntax error in the index!

The array indication is incomplete, e.g. one of the parenthesis is missing or exists twice.

Nesting not allowed!

Nested array indications, e.g.

```
COPY 5 , I_ARRAY[I_VAR[5]]
```

are not allowed.

Index variable not found!

This error case might occur, if the index of an array is a variable and this has not yet been defined or the definition of a LOC_CONST resp. LOC_VAR could not be found during nested search of local definitions.

Index variable not initialized!

This error case might occur, if the index of an array is a variable and this has not yet been inscribed.

Index too big or too small!

The index must be greater zero and must not exceed the value indicated in the variable definition.

Definition is no array!

This case occurs, if one of the operands is a variable with array indication, and if the variable definition itself, however, is no array.

```
VAR     I_ARRAY
COPY  5  , I_ARRAY[3]
```

The operand is no variable!

The destination must only be a variable definition (no constant definition).

Wrong operand types!

The types of the Source and the destination do not match, a type conversion is not possible.

Variable not initialized!

The variable definition of the source was not yet inscribed and therefore must not be read.

Reading not allowed!

The system variable must not be read, not even, if it was already inscribed.

Writing not allowed!

The system variable must not be described. In the normal case then only reading is allowed.

Step not defined!

During execution of the command the system software recognized an error in the structure of the user programs.

DOCUMENTATION

REIS GmbH & Co Obernburg

Title: **General Information for Palletizing**

Authors	: Wenzel/Elter/May
Date	: 26. November 1998
Control	: RSV
Software	: 2.3

Release: **1.12.98/Elter**

Description of application task

The palletizing function to a great extent facilitates generation of programs for palletizing tasks.

The RSV offers the possibility to fulfill palletizing tasks via the so called palletizing pattern programs. The number of the palletizing patterns is only limited by the size of the user program memory since they are also stored there.

Description of the palletizing pattern program

A palletizing pattern program has a fixed format containing the initial identification, four definitions for global or local integer variables, three definitions for local vector variables, three definitions for local integer constants, a tool step and four positions.

The contents of the four global variables indicate the current loading state of the pallet.

The local constants determine the partscounts between the central point and the three pallet corner points.

For the partscounts instead of the local constants (step 15 to 17 in the example) also local or global variable definitions may be used. Thus, the pallet pattern can be changed in automatic mode. **The corresponding pallet must be initialized after each manipulation of these partscounts so that the delta vectors contain the correct data.**

Position variables may be used for the central point and the corner points.

Comment lines are allowed in the palletizing pattern program. The variable names are freely selectable.

Example:

```
 1    PAL          <pallet pattern name>
 2    C            GLOBAL VARIABLES
 3    C            ================
 4    VAR          ITOTAL
 5    VAR          ACTVALUE1
 6    VAR          ACTVALUE2
 7    VAR          ACTVALUE3
 8    C            LOCAL VECTOR VARIABLES
 9    C            ======================
10    LOC_VAR      V_ZP_EP1
11    LOC_VAR      V_ZP_EP2
12    LOC_VAR      V_ZP_EP3
13    C            LOCAL CONSTANTS
14    C            ================
15    LOC_CONST    I_PARTS1,<xxx>
16    LOC_CONST    I_PARTS2,<xxx>
17    LOC_CONST    I_PARTS3,<xxx>
18    C            PALLET POSITIONS
19    C            ================
20    TOOL         T
21    POSITION     <central point>
22    POSITION     <corner point 1>
23    POSITION     <corner point 2>
24    POSITION     <corner point 3>
25    END
```

A pallet is defined by 7 parameters:

1.) Central point:
1st position in the palletizing pattern program
Position of the first part deposited on the pallet

2.) Corner point 1:
2nd position in the palletizing pattern program
With corner point 1 the first row of the palletizing pattern is defined. The parts are arranged into the direction of the corner point 1 beginning at the central point.

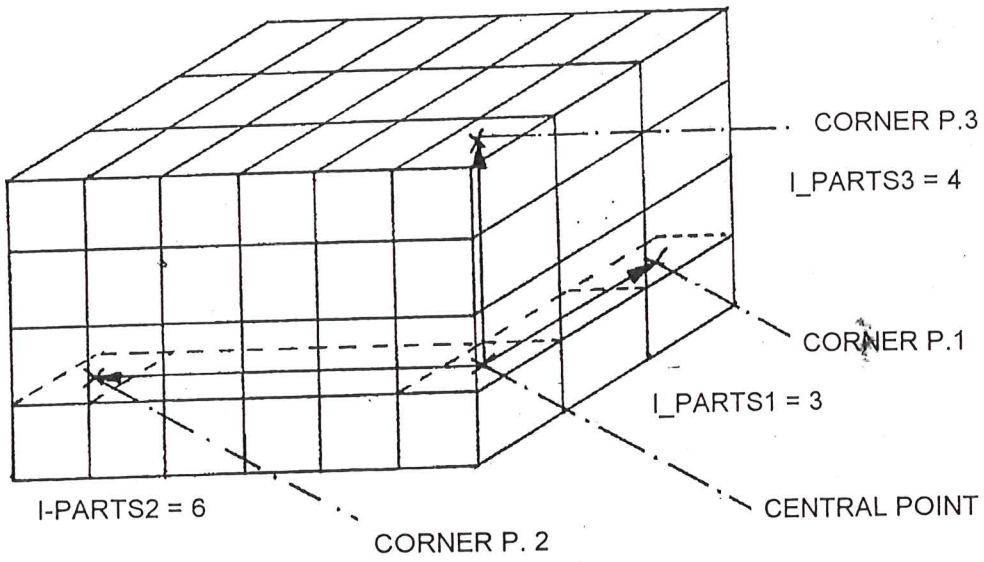
3.) Corner point 2:
3rd position in the palletizing pattern program
With corner point 2 the first layer of the pallet is defined. Further rows are arranged into the direction of the corner point 2 in parallel to the first row.

4.) Corner point 3:
4th position in the palletizing pattern program
This point defines the direction into which the different layers of the pallet are arranged.

5.) Local constant I_PARTS1:
Partscount per row into direction
Central point ---> corner point 1

6.) Local constant I_PARTS2:
Partscount per row into direction
Central point ---> corner point 2

7.) Local constant I_PARTS3:
Partscount per row into direction
Central point ---> corner point 3



Pallet patterns are defined in main programs and therefore can be treated like the remaining main programs.

The command PALLET
The command PALLET is located in the second submenu under the teach pendant key Spec. It has two parameters, the remark concerning the palletizing pattern program name and the command type. The system equates #ON, #OFF and #INIT describe the command type.

With each PALLET command it is checked whether the palletizing pattern program exists and whether it corresponds to the syntax agreed upon.

PALLET <Pal-program>, #INIT
This command initializes the corresponding variables in the palletizing pattern program. The user must ensure that the partscount per palletizing direction (local constants 1...3) and the pallet directions (positions 1...4) will be correctly programmed.

The command **PALLET <Pal-program>, #INIT** calculates the total partscount from the individual partscounts and stores it under the global variable ITOTAL. The global variables ACTVALUE1 ... ACTVALUE3 are allocated with zero and incremented according to the running time.

Delta vectors per palletizing direction (= displacement vector between two parts) are calculated from the corresponding position information and the partscounts and stored in the local vector variables.

The command **PALLET <Pal-program>, #INIT** must be processed at least once before using the command **PALLET <Pal-program>,#ON** since otherwise the variables are invalidated.

PALLET <Pal-program>, #ON
This command calculates from the actual global variables ACTVALUE1...ACTVALUE3 and the local vector variables V_ZP_EP1... V_ZP_EP3 the corresponding palletizing offset vector to the running time.

This palletizing displacement is added to the running time to the following positions. At the same time, the total parts counter ITOTAL is decremented by one. The displacement vector is kept so long until a new command **PALLET <Pal-program>,#ON** will be processed or until it will be deleted again by the command **PALLET <Pal-program>,#OFF**.

PALLET <Pal-program>,#OFF
The pallet displacement is reset again by this command. After #OFF resp. before the next #ON the total parts counter must be tested with the TEST command and with pallet being completely processed a corresponding reaction (pallet change) has to be introduced.

Example programs:

Three-dimensional palletizing:

```
 1 MP                    "PALTEST"
 2 TOOL                  variable:T
 3 MOVE_MODE             #PTP
 4 PTP_SPEED[%]:         80
 5 PTP_ACCEL [%]:        80
 6 PATH_SPEED [mm/s]:    200.0000
 7 PATH_ACCEL [%]:       100
 8 PALLET                Prog_name:"PATTERN1",#INIT
 9 POSITION              <start position>
10 MOVE_MODE             #LINEAR
11 LABEL                 "CYCLE"
12 PALLET                Prog_name:"PATTERN1",#ON
13 POSITION              <over gripping position (central point)>
14 POSITION              <gripping position (central point)>
15 POSITION              <over gripping position (central point)>
16 PALLET                Prog_name:"PATTERN1",#OFF
17 TEST                  #VARIABLE,Op_1:ITOTAL,#>,Op_2:0,label:"CYCLE"
18 STOP
19 END
```

Corresponding palletizing pattern program:

```
 1 MP                "PATTERN1"
 2 C
 3 C Global variables
 4 C ==============
 5 VAR               name:ITOTAL
 6 VAR               name:ACTVALUE1
 7 VAR               name:ACTVALUE2
 8 VAR               name:ACTVALUE3
 9 C
10 C Local vector variables
11 C ==================
12 LOC_VAR           name:V_ZP_EP1
13 LOC_VAR           name:V_ZP_EP2
14 LOC_VAR           name:V_ZP_EP3
15 C
16 C Local constants
17 C ==============
18 LOC_CONST         name:I_PARTS1,value:3
19 LOC_CONST         name:I_PARTS2,value:3
20 LOC_CONST         name:I_PARTS3,value:2
21 C
22 C Pallet positions
23 C ==============
24 TOOL              variable:T
25 POSITION          <central point>
26 POSITION          <corner point1>
27 POSITION          <corner point 2>
28 POSITION          <corner point 3>
29 END
```

Two-dimensional palletizing (single layers):

same as three-dimensional palletizing, but:

in the palletizing pattern program PATTERN1:

```
.
.
20  LOC_CONST       I_PARTS3,0
.
```

One-dimensional palletizing (rows):

same as three-dimensional palletizing, but:

in the palletizing pattern program PATTERN1:

```
.
.
19  LOC_CONST       I_PARTS2,0
20  LOC_CONST       I_PARTS3,0
.
.
```

Miscellaneous:

- Irregular palletizing patterns like:

```
X   X   X   X   X

  X   X   X   X

X   X   X   X   X

  X   X   X   X

X   X   X   X   X
```

require programming of 2 palletizing patterns like:

```
X   X   X   X   X


X   X   X   X   X


X   X   X   X   X
```

and like:

```
X   X   X   X


X   X   X   X
```

- Omitting individual rows:

```
            7   X   X   X   X   X
            6
            5   X   X   X   X   X
ACTVALUE2       4   X   X   X   X   X
  ^         3   X   X   X   X   X
  |         2   X   X   X   X   X
  |         1   X   X   X   X   X
  |         0   X   X   X   X   X
  |
  |             0   1   2   3   4
  | ----------------------> ACTVALUE1
```

By manipulating the parts' counters actual to the running time it is possible to omit entire rows resp. individual parts' positions.

Corresponding main program:

```
 1 MP                    "PALTEST"
 2 TOOL                  variable:T
 3 MOVE_MODE             #PTP
 4 PTP_SPEED[%]:         80
 5 PTP_ACCEL [%]:        80
 6 PATH_SPEED [mm/s]:    200.0000
 7 PATH_ACCEL [%]:       100
 8 PALLET                Prog_name:"PATTERN1",#INIT
 9 POSITION              <Start position>
10 MOVE_MODE             #LINEAR
11 LABEL                 "CYCLE"
12 PALLET                Prog_name:"PATTERN1",#ON
13 POSITION              <over gripping position (central point)>
14 POSITION              <gripping position (central point)>
15 POSITION              <over gripping position (central point)>
16 PALLET                Prog_name:"PATTERN1",#OFF
17 TEST                  #VARIABLE,Op_1:ACTVALUE2,#<>,Op_2:6,label:"CYCLE"
18 ADD                   Op_1:1,Ziel_Var:ACTVALUE2
19 SUB                   Op_1:5,Ziel_Var:ITOTAL
20 TEST                  #VARIABLE,Op_1:ITOTAL,#>,Op_2:0,label:"CYCLE"
21 STOP
22 END
```

Corresponding palletizing pattern program:

```
 1 MP   "PATTERN1"
 2 C
 3 C Global variables
 4 C ==============
 5 VAR                   name:ITOTAL
 6 VAR                   name:ACTVALUE1
 7 VAR                   name:ACTVALUE2
 8 VAR                   name:ACTVALUE3
 9 C
10 C Local vector-variables
11 C ====================
12 LOC_VAR               name:V_ZP_EP1
13 LOC_VAR               name:V_ZP_EP2
14 LOC_VAR               name:V_ZP_EP3
15 C
16 C Local constants
17 C ==============
18 LOC_CONST             name:I_PARTS1,value:5
19 LOC_CONST             name:I_PARTS2,value:8
20 LOC_CONST             name:I_PARTS3,value:0
21 C
22 C Pallet positions
23 C ==============
24 TOOL                  variable:T
25 POSITION              <central point>
26 POSITION              <corner point1>
27 POSITION              <corner point 2>
28 POSITION              <corner point 3>
29 END
```

ATTENTION:

If the actual parts' counters are manipulated, also the total parts' counter must be adapted.

If two pallets are simultaneously required in a user program with the same palletizing pattern (e.g. removal from pallet, treatment and deposit onto pallet), this palletizing pattern must be programmed twice. This is required, because the internal delta vectors have different directions.

EXTENSION OF THE PALLETIZING PATTERN

The palletizing pattern program can contain four further constants. These constants are optional, i.e. they need not be programmed. In this case an integer constant and three real constants are concerned.

If programmed, the integer constant is taken as a control constant. The functionalities are entered in bit code.

Bit 0 = 1 -> The direction vectors defined in the pallet program by the positions are rectangularly set to each other with each command PAL #INIT. In this case, the positions of the corner points 2 and 3 of the pallet program are modified.

Bit 1 = 1 -> Three real constants follow after the integer constant.
These three real constants set the distances between the individual parts' positions into the three dimensions (ZP-EP1, ZP-EP2 and ZP-EP3).
Thus, the programmed corner points of the pallet only set the shifting direction.

Remark:
The pallet pattern is only programmed in the correct manner, if either none of these optional constants or if only the integer constant or if all four constants are programmed.

Applications:
If the user knows that his pattern to be palletized is of rectangular shape, programming errors of the corner points can be compensated by setting the bit 0 in the integer variable. The value of the integer variable then must be '1'.

If the user knows the parts' distance, he can enter it into the three real constants. In this case he must program the integer constant with the value '2' and the three real constants with the corresponding parts' distance in mm. If the integer constant is allocated with the value '3', the pallet points are additionally set to each other in a rectangular way.

If unexpected irregularities occur with these measures, calibration of the robot is not exact enough.

Example for a pallet program with extension:

```
 1 MP                "PATTERN1"
 2 C
 3 C Global variables
 4 C ==============
 5 VAR               name:ITOTAL
 6 VAR               name:ACTVALUE1
 7 VAR               name:ACTVALUE2
 8 VAR               name:ACTVALUE3
 9 C
10 C Local vector variables
11 C ==================
12 LOC_VAR           name:V_ZP_EP1
13 LOC_VAR           name:V_ZP_EP2
14 LOC_VAR           name:V_ZP_EP3
15 C
16 C Local constants
17 C ==============
18 LOC_CONST         name:I_PARTS1,value:<number>
19 LOC_CONST         name:I_PARTS2,value:<number>
20 LOC_CONST         name:I_PARTS3,value:<number>
21 C
22 C Pallet positions
23 C ==============
24 TOOL              variable:<T-variable>
25 POSITION          <central point>
26 POSITION          <corner point1>
27 POSITION          <corner point 2>
28 POSITION          <corner point 3>
29 C
30 C Optional constants
31 C ==================
32 LOC_CONST         name:I_STEUER,value:<control bits>
33 LOC_CONST         name:R_ABST_ZP_EP1,value:<distance>
34 LOC_CONST         name:R_ABST_ZP_EP2,value:<distance>
35 LOC_CONST         name:R_ABST_ZP_EP3,value:<distance>
36 END
```

Structure of the palletizing pattern with position variables

As already mentioned, pallet programs may also contain position variables instead of the programmed positions. The corresponding variable definitions must be in the same pallet program between the local constant definitions and the tool step.

The number of the variable definitions may either only be = 0 (no position variables used) or = 4. Any other number of definitions causes a syntax error.

In a pallet program with position variables each combination from programmed positions and position variables is allowed for the central point and the corner points.

The four definitions of the position variables have the following significance:

Definition 1: central point
Definition 2: corner point 1
Definition 3: corner point 2
Definition 4: corner point 3

The variable names are freely selectable.

When using global definitions, the programmed pallet patterns can be arbitrarily modified to running time.

The position variables can be inscribed with the command ACT_POS.

Example:

```
 1 MP                "PATTERN1"
 2 C
 3 C Global variables
 4 C ==============
 5 VAR               name:ITOTAL
 6 VAR               name:ACTVALUE1
 7 VAR               name:ACTVALUE2
 8 VAR               name:ACTVALUE3
 9 C
10 C Local vector variables
11 C ==================
12 LOC_VAR           name:V_ZP_EP1
13 LOC_VAR           name:V_ZP_EP2
14 LOC_VAR           name:V_ZP_EP3
15 C
16 C Local constants
17 C ================
18 LOC_CONST         name:I_PARTS1,value:<number>
19 LOC_CONST         name:I_PARTS2,value:<number>
20 LOC_CONST         name:I_PARTS3,value:<number>
21 C
22 C Position variables
23 C ==============
24 VAR               name:P_ZP
25 VAR               name:P_EP1
26 VAR               name:P_EP2
27 VAR               name:P_EP3
28 C
29 C Pallet positions
30 C ==============
31 TOOL              variable:<T-variable>
32 VAR_POS           variable:P_ZP
33 POSITION          variable:P_EP1
34 POSITION          variable:P_EP2
35 POSITION          variable:P_EP3
36 END
```

DOCUMENTATION

REIS GMBH & CO MASCHINENFABRIK OBERNBURG

General

Title: Errors and messages of the RSV

Authors:	Laue / Wetzel
Date:	10.11.98
Control version:	RSV
Software:	02_01

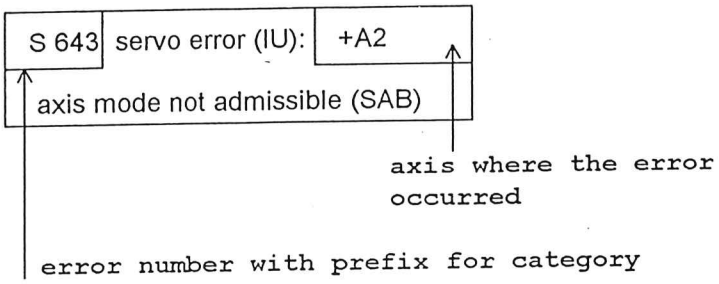
Date: 04.12.98

Release: *Som*

1 Regulator error messages

1.1 STRUCTURE OF THE ERROR MESSAGE

Structure of the regulator error message on the teach pendant:



- Servoregulator error (IU): error of the REIS-Interface-Unit
- Servoregulator error (CU): error of the IRT Control-Unit

The error range 641-700 includes the servoregulator errors, i.e., errors recognized on the axis regulators.
The former message regulator error xx thus is omitted from RSV-software version 02-01 on.

1.2 MEANING OF THE ERROR NUMBERS

Written in italics = message text indicated on the teach pendant

error number	meaning	possible cause
643	*axis mode not admissible (SAB)*	An axis mode not admissible for this axis was requested, e.g. passive switching for an axis which must not be released. What axis modes are the admissible ones results from the machine data IAXES_DESCR.
644	*axis mode not admissible (TMS)*	see error number 643
645	*data change not admitted with drives being active*	Machine data was modified with drives being active the modification of which is only allowed with drives being off.
646	*axis speed too high (FIPO)*	The control gives a speed being too high to this axis. The max. speed increase can be adjusted as factor RSPEED_OVERL in MD_PROG . Too high axis speed in CP_mode
647	*tracking distance too long*	The axis could not follow the given nominal value.
648	*monitoring of standstill*	The axis left the given standstill position.

649	*maximum axis speed reached (FIPO)*	The control gives a speed to this axis that cannot be processed by the servoamplifier.
650	*machine data exceeds value range*	A value was entered in MD_PROG that exceeds the admissible range.
651	*disturbed communication to the ADSP*	There is a communication error to the Control-Unit (CU).
652	*new setup-data (node number/ baudrate) are adjusted*	Setup data of the servo were changed.
653	*over- resp. undervoltage:* over- or undervoltage recognized! occurs in conjunction with error 654	see error 654
654	*error of the power pack:* servoamplifier power pack indicates an error!	1. failure of the voltage supply 2. hardware failure
656	*excess temperature power electronics:* the temperature of the servoamplifier is higher than 80 °C!	1. servoamplifier is overloaded 2. fan defective 3. ambient temperature at the control cabinet is too high
657	*I^2*t monitoring !* *motor overloaded in the cycle*	1. overload of the axis 2. sluggish mechanics 3. current vibration, unfavorable regulator parameters 4. current limit RMS too low
658	*excess temperature motor*	1. overload of the axis (See error 657) 2. cable break 3. current vibration, unfavorable regulator parameters
659	*ADSP has recognized failure of the Reis processor !* (with regard to the software)!	1. no sufficient grounding of the total installation 2. extremely disturbed environment 3. hardware failure of the servoamplifier
660	*ADSP-Firmware wrong:* servoamplifier software is faulty!	data loss in the FLASH of the CU
661	*ADSP-Parameter faulty:* parameters of the servoamplifier are faulty!	data loss in the FLASH of the CU
662	*Resolver signals faulty:* a safe position actual value generation is not possible!	1. resolver defective 2. problem in the resolver cabling
663	*I_max not adjustable:* the max. motor current adjusted in the machine data RCURRENT_MAX cannot be made available by the servoamplifier.	1. machine data was edited 2. machine data was read in from disk - check 3. servoamplifier was exchanged - check performance data of the servoamplifier

664	*I_max <= 0:* Imax of the servoamplifier is <= 0 ! The max. motor current adjusted in the machine data RCURRENT_MAX is negative and thus invalid.	check machine data
665	*I_max not adjustable:* The motor nominal current adjusted in the machine data RCURRENT_RMS cannot be made available by the servoamplifier.	1. machine data was edited 2. Machine data was read in from disk - check 3. servoamplifier was exchanged -Check performance data of the servoamplifier
666	*I_rms <= 0:* Irms of the servoamplifier <= 0! The motor nominal current adjusted in the machine data RCURRENT_RMS is negative and thus invalid.	check machine data
667	*CAN-Watchdog:* A failure of the CAN-bus was recognized!	1. Control was stopped (RESET) 2. No sufficient grounding of the total installation 3. Extremely disturbed environment 4. Wiring problem on the CAN-line 5. CAN-bus not correctly terminated (120 ohm at the beginning and end of the bus)
668	*TMS-Watchdog:* A failure of the Reis signal processor was recognized!	1. No sufficient grounding of the total installation 2. Extremely disturbed environment 3. Hardware failure of the servoamplifier
669	*SAB-Watchdog:* A failure of the micro controller was recognized!	1. No sufficient grounding of the total installation 2. Extremely disturbed environment 3. Hardware failure of the servoamplifier
670	*ADSP-Watchdog:* A failure of the IRT signal processor was recognized!	1. No sufficient grounding of the total installation 2. Extremely disturbed environment 3. Hardware failure of the servoamplifier
671	*ADSP Overspeed unfavorable regulator adjustment:* 120% of the max. admissible motor speed were reached!	Unfavorable adjustment of the current and speed regulator - reduce reinforcements !
672	*Nominal value filter not adjustable:* Nominal value filter is not adjustable and is limited to maximum or minimum !	1. Machine data IFILTER or IMAN_FILTER was edited 2. Machine data was read in from disk 3. Wrong filter adjustment in the user program 4. Interpolation cycle was reduced
673	*Cable break in the motor phase:* One or several of the three phases of the motor supply line is faulty!	1. cable break 2. defective contactors do not provide a connection to the motor

674	*Nominal position left or not reached:*	1. The given nominal position was not kept
675	*error in the filtered nominal value* : The sum of the filtered position nominal value features a deviation from the unfiltered position nominal value!	1. No sufficient grounding of the total installation 2. Extremely disturbed environment hardware failure of the servoamplifier
676	*error in the cycle relation IPO to the position regulator.* Wrong cycle relation interpolation to position regulator (no rest!)!	1. Machine data IIPO_FACT or IPOS_CNTRL_FACT was edited 2. Machine data was read in from disk - check
677	*reference point angle was modified*: the machine data RHOME_POS was modified! Since reference position is changed by this, the system is set asynchronous!	1. machine data was edited 2. data were read in from disk 3. new servoamplifier was installed
678	*Offset angle was modified:* The machine data IENC_OFFS_HOME was modified! Since reference position was changed by this, the system is set asynchronous!	1. machine data was edited 2. data were read in from disk 3. new servoamplifier was installed
679	*Traversing direction was changed:* The machine data IAXES_DIR was modified! Thus, the traversing direction of the corresponding axis is turned. In order to ensure a safe state, the system is set asynchronous!	1. machine data was edited 2. data were read in from disk 3. new servoamplifier was installed
680	*Checksum error back-up data (AC-Fail):* - system is set asynchronous! The data for position actual value generation are backed up in the switched-off state in the flash of the servoamplifier. During new start of the servoamplifier it is checked whether this data is OK.	1. hardware failure in external power pack of the servoamplifiers 2. wiring problem of the AC-fail line (ext. power pack - servoamplifier) 3. hardware failure on servoamplifier
681	*ADSP-revolution counter not zero after reset:* Revolution counter of the IRT position actual values is not zero - system is set asynchronous! After run-up of the system all revolution counters must be set to zero.	1. No sufficient grounding of the total installation 2. Extremely disturbed environment 3. Hardware failure of the servoamplifier
682	*Resolver position out of tolerance:* The system is set asynchronous! The resolver position changed in switched off condition.	1. Brake of the motor defective - axis slips 2. a big static load moves the axis 3. faulty resolver signals - perhaps in conjunction with error 5
683	*Drives were not switched off:* Drives were still active at the time of switch-off (main switch!) - system is set asynchronous!	1. The drives must be switched off when the installation is switched off 2. Power failure during operation

684	*CU-firmware must be up-dated!*	The servo software has recognized an old IU software for which an up-date cannot be made automatically. -> execute IU-update, read in loader version >= 30 and IU-software >= 2104

2 List of the „path errors“ with remedy measures

Here there are mentioned and commented selected error messages „PATH ERRORS ???“. If other messages occur in conjunction with path errors, the manufacturer must be informed accordingly.

No.	Meaning	Reaction
6	**Programming error:** Path execution has no data from the preparation.	Reduce speed or increase distance of the positions.
102/105	**Programming error:** BAHN_RADIUS = BAHN_DISTANZ 0 mm with activated approximation.	Program BAHN_RADIUS unequal 0 .
504	**Error:** algorithm for position reparameterization divergent.	Acknowledge and check for reproducibility. If error is reproducible, slightly change the speed (via override or command BAHN_GESCHW). In any case **inform the manufacturer.**
505	**Error:** algorithm for orientation reparameterization divergent.	Acknowledge and check for reproducibility. If the error occurs with reversion of orientation, set _IORI_TENS = 0. If this is not the case, **inform the manufacturer.**
611	**Error** with calculation of the speed profile for TCP-path.	Avoid a possibly programmed BAHN_ZEIT-command. In this case perhaps the machine data RSYNC_WEIGHT, RSMOOTH_WEIGHT, RSYNC_MIN_PHASE were set in the wrong manner. **Inform the manufacturer.**
615 - 619	**Error** in conjunction with start/stop operation	Acknowledge, move robot away and start anew. Check for repeatability. **Inform manufacturer.**

No.	Meaning	Reaction
620	**Programming error:** Approximation together with a path without distance to be traversed and without orientation readjustment (path zero step) is not admitted.	Switch off approximation at the beginning and end of path zero steps or delete point that was programmed twice.
621	**Programming error:** Approximation together with a path without distance to be traversed is not admitted.	Switch off approximation or avoid points with same TCP.
622	**Programming error:** Programming of a BAHN_ZEIT for current path where only orientation must be traversed is missing.	Program BAHN_ZEIT or avoid points with same TCP.
623	**Programming error:** Approximation to the following path where neither distance must be traversed nor orientation must be readjusted, is not admitted.	Switch off approximation at the beginning and end of this path or delete point that was programmed twice.
624	**Programming error:** Approximation to the following path where only orientation must be readjusted, is not admitted.	Switch off approximation at the beginning and end of the following path.
625	**Programming error:** Programming of a BAHN_ZEIT for following path where only orientation must be traversed, is missing.	Program BAHN_ZEIT or avoid points with the same TCP.
706	**Programming error:** In the time given by the path, additional axes cannot reach their end positions without being overloaded.	Increase duration of movement by a smaller path speed (command BAHN_GESCHW) or by higher BAHN_ZEIT, if it is programmed.
810/811	**Error:** Data of the start-up point do not coincide with the data of the path preparation.	Faulty START/STOP cases: move robot out of dangerous area and force step preselection. **Inform manufacturer.**
903	**Programming error** Weighted orientation for two successive points is not allowed.	COPY command in the destination variable _IORI_WEIGHT or remove _IORI_CIRCHP.