Jetter

DELTA

Controller Card

D-CON3

Description

Edition 1.00

Table of Contents

1 Description

D-CON3 controller cards are axis controller cards for 3D interpolation. The D-CON3 is equipped with 3 * D-SV1 sub-boards.

The axes can be used in the following operating modes:

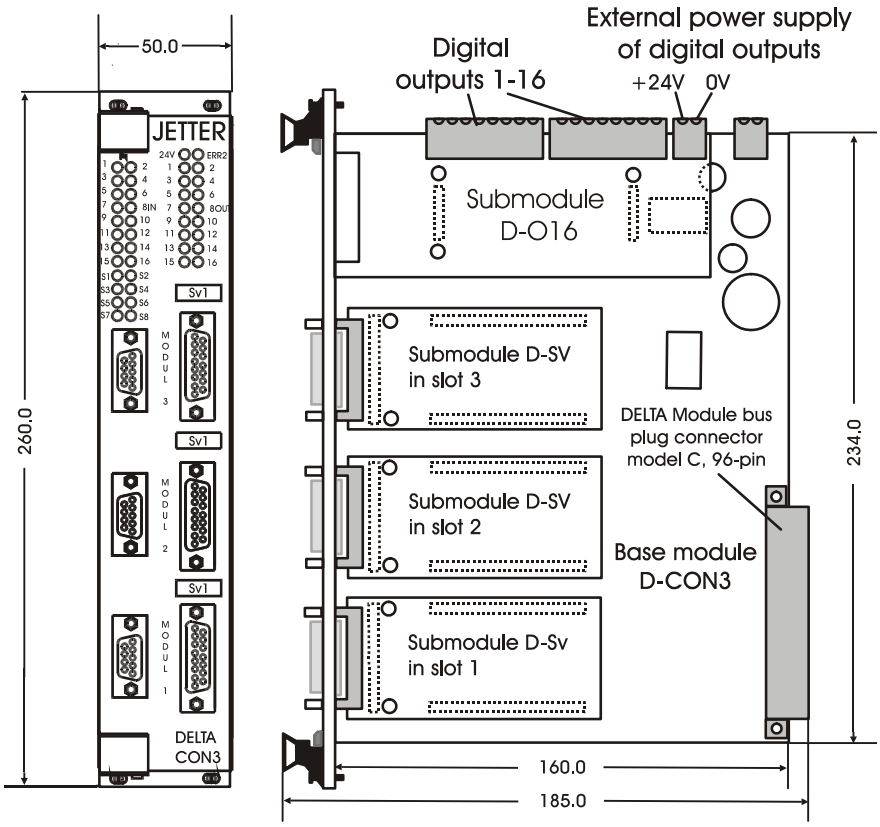
- point to point
- 2D interpolations (only axis 1 with axis 2)
- 3D interpolations

Unlike the axis controller card D-CON+, the following functionalities are not available for D-CON3:

- follower mode with third axis
- cam group
- coordination for backplane bus, follower mode via backplane bus
- processing of analog values

2 Hardware

2.1 Mounting Dimensions



3 Software / Programming

This chapter describes the additional functions of the D-CON3 compared to the axis controller card D-CON+. For single interpolation, the registers of axis 4 are used (1x4zzz). For continuous interpolation, the registers of axis 42 must be used (1x42zz).

3.1 3D Linear Interpolation

3.1.1 Register Assignment

Register 1x4001/1x4201: Instruction Register	
Function	**Description**
Read	Latest instruction
Write	Pass new instruction to controller card
Value range	0 – 255
Value after reset	0

The following instructions are defined for 3D linear interpolation:

24	In the case of continuous interpolation, the current interpolation is added and is started automatically during the process. Register 1x4201 must be used for continuous interpolation.
41	In the case of single interpolation, the current interpolation is started.
42	The interpolation mode of the current axis is quit. For this purpose, clearing the busy bit in the respective status register (bit 13 in register 1xy000) must be waited for.
94	Instruction 94 defines the 3D linear interpolation. This instruction must be given prior to each linear interpolation.

Register 1x4030/1x4230: Set Position of Axis 1	
Function	**Description**
Read	Actual set position of axis 1
Write	Determine new set position of axis 1
Value range	-8388608...+8388607
Value after reset	0

The set position of the first axis is written into this register. For continuous interpolation, register 1x4230 must be used.

Register 1x4031/1x4231: Set Position of Axis 2	
Function	**Description**
Read	Actual set position of axis 2
Write	Determine new set position of axis 2
Value range	-8388608...+8388607
Value after reset	0

The set position of the second axis is written into this register. For continuous interpolation, register 1x4231 must be used.

Register 1x4097/1x4297: Set Position of Axis 3	
Function	**Description**
Read	Actual set position of axis 3
Write	Determine new set position of axis 3
Value range	-8388608...+8388607
Value after reset	0

The set position of the third axis is written into this register. For continuous interpolation, register 1x4297 must be used.

3.1.2 Exemplary Program

```
TASK 0 ------------------------------------------------
    ;
    ; Mode 2 for axes 21, 22, 23
    REGISTER_LOAD [121098 with 2]
    ;
    ; Release all axes
    REGISTER_LOAD [121001 with 1]     ;Release axis 21
    REGISTER_LOAD [122001 with 1]     ;Release axis 22
    REGISTER_LOAD [123001 with 1]     ;Release axis 23
    ;
    REGISTER_LOAD [124001 with 94]    ;3D-Interpolation
    ;
    REGISTER_LOAD [124003 with 600]   ;Speed
    REGISTER_LOAD [124005 with 1000]  ;Acceleration ramp
    REGISTER_LOAD [124006 with 1000]  ;Deceleration ramp
    ; Define positions
    REGISTER_LOAD [124030 with 70000] ;Set position axis 1
    REGISTER_LOAD [124031 with 80000] ;Set position axis 2
    REGISTER_LOAD [124097 with 90000] ;Set position axis 3
    ;
    REGISTER_LOAD [124001 with 41]    ;Start interpolation
    ;
 WHEN
    AXARR 21
    AXARR 22
    AXARR 23
  THEN
    REGISTER_LOAD [121001 with 42]    ;remove axes from
    REGISTER_LOAD [122001 with 42]    ; interpolation
    REGISTER_LOAD [123001 with 42]
WHEN
    BIT_CLEAR [121000, Bit 13]
    BIT_CLEAR [121000, Bit 13]
    BIT_CLEAR [121000, Bit 13]
  THEN
GOTO 0
End of program
```

3.2 3D Circular Interpolation

3D circular interpolation serves to run a 2D circle in a 3D space.

3.2.1 Register Assignment

Register 1x4001/1x4201: Instruction Register	
Function	**Description**
Read	Latest instruction
Write	Pass new instruction to controller card
Value range	0 – 255
Value following reset	0

The following instructions are defined for 3D circular interpolation:

24	In the case of continuous interpolation, the current interpolation is added and is started automatically during the process. Register 1x4201 must be used for continuous interpolation.
41	In the case of single interpolation, the current interpolation is started.
42	The interpolation mode of the current axis is quit. For this purpose, clearing the busy bit in the respective status register (bit 13 in register 1xy000) must be waited for.
107	Activate 3D circular interpolation
108	Deactivate 3D circular interpolation
113	Start calculation for 3D circular interpolation over 3 teach values
115	3D circular interpolation over 3 teach points
116	3D circular interpolation over 3 angles

Register 1x4041/1x4241: Centre Point 1	
Function	**Description**
Read	Actual centre point 1
Write	Centre point 1 for 3D circular interpolation over levels / result of the calculation of 3D circular interpolation over points
Value range	-8388608 .. 8388607
Value after reset	0

3D circular interpolation over levels:	The centre point 1 present on the projected x-y level is written into this register
3D circular interpolation over points:	After completed calculation, the result in register 1x4074 must be copied into this register.

Register 1x4042/1x4242: Centre Point 2	
Function	**Description**
Read	Actual centre point 2
Write	Centre point 2 for 3D circular interpolation over levels / result of the calculation of 3D circular interpolation over points
Value range	-8388608 .. 8388607
Value after reset	0

3D circular interpolation over levels: The centre point 2 present on the projected x-y level is written into this register

3D circular interpolation over points: After completed calculation, the result in register 1x4075 must be copied into this register.

Register 1x4045/1x4245: Set Angle	
Function	**Description**
Read	Actual set angle
Write	Set angle for 3D circular interpolation over levels / result of the calculation of 3D circular interpolation over points
Value range	-8388608 .. 8388607
Value after reset	0

3D circular interpolation over levels: The circular angle present on the projected x-y level is written into this register

3D circular interpolation over points: After completed calculation, the result in register 1x4076 must be copied into this register.

Register 1x4074: Centre Point of Virtual Axis 1	
Function	**Description**
Read	Result of the calculation of 3D circular interpolation over points
Write	Illegal
Value range	-8388608 .. 8388607
Value after reset	0

After completed calculation of the 3D circular interpolation over points, this register must be copied into register 1x4041.

Register 1x4075: Centre Point of Virtual Axis 2	
Function	**Description**
Read	Result of the calculation of the 3D circular interpolation over points
Write	Illegal
Value range	-8388608 .. 8388607
Value after reset	0

After completed calculation of the 3D circular interpolation over points, this register must be copied into register 1x4042.

Register 1x4076: Circular Angle of Virtual Axes	
Function	**Description**
Read	Result of the calculation of the 3D circular interpolation over points
Write	Illegal
Value range	-8388608 .. 8388607
Value after reset	0

After completed calculation of the 3D circular interpolation over points, this register must be copied into register 1x4045.

Register 1x4077: X-Coordinate Teach Point 1	
Function	**Description**
Read	Verify teach point
Write	Write teach point
Value range	-8388608 .. 8388607
Value after reset	0

When programming over points, the x-coordinate of the first teach point must be written into this register.

Register 1x4078: Y-Coordinate Teach Point 1	
Function	**Description**
Read	Verify teach point
Write	Write teach point
Value range	-8388608 .. 8388607
Value after reset	0

When programming over points, the y-coordinate of the first teach point must be written into this register.

Register 1x4079: Z-Coordinate Teach Point 1	
Function	**Description**
Read	Verify teach point
Write	Write teach point
Value range	-8388608 .. 8388607
Value after reset	0

When programming over points, the z-coordinate of the first teach point must be written into this register.

Register 1x4080: X-Coordinate Teach Point 2	
Function	**Description**
Read	Verify teach point
Write	Write teach point
Value range	-8388608 .. 8388607
Value after reset	0

When programming over points, the x-coordinate of the second teach point must be written into this register.

Register 1x4081: Y-Coordinate Teach Point 2	
Function	**Description**
Read	Verify teach point
Write	Write teach point
Value range	-8388608 .. 8388607
Value after reset	0

When programming over points, the y-coordinate of the second teach point must be written into this register.

Register 1x4082: Z-Coordinate Teach Point 2	
Function	**Description**
Read	Verify teach point
Write	Write teach point
Value range	-8388608 .. 8388607
Value after reset	0

When programming over points, the z-coordinate of the second teach point must be written into this register.

Register 1x4083: X-Coordinate Teach Point 3	
Function	**Description**
Read	Verify teach point
Write	Write teach point
Value range	-8388608 .. 8388607
Value after reset	0

When programming over points, the x-coordinate of the third teach point must be written into this register.

Register 1x4084: Y-Coordinate Teach Point 3	
Function	**Description**
Read	Verify teach point
Write	Write teach point
Value range	-8388608 .. 8388607
Value after reset	0

When programming over points, the y-coordinate of the third teach point must be written into this register.

Register 1x4085: Z-Coordinate Teach Point 3	
Function	**Description**
Read	Verify teach point
Write	Write teach point
Value range	-8388608 .. 8388607
Value after reset	0

When programming over points, the z-coordinate of the third teach point must be written into this register.

Register 1x4087/1x4287: Rotation around the X-Axis	
Function	**Description**
Read	Actual rotation around the x-axis
Write	Write new rotation around the x-axis
Value range	-180000 ... +180000
Value after reset	0

When programming over levels, the x-y level is rotated by the three levels of the Cartesian coordinate system to allow a circle in the space. This register indicates the rotation of the x-y levels around the x-axis.

Register 1x4088/1x4288: Rotation around the Y-Axis	
Function	**Description**
Read	Actual rotation around the y-axis
Write	Write new rotation around the y-axis
Value range	-180000 ... +180000
Value after reset	0

When programming over levels, the x-y level is rotated by the three levels of the Cartesian coordinate system to allow a circle in the space. This register indicates the rotation of the x-y levels around the y-axis.

Register 1x4089/1x4289: Rotation around the Z-Axis	
Function	**Description**
Read	Actual rotation around the z-axis
Write	Write new rotation around the z-axis
Value range	-180000 ... +180000
Value after reset	0

When programming over levels, the x-y level is rotated by the three levels of the Cartesian coordinate system to allow a circle in the space. This register indicates the rotation of the x-y levels around the z-axis.

Register 1x4093: Buffer for Circular Interpolation	
Function	**Description**
Read	Actual buffer
Write	Set new buffer
Value range	1 –10
Value after reset	1

The buffer for 3D circular interpolation over points can be selected by means of this register.

Register 1x1166: Error Register Circular Interpolation	
Function	**Description**
Read	Current error condition
Write	Illegal
Value range	23 bits signed integer
Value after reset	0

The individual bits have the following meaning:

0	An error has occurred
1	Teach points are too close to each other. This bit is set when the difference between starting point to centre point, centre point to end point, or end point to starting point is less then 100 increments.
2	All 3 teach points are located on a line. This bit is set when the amount of the cross product is less than 173 inc²
4	Internal memory allocation error

3.2.2 Input by Angles

Circular programming:

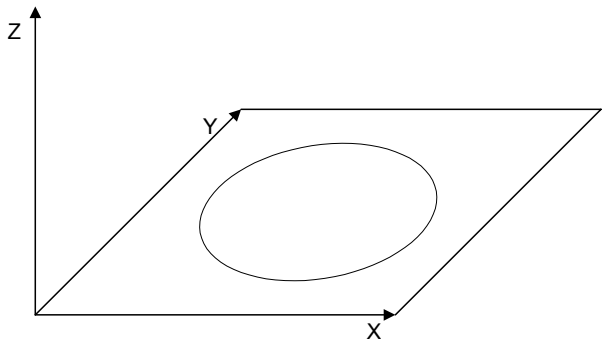
A 3D circle is basically programmed like a 2D circle. The reason is that a 3D circle is derived from a 2D circle by placing the 2D circle into the level of the 3D circle. The centre point (registers 1zy041 and 1zy042) is not entered absolutely but in relation to the level, i.e. only the distance from the starting point to the centre point is entered. This results in the following modifications for programming:

- There are three new registers: 1z4087,1z4088 and 1z4089. These registers indicate the rotation of the x-y level with respect to the respective coordinate axis. This means, 1z4087 indicates the rotation of the x-y level in relation to the x-axis. 1z4088 indicates the rotation of the x-y level in relation to the y-axis, and 1z4089 indicates the rotation of the x-y level in relation to the z-axis. Rotations take place clockwise with the respective axis.

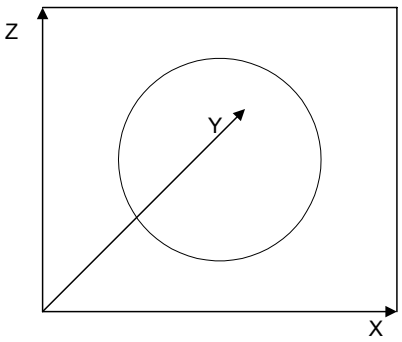
3D circular interpolation over levels can also be carried out continuously.

These three new registers allow to describe any circle in a space. The following applies to special cases of the axis segments:

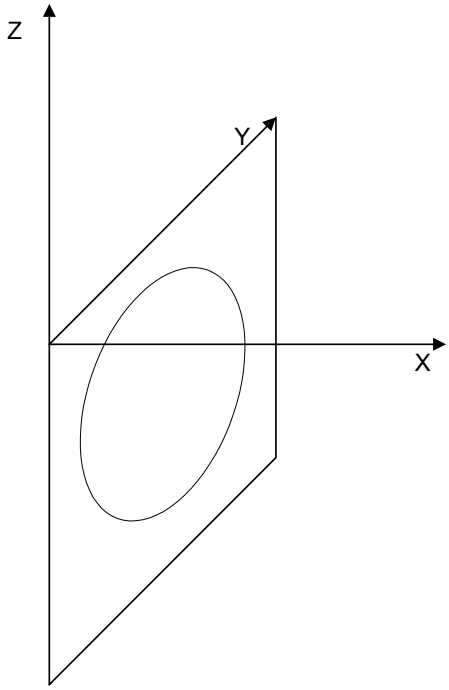
Case 1: Circle in the x-y level.
1z4087=0
1z4088=0
1z4089=0



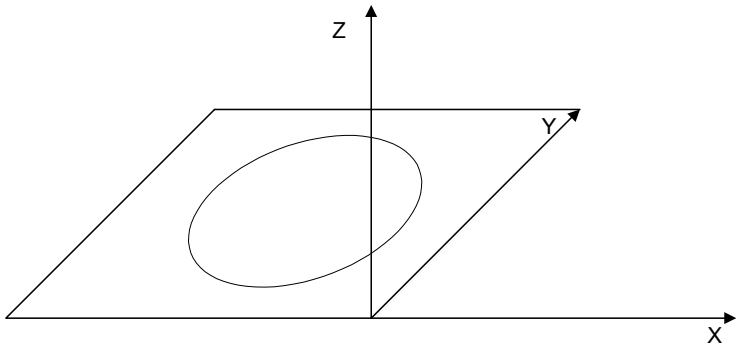
Case 2: Circle in the x-z level. Rotated 90° around the x-axis. 1z4087=90000 corresponds to 90°
1z4088=0
1z4089=0



Case 3: Circle in the y-z level. Rotated 90° around the y-axis.
1z4087=0
1z4088=90000 corresponds to 90°
1z4089=0



Case 4: Circle in the x-y level. Rotated 90° around the z-axis.
1z4087=0
1z4088=0
1z4089=90000 corresponds to 90°



3.2.3 Exemplary Program

The D-CON3 module is plugged in slot 2.

```
TASK 0 ------------------------------------------------------
    ; Mode 2 for axes 21, 22, 23
    REGISTER_LOAD [121098 with 2]
    ;
    ; Referencing and releasing both axes
    REGISTER_LOAD [121001 with 1]      ;Release axis 21
    REGISTER_LOAD [122001 with 1]      ;Release axis 22
    REGISTER_LOAD [121001 with 3]      ;Ref 21 ActPos=0
    REGISTER_LOAD [122001 with 3]      ;Ref 22 ActPos=0
    ;
    ; Declaration of axes
    ; involved in circular interpolation
    REGISTER_LOAD [124128 with 21]     ;Axes 21 and 22 involved in
    REGISTER_LOAD [124129 with 22]     ;circular interpolation


    REGISTER_LOAD [124001 with 107]    ;3-D circle active
    ;
    REGISTER_LOAD [124087 with 90000]  ;Rotating around x-axis
    REGISTER_LOAD [124088 with 90000]  ;Rotating around y-axis
    REGISTER_LOAD [124089 with 0]      ;Rotating around z-axis
    ;
    REGISTER_LOAD [124003 with 600]    ;Speed
    REGISTER_LOAD [124005 with 1000]   ;Acceleration ramp
    REGISTER_LOAD [124006 with 1000]   ;Deceleration ramp
    ;
    ; Define positions
    REGISTER_LOAD [124041 with 40000]  ;Centre point of axis 21
    REGISTER_LOAD [124042 with 35000]  ;Centre point of axis 22
    REGISTER_LOAD [124045 with 16383]  ;Nominal angle 180°
    ;
    REGISTER_LOAD [124001 with 116]    ;3-D circle with angle
    REGISTER_LOAD [124001 with 41]     ;Start interpolation
    ;
 WHEN
      AXARR 21
      AXARR 22
      AXARR 23
  THEN
    REGISTER_LOAD [121001 with 42]     ;End of interpolation
    REGISTER_LOAD [122001 with 42]
    REGISTER_LOAD [123001 with 42]
WHEN
    BIT_CLEAR [121000 Bit 13]          ;Busy bit cleared
    BIT_CLEAR [121000 Bit 13]
    BIT_CLEAR [121000 Bit 13]
  THEN
    GOTO 0
End of program
```

3.2.4 Input by Teach Values

A segment of a circle in a space can also be described by three points as described in 3.2.2. These 9 numerical values are written into the registers 1x4077 through 1x4085. The calculation can then be started with instruction 113. Register 1x4093 is an index register, i.e. max. 10 circles can be
pre-calculated. Therefore, the value range of the index is 1..10.
The centre point and the angle to be covered is then determined by means of these 9 numerical values. Instruction 113 starts the calculation. The initial data are available in the registers 1x4074 through 1x4076. These three initial data must then be copied 1:1 into the registers 1x4041, 1x4042, 1x4045, or registers 1x4241, 1x4242, 1x4245. Instruction 115 starts the motion for the three points.

3.2.5 Exemplary Program

The D-CON3 module is plugged in slot 2.

```
TASK 0 ---------------------------------------------------
    ; Mode 2 for axes 21, 22, 23
    REGISTER_LOAD [121098 with 3]
    ; Referencing and releasing both axes
    REGISTER_LOAD [121001 with 1]      ;Release axis 21
    REGISTER_LOAD [122001 with 1]      ;Release axis 22
    REGISTER_LOAD [121001 with 3]      ;Ref 21 ActPos=0
    REGISTER_LOAD [122001 with 3]      ;Ref 22 ActPos=0
    REGISTER_LOAD [123001 with 3]      ;Ref 23 ActPos=0
    ; Declaration of axes
    ; involved in circular interpolation
    REGISTER_LOAD [124128 with 21]     ;Axes 21 and 22 involved in
    REGISTER_LOAD [124129 with 22]     ;interpolation
    REGISTER_LOAD [124093 with 1]     ;Use buffer 1 to store
                                      ;the circle data.
    REGISTER_LOAD [124001 with 107]   ;3-D circle active
    REGISTER_LOAD [124077 with 0]     ;Point 1 x-axis
    REGISTER_LOAD [124078 with 0]     ;Point 1 y-axis
    REGISTER_LOAD [124079 with 0]     ;Point 1 z-axis
    REGISTER_LOAD [124080 with 8659]  ;Point 2 x-axis
    REGISTER_LOAD [124081 with 10000]  ;Point 2 y-axis
    REGISTER_LOAD [124082 with -5000]  ;Point 2 z-axis
    REGISTER_LOAD [124083 with 0]      ;Point 3 x-axis
    REGISTER_LOAD [124084 with 20000]  ;Point 3 y-axis
    REGISTER_LOAD [124085 with 0]      ;Point 3 z-axis
    REGISTER_LOAD [124001 with 113]    ;Start calculation
WHEN
     BIT_CLEAR[121000 BIT 13]        ;Calculation takes about 20ms
  THEN
    REGISTER_LOAD [124003 with 600]       ;Speed
    REGISTER_LOAD [124005 with 1000]      ;Acceleration ramp
    REGISTER_LOAD [124006 with 1000]      ;Deceleration ramp
    REGISTER_LOAD [124093 with 1]         ;Use buffer 1.
    REGISTER_LOAD [124041 with R(124074)]  ;Centre point axis 1
    REGISTER_LOAD [124042 with R(124075)]  ;Centre point axis 2
    REGISTER_LOAD [124045 with R(124076)]  ;Nominal angle
    REGISTER_LOAD [124001 with 115]       ;3-D with 3 positions
    REGISTER_LOAD [124001 with 41]        ;Start interpolation
```

```
    WHEN
      AXARR 21
      AXARR 22
      AXARR 23
  THEN
    REGISTER_LOAD [121001 with 42]      ;End of interpolation
    REGISTER_LOAD [122001 with 42]
    REGISTER_LOAD [123001 with 42]
WHEN
    BIT_CLEAR [121000 Bit 13]      ;Busy bit cleared
    BIT_CLEAR [121000 Bit 13]
    BIT_CLEAR [121000 Bit 13]
  THEN
     GOTO 0
End of program
```

3.3 3D Spline Programming

Spline programming is used when the axes cover a path that is calculated by means of a mathematical formula. First, the controller card is informed about the teach points to be covered by the axis. In a subsequent calculation, further interpolation points (spline points) are set between these teach points to allow a continuous path. 3D linear interpolation is performed between the individual spline points. The starting direction is defined by the direction from teach point 1 to teach point 2. Of course, the section from teach point 1 to teach point 2 can then be left. The same applies to the direction at the end of the spline movement. The speed can be modified during the spline movement. For a necessary action that has to be started during the spline movement, a counter can be read that indicates the covered teach points.

3.3.1 Register Assignment

Register 1x1001/1x4001: Instruction Register	
Function	**Description**
Read	Latest instruction
Write	Pass new instruction to controller card
Value range	0 – 255
Value after reset	0

The following instructions are defined for the spline movement:

42 The interpolation mode of the current axis is quit. For this purpose, clearing the busy bit in the respective status register (bit 13 in register 1xy000) must be waited for.

94 Interpolation using all three axes on the DIMA4-3 module

109 Starting the spline calculation. After the start, the busy bit must be queried (bit 13 in register 1x1000). Depending on the number of spline points, this calculation lasts between 1 and 10 seconds.

110 This instruction starts the spline movement of the axes. After this instruction, the clearing of the busy bit of the first axis must be waited for.

142 New calculation of the spine length (see chapter 3.3.9). After this instruction, clearing the busy bit of the first axis must be waited for.

Register 1x1187: Spline Number	
Function	**Description**
Read	Read actual spline number
Write	Write new spline number
Value range	1 .. 10
Value after reset	1

The controller can calculate and store up to 10 splines.

Register 1x1188: X-Coordinate Spline Point	
Function	**Description**
Read	Read actual x-coordinate
Write	Write actual x-coordinate
Value range	-8388608 .. 8388607
Value after reset	0

After completed calculation, the x-coordinates of the spline points can be read in this register.

Register 1x1189: Y-Coordinate Spline Point	
Function	**Description**
Read	Read actual y-coordinate
Write	Write actual y-coordinate
Value range	-8388608 .. 8388607
Value after reset	0

After completed calculation, the y-coordinates of the spline points can be read in this register.

Register 1x1190: Z-Coordinate Spline Point	
Function	**Description**
Read	Read actual z-coordinate
Write	Write actual z-coordinate
Value range	-8388608 .. 8388607
Value after reset	0

After completed calculation, the z-coordinates of the spline points can be read in this register.

Register 1x1191: Assigning a Teach Point	
Function	**Description**
Read	Read actual z-coordinate
Write	Write teach point for the spline coordinate
Value range	0 .. 500
Value after reset	0

During the spline run, this register can be queried whether a teach point was reached.

Register 1x1192: Pointer to Teach or Spline Coordinates	
Function	**Description**
Read	Read actual pointer value
Write	Write pointer value
Value range	1 .. 3000
Value after reset	0

Before the calculation, the pointer register accesses a teach coordinate.
After completed calculation, the pointer register accesses a spline coordinate.

Register 1x1193: Number of Calculated Spline Points	
Function	**Description**
Read	Read actual number of spline points
Write	Illegal
Value range	0 .. 3000
Value after reset	0

The number of calculated spline points is written into this register when the spline is calculated. The number of spline points can be decremented artificially to limit the spline.

Register 1x1194: Maximum Speed for Spline Interpolation	
Function	**Description**
Read	Read actual maximum speed
Write	Write new maximum speed
Value range	0 ... Register 1xy021
Value after reset	1000

Before the calculation, the maximum speed reached during the spline run must be written into this register. The number of calculated spline points varies depending on the maximum speed:

- Low maximum speed results in a large number of spline points
- High maximum speed results in a smaller number of spline points

The reason for this interrelation is that two spline points must have a distance from each other of at least 20 ms at maximum speed.

Register 1x1195: Status Register for Spline Calculation	
Function	**Description**
Read	Read actual status
Write	Illegal
Value range	24 bits (bit-coded)
Value after reset	0

- Bit 0: Indicates that an error has occurred
- Bit 2: Internal memory error, insufficient memory.
- Bit 3: Number of max. teach points is reached (e.g. 30)
- Bit 4: Max. number of interpolation points exceeded (e.g. 500).
- Bit 5: Max. number of splines is reached (10).
- Bit 6: Two teach points located one behind the other are identical.

Register 1x1202: Calculation with Correction	
Function	**Description**
Read	Actual correction type
Write	Modify correction type
Value range	0 ... 1
Value after reset	1

Correction for the spline calculation can be enabled or disabled by means of this register. If a length under 10% of 20ms is detected when calculating the individual spline segments, then the spline length is corrected.

Register 1x1204: X-Coordinate Teach Point	
Function	**Description**
Read	Read actual x-coordinate
Write	Write actual x-coordinate
Value range	-8388608 .. 8388607
Value after reset	0

The x-coordinates of the teach points are written into this register before the calculation.

Register 1x1205: Y-Coordinate Teach Point	
Function	**Description**
Read	Read actual y-coordinate
Write	Write actual y-coordinate
Value range	-8388608 .. 8388607
Value after reset	0

The y-coordinates of the teach points are written into this register before the calculation.

Register 1x1206: Z-Coordinate Teach Point	
Function	**Description**
Read	Read actual z-coordinate
Write	Write actual z-coordinate
Value range	-8388608 .. 8388607
Value after reset	0

The z-coordinates of the teach points are written into this register before the calculation.

Register 1x1240: Maximum Number of Teach Points	
Function	**Description**
Read	Memory created for teach points
Write	Determine new memory for teach points
Value range	3 .. 500
Value after reset	30

The number of teach points to be reserved before the spline calculation can be determined by means of this register. The new memory is created when register 1x1240 is written. Therefore, the register must be determined before the teach points are written.

Register 1x1241: Maximum Number of Spline Points	
Function	**Description**
Read	Memory created for interpolation points
Write	Determine new memory for interpolation points
Value range	3 .. 3000
Value after reset	500

The number of teach points to be created for the spline calculation can be determined by means of this register. If more spline points are calculated during the calculation, bit 4 is set in the status register 1x1195.

The maximum number of spline points must be defined prior to the calculation. Each spline may have a varying number of spline points.

Existing old memory is released for the current spline when this register is written. The new memory is created when the spline points are calculated. The memory requirement is approx. 14 * registers 1x1241. Further internal data are required additionally during the calculation. This memory requirement is a little higher than the spline data that will be generated. If the memory is insufficient for the calculation of the spline points, bit 2 is set in register 1x1195. If the calculation of the spline points exceeds the number of spline points mentioned here, then bit 4 is set in register 1x1195.

3.3.2 Process Flow

The spline is programmed as follows:

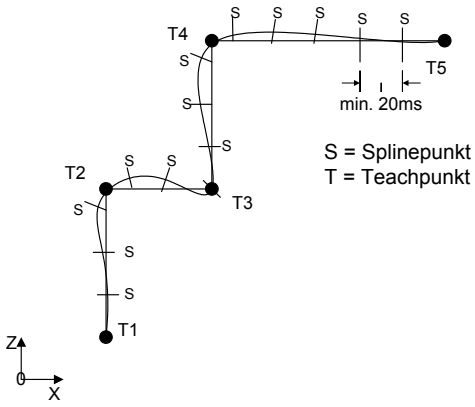
1 Data input
2 Starting the (spline) calculation
3 Starting the motion
4 Switching outputs depending on the teach points
5 Stopping the motion

3.3.3 Data Input

Writing teach points:
First, the spline number must be determined in register 1x1187. Prior to the calculation, the maximum speed must be defined to limit the time for changing the register sets (20ms).
Now the current teach point is written into register 1x1192. Next, the x, y, z coordinates (registers 1x1188 through 1x1190) are written. This is repeated until all teach points are written. Two teach points located one behind the other may not be identical.

3.3.4 Starting the (Spline) Calculation

The spline calculation is started with instruction 109. The busy bit is set during the calculation (bit13 in register 1x1000). After the calculation, the number of points to be covered (calculated spline points) is displayed by register 1x1193. This number results from the initial data set, the maximum speed (reg 1x1194), and an internal limitation of the time needed to change the sets (20ms). Consequently, register 1x1194 must be defined before the calculation is started. For linear interpolation between the individual spline points, a finite small time for the change of sets must be taken into account (e.g. 20ms). Therefore, the distance between the calculated spline points may not become too close. Due to the dependency to the driving speed, a certain speed must already have been defined for pre-interpolation of the splines that will not be exceeded when the path is covered. The lower this maximum speed is set, the more detailed will be the grid of spline interpolation points defined by the spline curve. If the speed of the path is exceeded, a new calculation is necessary with accordingly adapted register 1x1194. The number of calculated spline points is limited in the register 1x1241. If this number is exceeded, bit 4 is set in the register 1x1195.



3.3.5 Starting the Motion

Command 94 starts the initialization of the interpolation mode for all three axes. Spline interpolation is started with instruction 110. If not the entire spline is to be covered, register 193 must first be decreased to the desired number. Instruction 110 always starts at the beginning of the desired spline. Spline interpolations are carried out in axis 42. For this purpose, registers 1x1082 (path segment counter), 1x1083 (filling level) and 1x1084 (wait cycles) can be taken into account.

3.3.6 Switching Outputs Depending on Teach Points

Register 1x1191 can be used to obtain an assignment of teach points and calculated spline points. After having started a spline, register 1x1191 changes from 1 up to the maximum number of teach points. This way, the Sympas program can start any action (e.g. set outputs) when a teach point is reached.

3.3.7 Stopping the Motion

After completed spline, the AXARR bit is set for each axis. In order to quit the interpolation mode, all axes need instruction 42. The spline can also be stopped with instruction 42 for axes 1 through 3.

3.3.8 Exemplary Spline Programming

```
0: TASK 0 ----------------------------------------
1       REGISTER_LOAD [121098 with 2]                ;Mode 2
2:      REGISTER_LOAD [121187 with 1]                ;Spline number 1
3:      REGISTER_LOAD [121240 with 10]               ;Max. 10 teach points
4:   IF
5:      BIT_SET [121195 Bit 2]
6:   THEN
7:       ;error message: insufficient memory
8:   THEN
4:      REGISTER_LOAD [121241 with 600]     ;Max 600 spline points
5:      REGISTER_LOAD [121194 with 500]     ;Max. speed for spline
6:      ; Teach point 1
7:      REGISTER_LOAD [121192 with 1]       ;Teach point number 1
8:      REGISTER_LOAD [121204 with 400]     ;X-position
9:      REGISTER_LOAD [121205 with 2000]    ;Y-position
10:     REGISTER_LOAD [121206 with 0]       ;Z-position
11:     ; Teach point 2
12:       REGISTER_LOAD [121192 with 2]
13:       REGISTER_LOAD [121204 with 400]
14:       REGISTER_LOAD [121205 with 2000]
15:      REGISTER_LOAD [121206 with 200]
16:  ; Teach point 3
17:      REGISTER_LOAD [121192 with 3]
18:      REGISTER_LOAD [121204 with 400]
19:     REGISTER_LOAD [121205 with 2000]
20:     REGISTER_LOAD [121206 with 900]
21:   ; Teach point 4
22:      REGISTER_LOAD [121192 with 4]
23:      REGISTER_LOAD [121204 with 900]
24:       REGISTER_LOAD [121205 with 2000]
25:       REGISTER_LOAD [121206 with 900]
26:       ; Teach point 5
27:       REGISTER_LOAD [121192 with 5]
28:       REGISTER_LOAD [121204 with 900]
29:       REGISTER_LOAD [121205 with 2000]
30:       REGISTER_LOAD [121206 with 1600]
31:       ; Teach point 6
32:       REGISTER_LOAD [121192 with 6]
33:       REGISTER_LOAD [121204 with 2000]
34:       REGISTER_LOAD [121205 with 2000]
35:       REGISTER_LOAD [121206 with 1600]
36:     ; Teach point 7
37:       REGISTER_LOAD [121192 with 7]
38:       REGISTER_LOAD [121204 with 2200]
39:       REGISTER_LOAD [121205 with 2000]
40:       REGISTER_LOAD [121206 with 1600]
41:       ;
42:       REGISTER_LOAD [121001 with 109]   ;Start calculation
43:    WHEN
44:       BIT_CLEAR [REG=121000, Bit=13]    ;Calculation completed
45:    THEN
46:    IF
47:       REG 121195                        ;Reg 11195 # 0 ?
```

```
48:   THEN
49:     ; Error messages ....
50:   THEN
51:     REGISTER_LOAD [124001 with 94]     ;Interpolation 3 axes
52:     REGISTER_LOAD [124003 with 500]    ;Speed
53:                                        ;(max. reg 1x1194)
54:     REGISTER_LOAD [124005 with 500]    ;Acceleration ramp
55:     REGISTER_LOAD [124006 with 500]    ;Deceleration ramp
56:     ;
57:     REGISTER_LOAD [121187 with 1]      ;Spline number 1
58:     REGISTER_LOAD [121001 with 110]    ;Starting interpolation
59:  WHEN
60:     BIT_CLEAR [REG=121000, Bit=13]     ;Spline started ?
61:   THEN
62:     ;
63:  WHEN                                  ;Set output at
64:     REG 121191                         ;teach point 4
65:     >
66:     3
67:   THEN
68:     A 101
69:  WHEN                                  ;Set output at
70:     REG 121191                         ;teach point 6
71:     >
72:     5
73:   THEN
74:     A 102
75:  WHEN                                  ;End of spline interpolation
76:     AXARR axis-=21
77:     AXARR axis=22
78:     AXARR axis=23
79:   THEN
80:     REGISTER_LOAD [121001 with 42]
81:     REGISTER_LOAD [122001 with 42]
82:     REGISTER_LOAD [123001 with 42]
83: WHEN
84:     BIT_CLEAR [121000 Bit 13]     ;Busy bit cleared?
85      BIT_CLEAR [122000 BIT 13]
86:     BIT_CLEAR [123000 BIT 13]
87:   THEN
88:     GOTO 0
```

3.3.9 Modifying / Adding Spline Data

From software 2.800 onwards, it is possible to modify calculated splines. This is necessary e.g. when a rounded square is to be covered. For this purpose, the “round corners” must be calculated as spline; the line inbetween is written as spline segment. The number of spline points must be written into the register 1x1193. After completed spline modification, the overall length of the spline must be calculated again with instruction 142.

Exemplary program:

```
      ; ************************************
      ; * Subprogram adds spline 2          *
      ; * to spline 1.                      *
      ; ************************************
DEF_FUNCTION [AddSpline, AS]
   Par: Spline1, Spline2
   Var: rzSpline1Point, rzSpline2Point, rSpline2no,
        rSplineX, rSplineY, rSplineZ, rSplineTeach
LABEL 50
      REGISTER_LOAD [121187 with R(Spline1)] ;Spline 1
      REG rzSpline1Point                    ;Spline-point pointer1
      =
      REG 121193
      +
      1
      REGISTER_LOAD [121187 with R(Spline2)] ;Spline 2
      REGISTER_LOAD [rzSpline2Point with 1]  ;Spline-point pointer 2
      REGISTER_LOAD [rSpline2no with R(121193)] ;Spline point 2
LABEL  sAdd
      ; Read spline point from spline 2
      REGISTER_LOAD [121187 with R(Spline2)]  ;Spline 2
      REGISTER_LOAD [121192 with R(rzSpline2Point)]
      REGISTER_LOAD [rSplineX with R(121188)] ;Spline point X
      REGISTER_LOAD [rSplineY with R(121189)] ;Spline point Y
      REGISTER_LOAD [rSplineZ with R(121190)] ;Spline point Z
      REGISTER_LOAD [rSplineTeach with R(121191)] ;Teach point
                                                   number
      ; Add spline point to spline 1
      REGISTER_LOAD [121187 with R(Spline1)]  ;Spline 1
      REGISTER_LOAD [121192 with R(rzSpline1Point)]
      REGISTER_LOAD [121188 with R(rSplineX)] ;Spline point X
      REGISTER_LOAD [121189 with R(rSplineY)] ;Spline point Y
      REGISTER_LOAD [121190 with R(rSplineZ)] ;Spline point Z
      REGISTER_LOAD [121191 with R(rSplineTeach)] ;Teach point
  IF                                           ; number
      REG rzSpline2Point                    ;Spline-point pointer 2
      #
      REG rSpline2no                        ;Spline points spline2
    THEN
      REGINC rzSpline1Point
      REGINC rzSpline2Point
      GOTO sAdd
    THEN
      REGISTER_LOAD [121187 with R(Spline1)]
      ; Write number of spline points again
      REGISTER_LOAD [121193 with R(rzSpline1Point)]
      REGISTER_LOAD [121001 with 142]         ;calculate length
  WHEN
      BIT_CLEAR [Reg=121000, Bit=13]         ;Busy bit
```

```
    THEN
      RETURN
END_DEF
      ;
TASK 0
      REGISTER_LOAD [121098 with 3]
      REGISTER_LOAD [121007 with 10]
      REGISTER_LOAD [122007 with 10]
      REGISTER_LOAD [123007 with 10]
      ;
      AXARR axis=21
      AXARR axis=22
      AXARR axis=23
      REGISTER_LOAD [121001 with 1]
      REGISTER_LOAD [122001 with 1]
      REGISTER_LOAD [123001 with 1]
      REGISTER_LOAD [121001 with 42]
      REGISTER_LOAD [122001 with 42]
      REGISTER_LOAD [123001 with 42]
  WHEN
      BIT_CLEAR [Reg=121000, Bit=13]
      BIT_CLEAR [Reg=122000, Bit=13]
      BIT_CLEAR [Reg=123000, Bit=13]
    THEN
      REGISTER_LOAD [121187 with 1]         ;Select spline 1
      REGISTER_LOAD [121241 with 500]       ;max 500 spline points
      ; Teach point 1
      REGISTER_LOAD [121192 with 1]         ;calculate
      REGISTER_LOAD [121204 with 38640]     ;first rounding
      REGISTER_LOAD [121205 with 16000]
      REGISTER_LOAD [121206 with 0]
      ; Teach point 2
      REGISTER_LOAD [121192 with 2]
      REGISTER_LOAD [121204 with 32810]
      REGISTER_LOAD [121205 with 20110]
      REGISTER_LOAD [121206 with 0]
      ; Teach point 3
      REGISTER_LOAD [121192 with 3]
      REGISTER_LOAD [121204 with 29630]
      REGISTER_LOAD [121205 with 35720]
      REGISTER_LOAD [121206 with 0]
      ; Teach point 4
      REGISTER_LOAD [121192 with 4]
      REGISTER_LOAD [121204 with 33340]
      REGISTER_LOAD [121205 with 43390]
      REGISTER_LOAD [121206 with 0]
      ;
      REGISTER_LOAD [121194 with 200]       ;calculation speed
      REGISTER_LOAD [121001 with 109]       ;calculate spline
      ; results in 116 spline points
  WHEN
      BIT_CLEAR [Reg=121000, Bit=13]
      REGZERO 121195
    THEN
      ;
      REGISTER_LOAD [121187 with 2]         ;Select spline 2
      REGISTER_LOAD [121241 with 200]       ;max 200 spline points
      ; Teach point 1
      REGISTER_LOAD [121192 with 1]         ;calculate
      REGISTER_LOAD [121204 with 73290]     ;second rounding
      REGISTER_LOAD [121205 with 105040]
      REGISTER_LOAD [121206 with 0]
      ; Teach point 2
```

```
        REGISTER_LOAD [121192 with 2]
        REGISTER_LOAD [121204 with 76990]
        REGISTER_LOAD [121205 with 110330]
        REGISTER_LOAD [121206 with 0]
        ; Teach point 3
        REGISTER_LOAD [121192 with 3]
        REGISTER_LOAD [121204 with 94990]
        REGISTER_LOAD [121205 with 113770]
        REGISTER_LOAD [121206 with 0]
        ; Teach point 4
        REGISTER_LOAD [121192 with 4]
        REGISTER_LOAD [121204 with 102660]
        REGISTER_LOAD [121205 with 108210]
        REGISTER_LOAD [121206 with 0]
        ;
        REGISTER_LOAD [121194 with 200]      ;calculation speed
        REGISTER_LOAD [121001 with 109]       ;calculate spline
        ; results in 126 spine points
    WHEN
        BIT_CLEAR [Reg=121000, Bit=13]
        REGZERO 121195
      THEN
        SplineAdd [Spline1=1, Spline2=2]
  LABEL 40
        -FLAG 100
    WHEN
        FLAG 100
      THEN
        POS [axis=21, Pos=38640, v=1000]
        POS [axis=22, Pos=16000, v=1000]
        POS [axis=23, Pos=0, v=1000]
    WHEN
        AXARR axis=21
        AXARR axis=22
        AXARR axis=23
      THEN
        -FLAG 100
    WHEN
        FLAG 100
      THEN
        REGISTER_LOAD [124001 with 94]
        REGISTER_LOAD [124005 with 1000]     ;acceleration ramp
        REGISTER_LOAD [124006 with 1000]     ;deceleration ramp
        REGISTER_LOAD [124003 with 200]      ;starting speed
        REGISTER_LOAD [121187 with 1]         ;Spline number
        REGISTER_LOAD [121001 with 110]       ;Start spline
    WHEN
        BIT_CLEAR [Reg=121000, Bit=13]
      THEN
    WHEN
        AXARR axis=21
        AXARR axis=22
        AXARR axis=23
      THEN
        REGISTER_LOAD [121001 with 42]
        REGISTER_LOAD [122001 with 42]
        REGISTER_LOAD [123001 with 42]
    WHEN
        BIT_CLEAR [Reg=121000, Bit=13]
        BIT_CLEAR [Reg=122000, Bit=13]
        BIT_CLEAR [Reg=123000, Bit=13]
      THEN
        GOTO 40
```

4 Technical Data

Condition		Standard
Ambient Temperature	0 ... +50 ^{O}C	
Storage Temperature	-10 ... +70 ^{O}C	
Air Humidity	5% ... 95%	RH-2 according to IEC61131-2 non-condensing
Pollution Degree		
Vibration Resistance		
Degree of Protection	IP20	
Class of Protection		
ESD		

Mechanical and Electrical Specifications	
Power Supply	+24V -15% / +20%
Connections	
Dimensions (H x W x D in mm)	
Power Consumption	??mA
Weight	
Installation	
Enclosure	