ROBOTICS

Application manual

Dispense

Trace back information:
Workspace Main version a564
Checked in 2024-02-07
Skribenta version 5.5.019

Application manual
Dispense

DispenseWare for Omnicore

Document ID: 3HAC089025-001

Revision: A

The information in this manual is subject to change without notice and should not be construed as a commitment by ABB. ABB assumes no responsibility for any errors that may appear in this manual.

Except as may be expressly stated anywhere in this manual, nothing herein shall be construed as any kind of guarantee or warranty by ABB for losses, damage to persons or property, fitness for a specific purpose or the like.

In no event shall ABB be liable for incidental or consequential damages arising from use of this manual and products described herein.

This manual and parts thereof must not be reproduced or copied without ABB's written permission.

Keep for future reference.

Additional copies of this manual may be obtained from ABB.

Original instructions.

Table of contents

Overview of this manual

About this manual

This manual describes the *DispenseWare* addin and contains instructions for the configuration.

Note

It is the responsibility of the integrator to provide safety and user guides for the robot system.

This manual should be used during installation and configuration of the *DispenseWare* addin.

Who should read this manual?

This manual is intended for:

- Personnel responsible for installations and configurations of field bus hardware/software
- Personnel responsible for I/O system configuration
- System integrators

Prerequisites

The reader should have the required knowledge of:

- Mechanical installation work
- Electrical installation work
- System parameter configuration
- RAPID programming language

References

References	Document ID
Operating manual - OmniCore	*3HAC065036-001*
Operating manual - RobotStudio	*3HAC032104-001*
Technical reference manual - System parameters	*3HAC065041-001*
Technical reference manual - RAPID Instructions, Functions and Data types	*3HAC065038-001*
Technical reference manual - RAPID Overview	*3HAC065040-001*

Revisions

Revision	Description
A	Released with DispenseWare 1.0.0.

This page is intentionally left blank

1 Introduction to DispenseWare

1.1 DispenseWare

Introduction

The Dispense package provides support for different types of dispensing processes such as gluing, sealing and similar.

The Dispense application provides fast and accurate positioning combined with a flexible process control. Communication with the dispensing equipment is carried out by digital and analog outputs.

Dispense is a package that can be extensively customized. User data and routines can be adapted to suit a specific dispensing equipment and environmental situation.

1.2 Dispensing features

Option content

- Dispensing instructions for both linear and circular paths.
- Fast and accurate positioning and process control.
- Handling of on/off guns as well as proportional guns.
- Speed proportional or constant analog outputs.
- Four different gun equipment can be handled in the same program, each controlled by 1-5 digital output signals and/or 1-2 analog output signals.
- Possibility to use different anticipated times for the digital and analog signals.
- Equipment delay compensation for the TCP speed proportional analog signals.
- Dispensing in wet or dry mode.
- Possibility to restart an interrupted dispense sequence.
- Possibility to automatically reduce the robot acceleration/deceleration during dispensing.
- Wide customizing possibilities.
- A dedicated Dispense operator interface on the FlexPendant.

1.3 Programming principles

Introduction

Both the robot movement and the dispensing process control are embedded in the instructions, `DispL` and `DispC`. See *RAPID reference on page 25*

The dispensing process is specified by:

- Bead specific dispensing data. See *beaddata on page 36*.
- Equipment specific dispensing data. See *equipdata on page 39*.
- RAPID routines and global data for customizing purposes. See *System modules on page 45*.
- System parameters. See *System parameters on page 47*.

Dispensing instructions

Instruction	Used to:
`DispL`	Move the TCP along a linear path and perform dispensing with the given data.
`DispC`	Move the TCP along a circular path and perform dispensing with the given data.

Dispensing data

Data type	Used to define:
`beaddata`	Dispensing data for the different beads.
`equipdata`	Equipment specific dispensing data.

Signals

The available signals are set during bead segments according to the configuration possibilities described in this manual, and in relation to the position of the tool center point and the programmed targets. The signals can be used for any equipment-specific purposes, and many setups will not have use for all signals. Some equipment might use multiple signals for controlling two physical pieces of equipment at once, and others may use multiple signals for more advanced equipment functions.

The digital gun signals can typically be used for simple purposes requiring only an on/off state during a bead, such as opening a gun or triggering an external event.

The analog flow signals can typically be used for purposes where different levels are required, such as pressure or function speed. These levels can be adjusted according to bead type and can also optionally be automatically adjusted during robot movement in relation to the actual speed of the tool center point.

This page is intentionally left blank

2 Programming

2.1 Introduction to programming

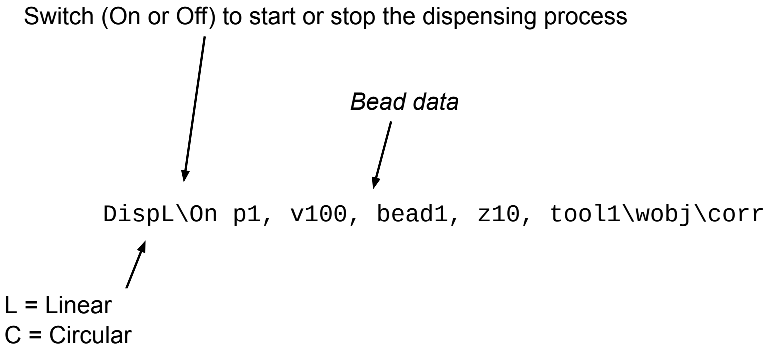
Introduction

When Dispense option is added to the RobotWare base system, it is ready to use with a default functionality. However, the functionality can be customized to adapt the package to a specific dispensing equipment, see *Customizing Dispense on page 85*.

This chapter describes primarily the default setup.

Dispense instructions

A dispense instruction contains the same type of information as a Move instruction. However, each dispense instruction also include further arguments for the dispensing process. See *DispL/DispC on page 25*.



```
DispL\On p1, v100, bead1, z10, tool1\wobj\corr
```

xx1200000143

Dispense data types

The dispense data should be defined before programming. The dispense data is divided in two types:

- `equipdata`
- `beaddata`

equipdata

The data type `equipdata` is used to hold equipment specific data. This is dispensing data which normally does not vary between different beads. It is possible to use up to four different pieces of equipment. `equipdata` for equipment 1-4 are located in an array (*equipd*) in the system module DPUSER. The data in current `equipdata` influences the process when dispensing instructions are executed.

`equipdata` has the following components:

- Information string (80 bytes)
- Reference speed for the scale calculations.
- Acceleration/deceleration limits for the robot movements.

Continues on next page

- Predict times for the digital (gun on and off) signals.
- Predict times for the analog flow signals.
- Equipment delay compensation times.
- Flow rate correction factors.

Change the data in the *equipd* array so it corresponds to the equipment used. The data is correctly calibrated when the bead is started/changed/ended in the programmed positions and when the bead width is constant in corners. See *equipdata on page 39*.

beaddata

`beaddata` is used to hold bead specific data. This is the dispensing data that determines the appearance of the bead and which normally varies between different beads. One `beaddata` (*bead1*) is predefined as default in the dispensing system module DPUSER.

`beaddata` has the following components:

- Information string (80 bytes)
- Flow rates for flow1 and flow2
- Flow types for flow1 and flow2
- Equipment number
- Gun number

We recommend that one `beaddata` for each used bead type is defined and stored in DPUSER, before the instruction sequence is programmed. See *beaddata on page 36*.

Dispense programming philosophy

Dispense supports a bead oriented programming philosophy. Parameters affecting the characteristics of the bead are included in the `beaddata` for each beadtype. This technique separates the defining of the bead parameters from robot path teaching and simplifies off line programming as well as manual teaching and touch up.

In the default `beaddata` and `equipdata` version the most used data components are present. During customizing it is possible to hide (delete) components if they not are valid for the specific process or equipment. it is also possible to add some components if desired. For more information, see *Customizing Dispense on page 85*.

2.2 Programming dispense instructions

Programming

	Action	Note
1	Jog the robot to the desired position for the dispensing start.	
2	Tap **Add Instruction** and select the **Dispense** pick list.	
3	Add the instruction `DispL\On` or `DispC\On`.	The arguments are set in relation to the latest programmed dispense instruction.
4	If needed, select `beaddata` and change other arguments.	
5	Jog to next position on the bead.	Normally a point where the path is changed or a position where the bead size is changed.
6	Add another instruction, `DispL` or `DispC`.	
7	If needed, change the arguments.	
8	Continue this way until the last position on the bead is reached.	
9	Add the end instruction `DispL\Off` or `DispC\Off`, and change the arguments if needed.	

Limitations

One equipment is controlled by 1-5 gun on/off signals and 1-2 analog flow signals. It is possible to use up to four different pieces of equipment.

It is not possible to use different pieces of equipment within one sequence of dispensing instructions (from `DispL\On` to `DispL\Off`).

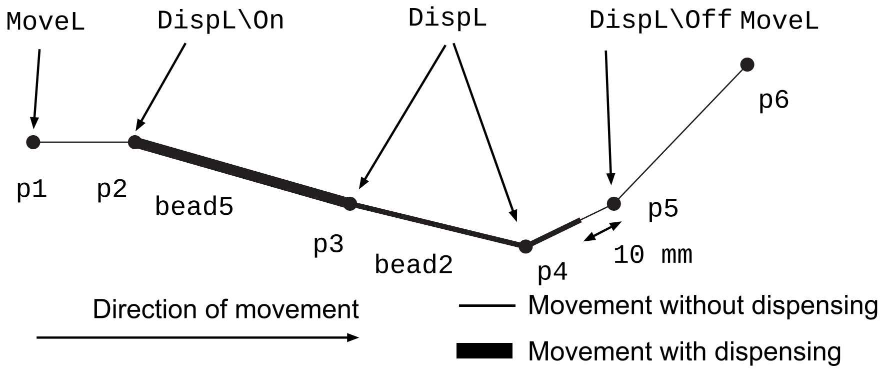
It is not possible to change from flow control type 2 to flow control type 1 within one sequence of dispensing instructions.

It is not recommended to use system inputs to change the programmed TCP speed during dispensing. This includes the following features:

- System input *Limit Speed*
- Setting *Speed Override* from the FlexPendant
- System input *Set Speed Override*

Continues on next page

Programming example



xx1200000606

```
bead5
  info = "width 5mm"
  flow1 = 80
  flow2 = 40
  flow1_type = 2
  flow2_type = 1
  equip_no = 1
  gun_no = 1
bead2
  info = "width 2mm"
  flow1 = 35
  flow2 = 20
  flow1_type = 2
  flow2_type = 1
  equip_no = 1
  gun_no = 1
```

```
equipd{1}
  info = "equipment 1"
  on_time = 0.05
  off_time = 0.05
  ref_speed = 800
  acc_max = 5
  decel_max = 5

  fl1_on_time = 0.10
  fl1_off_time = 0
  fl1_inc_time = 0.06
  fl1_dec_time = 0.12
  fl1_delay = 0.05
  fl1_corr = 100

  fl2_on_time = 0.15
  fl2_off_time = 0
  fl2_inc_time = 0.08
  fl2_dec_time = 0.1
  fl2_delay = 0.04
  fl2_corr = 100
```

RAPID code sequence:

```
MoveL p1, v600, fine, tool1;
DispL \On, p2, v400, bead5, z10, tool1;
DispL p3, v400, bead2, z10, tool1;
DispL p4, v500, bead2, z10, tool1;
DispL \Off, p5, v500, bead2 \D:=10, z10, tool1;
MoveL p6, v600, fine, tool1;
```

2.3 Editing dispense instructions

Changing a beaddata parameter

	Action
1	Select Data in the Dispense GUI.
2	Select `beaddata` in the **Data** menu in the command bar.
3	Select desired `beaddata` and change values.
4	Tap **OK**.

Changing to another beaddata

	Action
1	Select current `beaddata` in the instruction.
2	Tap **Edit** and then **Change Selected**.
3	Select `beaddata` from the list.
4	Tap **OK**.

2.4 Testing dispense instructions

Testing without dispensing

It is possible to run the program in simulation mode without activating any dispense signals. This can be done by setting the DPUSER data *dp_dry* to TRUE.

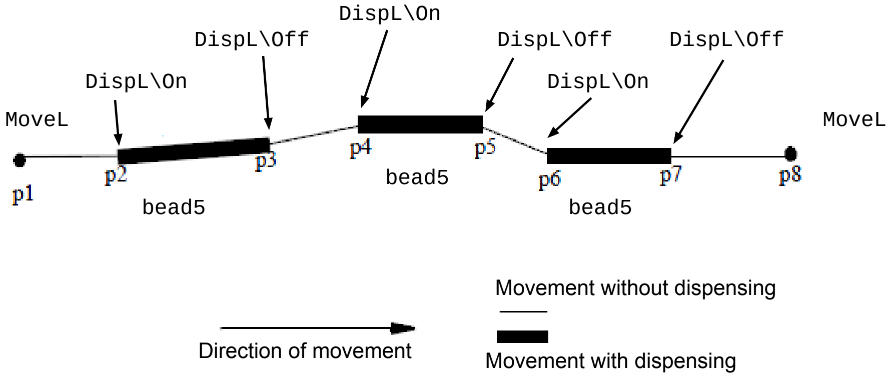
Testing dispense instructions step by step

The dispense signals are not activated when dispense instructions are executed step by step.

2.5 Programming when short beads are used

Example 1

In this example a number of short beads are programmed. The On and Off argument is used for all beads that will terminate the dispensing process between the beads.



xx1200000627

```
bead5
  info = "width 5mm"
  flow1 = 80
  flow2 = 40
  flow1_type = 2
  flow2_type = 2
  equip_no = 1
  gun_no = 1
```

RAPID code sequence:

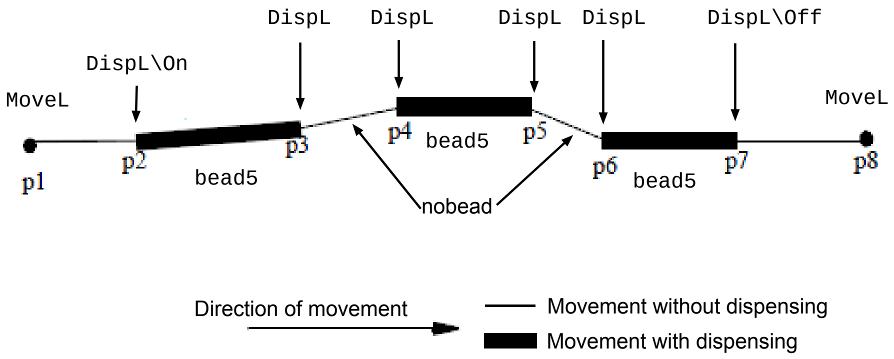
```
MoveL p1, v600, fine, tool1;
DispL \On, p2, v600, bead5, z50, tool1;
DispL \Off, p3, v600, bead5, z50, tool1;
DispL \On, p4, v600, bead5, z50, tool1;
DispL \Off, p5, v600, bead5, z50, tool1;
DispL \On, p6, v600, bead5, z50, tool1;
DispL \Off, p7, v600, bead5, z50, tool1;
MoveL p8, v600, fine, tool1;
```

In this example the analog flow signals are deactivated between every bead, but often it is desired to run short bead interrupts without deactivating the analog flow signals. Only the digital gun signal(s) are deactivated during the interrupt. This can be done as shown in next example.

Continues on next page

Example 2

The same beads as in previous example are programmed but in this case we do not deactivate the analog flow signals between the beads. A separate bead data, *nobead*, is created to be used between the beads. In this `beaddata` the data component `gun_no` is set to zero, all other components are set as in previous `beaddata`. The process is started with the `On` argument in the beginning of the first bead, and terminated with the `Off` argument in the end of the last bead.



xx1200000627

```
bead5
  info = "width 5mm"
  flow1 = 80
  flow2 = 40
  flow1_type = 2
  flow2_type = 2
  equip_no = 1
  gun_no = 1
```

```
nobead
  info = "no bead"
  flow1 = 80
  flow2 = 40
  flow1_type = 2
  flow2_type = 2
  equip_no = 1
  gun_no = 0
```

RAPID code sequence:

```
MoveL p1, v600, fine, tool1;
DispL \On, p2, v600, bead5, z50, tool1;
DispL p3, v600, nobead, z50, tool1;
DispL p4, v600, bead5, z50, tool1;
DispL p5, v600, nobead, z50, tool1;
DispL p6, v600, bead5, z50, tool1;
DispL \Off, p7, v600, bead5, z50, tool1;
MoveL p8, v600, fine, tool1;
```

Keeping the signals between beads

If the argument `fl1_off_time` or `fl2_off_time` is set to -1 then the automatic reset of the flow signal is deactivated. The signal holds the latest value given from the `beaddata` in the `\Off` instruction until start of next bead, see *equipdata on page 39*.

By using this method it is possible to program as in example 1, without activating the analog signals between the beads.

Continues on next page

Limitations

When short beads or bead interrupts are used in combination with high process speeds it is not possible to use shorter beads than the difference between the on and off time components in equipdata. If the beads are shorter the signals are deactivated before they are activated, that is the beads are not dispensed. In these situations the process speed has to be reduced.

2.6 Softening the robot movements

Description

If necessary it is possible to reduce the robot acceleration or deceleration to get the best behavior from the used dispensing equipment.

If the data components `acc_max` or `decel_max` are activated in the `equipdata` for a specific equipment, then the limitation is automatically activated in the beginning of the bead and reset to previous value in the end of the bead, where this equipment is used. See *equipdata on page 39*.

2.7 Using the Dispense Restart function

About the Dispense Restart function

The system parameters for the Dispense Restart function is found in topic *Process*, type *Dispense Restart*.

There is one set of parameters for each available equipment, *Equipment_1* to *Equipment_4*.

Activating the Dispense Restart function

Set the restart type by changing the system parameter *Restart type*. It is possible to set three different restart levels for each equipment.

0	Restart is disconnected. (Default). The process is started in the next bead, i.e. in the next `Disp\On` instruction.
1	Restart of current bead with possibilities to select wet or dry.
2	Restart of current bead in a no query mode (without questions)

See *DispL/DispC on page 25*.

Set up desired restart behavior

Test a common restart situation for each equipment by adjusting the following system parameters to get the desired restart behavior:

Gun on anticipate	The preaction time for the digital gun signal
Flow anticipate	The preflow time for the analog signals
Preflow value flow1	The preflow value for the analog signal 1
Preflow value flow2	The preflow value for the analog signal 2
Backward distance after stop	The backward distance before restart after a program stop
Backward distance after halt	The backward distance before restart after a halt with motors off

For more information about system parameters, see *Topic Process on page 49*.

Using a programmable key for the backward on path function

Normally the TCP is moved a predefined distance backward on the path before the process is restarted. But if restart type 1 is selected it is also possible to use one of the FlexPendant programmable keys for activation of the movement. The backward movement is going on as long as the key is pressed.

Activate this function by changing the system parameter *Backward key* for each equipment.

0	No key is used. (Default)
1 - 4	Desired key number

If this function is activated the selected key has to be configured as a programmable key, see *Configuring a key for the backward on path function on page 24*.

Continues on next page

Configuring a key for the backward on path function

1. On the FlexPendant, open the **Settings** view.
2. Tap **Personalization**, **Programmable keys**.
3. Select the key to be used.
4. Set following parameters for the key:
   - **Type**: **Output**
   - **Digital output**: **doBwdOnPath**
   - **Key Pressed**: **Press/Release**
   - **Allow in Auto**: **Yes**

3 RAPID reference

3.1 Instructions

3.1.1 DispL/DispC

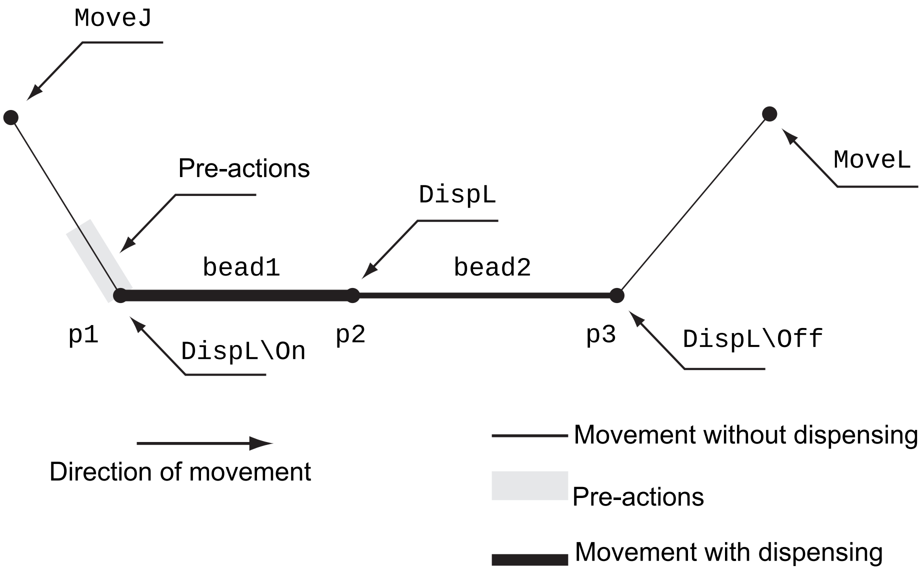
Usage

`DispL` and `DispC` are used in dispensing applications to control the robot motion, gun opening, and the dispensing process. `DispL` moves the TCP linearly to the end position. `DispC` moves the TCP circularly to the end position.

Basic examples

In this example a bead starts at point p1, changes to another bead at p2, and then ends at point p3.

```
DispL \On, p1, v250, bead1, z30, tool7;
DispL p2, v250, bead2, z30, tool7;
DispL \Off, p3, v250, bead2, z30, tool7;
```



xx1200000146

1. The TCP for tool 7 is moved on a linear path to the position p1 with the speed given in v250. Due to the `\On` argument the gun opens and the dispense flow is started in advance on its way to p1. The flow rates and the times when the signals are activated, are specified in `beaddata bead1` and in current `equipdata` in the `equipdata` array in DPUSER. See *System modules on page 45*.
2. The TCP is then moved from p1 towards p2 with the flow values given by `beaddata bead1` activated in the preceding dispensing instruction. Before

Continues on next page

p2 is reached, the flow values are changed according to the data specified in bead2. The time when this is performed is specified in current equipdata.

3. The TCP is then moved from p2 towards p3 with the flow values specified by beaddata bead2 activated in the preceding dispensing instruction. Due to the Off argument the outputs will be reset, according to the times specified in current equipdata connected to bead2, before p3 is reached.

Arguments

```
DispL [\On]|[\Off] ToPoint Speed Bead [\D] Zone Tool [\WObj] [\Corr]
     [\T1] | [\TArray{*}] [\TLoad]
DispC [\On]|[\Off] CirPoint ToPoint Speed Bead [\D] Zone Tool
     [\WObj] [\Corr] [\T1] | [\TArray{*}] [\TLoad]
```

[\On]

Data type: switch

The argument \On is used in the first dispensing instruction in a sequence, to start the dispensing process. The argument may only be used in the first dispense instruction in a sequence, to perform the necessary gun opening and setting of the flow in advance. Executing two consecutive instructions with \On argument will result in an error message.

[\Off]

Data type: switch

The argument \Off is used in the last dispense instruction in a sequence, to terminate the process when the programmed position is reached. On the way to the end position the output for opening the gun as well as the flow outputs will be reset according to the given time within the specified data.

CirPoint

Data type: robtarget

The circle point of the robot. The circle point is a position on the circle between the start point and the destination point. To obtain the best accuracy it should be placed about halfway between the start and destination points. If it is placed too close to the start or destination point, the robot may give a warning. The circle point is defined as a named position or stored directly in the instruction (marked with an * in the instruction). The position of the external axes are not used.

This argument is only used in DispC.

ToPoint

Data type: robtarget

The destination point of the robot and external axes. It is defined as a named position or stored directly in the instruction (marked with an * in the instruction).

Speed

Data type: speeddata

The speed data that applies to movements. Speed data defines the velocity of the TCP, the tool reorientation, and external axes.

Continues on next page

Bead

Data type: `beaddata`

Bead specific data, for example the flow rates and flow types. For further details, see *beaddata on page 36*.

[\D]

Distance

Data type: `num`

The optional argument `\D` gives the possibility to offset all pre-actions a given distance (mm) on the programmed path.

A positive value will offset the actions to before the programmed position, while a negative value will offset the actions to after the programmed position.

This argument is useful for adjusting the start and stop of individual bead segments without altering the programmed path.

Limitation: If the given distance causes an overlap between the pre-actions of this position and the next/previous one, the resultant behavior might not be what is intended. Take care when using this argument in combination with shorter segments.

`Zone`

Data type:`zonedata`

Zone data for the movement. Zone data describes the size of the generated corner path.

`Tool`

Data type: `tooldata`

The tool in use when the robot moves. The tool center point is the point that is moved to the specified destination point.

`[ \WObj ]`

Work Object

Data type: `wobjdata`

The work object (object coordinate system) to which the robot position in the instruction is related.

This argument can be omitted and if it is then the position is related to the world coordinate system. If, on the other hand, a stationary TCP or coordinated external axes are used this argument must be specified in order for a circle relative to the work object to be executed.

`[ \Corr ]`

Correction

Data type: `switch`

Correction data written to a corrections entry by the instruction `CorrWrite` will be added to the path and destination position if this argument is present.

The RobotWare option *Path Corrections* is required when using this argument.

Continues on next page

[\T1]

Trigg 1

Data type: `triggdata`

Variable that refers to trigger conditions and trigger activity, defined earlier in the program using the instructions `TriggIO`, `TriggEquip`, `TriggInt`, and similar. Compare with the `TriggL` instruction.

[\TArray{*}]

Trigg Data Array Parameter

Data type: `triggdata`

Array variable that refers to trigger conditions and trigger activity defined earlier in the program using the instructions `TriggIO`, `TriggEquip`, `TriggInt`, and similar.

The limitation is 10 elements in the array.

[\TLoad]

Total Load

Data type: `loaddata`

The total load used in the movement. The total load is the tool load together with the payload the tool is carrying. If the `\TLoad` argument is used, then the `loaddata` in current tooldata is not considered. If the `\TLoad` argument is set to load0, then the `\TLoad` argument is not considered and the `loaddata` in current `tooldata` is used instead. For a complete description of the `TLoad` argument, see *MoveL - Moves the robot linearly*.

Program execution

Internal sequence when a `DispL/DispC` instruction is executed:

- The gun starts to move towards the position.
- If the argument `\On` is used, the gun opening output (or outputs) and the analog outputs are set at the specified time before the position is reached. See *equipdata on page 39*. The Equipment Active signal (for example the output signal doEqu1Active for equipment 1) is set together with the gun opening output.
- If nor `\On` or `\Off` is used the gun opening output (or outputs) and the analog outputs are set (or changed) at the specified time before the position is reached.
- If the argument `\Off` is used the gun opening output (or outputs) and the analog outputs are reset at the specified times before the position is reached. The Equipment Active signal is reset together with the digital gun signals.
- If the argument `\D` is used, all pre-actions are performed specified times (in equipdata) plus given distance (mm) in advance of the programmed position.
- When the programmed position is reached, the program execution continues with the next instruction.

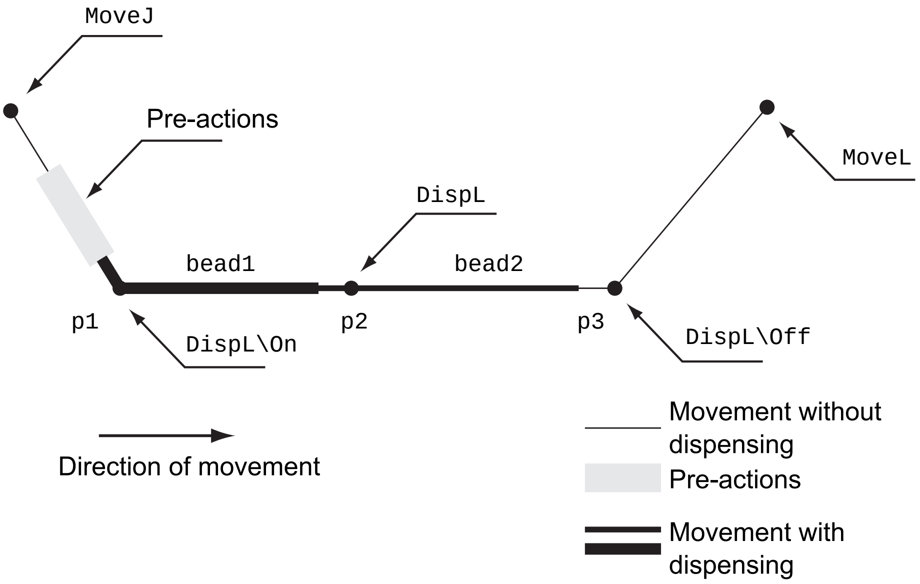
Continues on next page

More examples

Example 1

In this example, everything is the same as in the previous example except that all bead changes are performed an additional distance (50 mm) before the programmed positions are reached.

```
DispL \On, p1, v250, bead1\D:=50, z30, tool7;
DispL p2, v250, bead2 \D:=50, z30, tool7;
DispL \Off, p3, v250, bead2 \D:=50, z30, tool7;
```



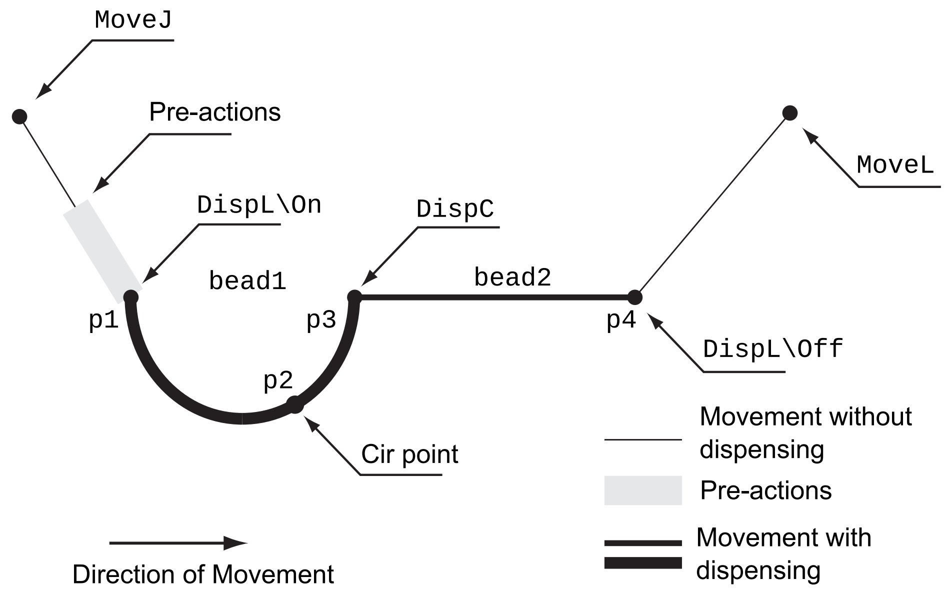
xx1200000147

Example 2

In this example, `DispC` is used:

```
DispL \On, p1, v250, bead1, z30, tool7;
DispC p2, p3, v250, bead2, z30, tool7;
DispL \Off, p4, v250, bead2, z30, tool7;
```

Continues on next page



xx1200000145

Instruction by instruction execution forwards

The gun is closed and motion without dispensing is done.

Instruction by instruction execution backwards

The gun is closed and the motion is performed backwards without dispensing.

Simulated dispensing

Activated by setting the variable `dp_dry` to `TRUE`. This will inhibit the gun opening and the flow signals. See *System modules on page 45*.

Limitations

`DispL/DispC` cannot be executed in a RAPID routine connected to any of the following special system events: PowerOn, Stop, QStop, Restart, Reset, or Step.

Error handling

The following error situations are handled:

- Instruction argument error.
- Not permitted data values.
- Start without \On argument.
- Start with two instructions with \On argument.
- One sequence of Disp instructions cannot use different equipment.
- Stop during execution of dispense instructions.

Error actions:

- The digital on/off signals are cleared.
- The digital *Equipment Active* signal is cleared.

Continues on next page

- The analog flow signals are cleared.
- The digital *Equipment Error* signal is set.
- The program execution is interrupted with the program pointer on the interrupted instruction.

Internal errors are handled internally, but to set the program pointer on the interrupted `Disp` instruction the error is also raised to the user level. But since this internal errors not are handled by the error handlers on the user level, the system error handler automatically will report the error and stop the program execution.

Flow Control Types

There are two different Flow Control Types for the analog signals. See *beaddata on page 36*:

Speed independent (1)	The analog flow is a constant value, proportional to the flow rate specified in the current bead data.
TCP speed proportional(2)	The analog flow is directly proportional to the actual robot velocity. When for example the speed is reduced in a corner path, the flow will also be reduced correspondingly.

Calculation of flow values

The following data is used when the logical flow1 value is calculated. (The logical flow2 is calculated in same way):

flow1	flow rate from current bead data. See *beaddata on page 36*.	
dp_fl1_corr	global flow rate correction factor. See *System modules on page 45*.	Default: 100
fl1_corr	flow rate correction factor. See *equipdata on page 39*.	Default: 100
current speed	current robot speed (mm/s).	
ref_speed	reference speed (mm/s). See *equipdata on page 39*.	

Calculation of logical flow1 when flow1_type = Speed independent (1):

```
logical flow1 = flow1 * dp_fl1_corr * fl1_corr/ 10000
```

Calculation of logical flow1 when flow1_type = TCP speed proportional (2):

```
logical flow1 = (flow1 * dp_fl1_corr * fl1_corr/ 10000) * current
     speed/ref_speed
```

The physical values of the signals are then determined by how the analog signals are configured in the system parameters (relationship between physical and logical values).

For example:

With the default values above and with the default setup for logical max. and min. for the analog outputs (see *System parameters on page 47*), the following result is obtained:

If *flow1_type* = Speed independent (1):

Physical max. value is activated if flow1 = 100 in current bead data.

Or

Continues on next page

If *flow1_type* = TCP speed proportional (2):

Physical max. value is activated if flow1 = 100 in current bead data and the actual speed is the same as *ref_speed* in current *equipdata*.

Soften the robot movements

If necessary it is possible to reduce the robot acceleration or deceleration to get the best behavior from the used dispensing equipment.

If the data components `acc_max` or `decel_max` are activated in the equipdata for a specific equipment, then the limitation is automatically activated in the beginning of the bead and reset to previous value in the end of the bead, where this equipment is used. See *equipdata on page 39*.

Stop and restart during the dispensing process

It is possible to set three different restart levels for each equipment.

0	Restart is disconnected. (Default). The process is started in the next bead, i.e. in the next `Disp\On` instruction.
1	Restart of current bead with possibilities to select wet or dry.
2	Restart of current bead in a no query mode (without questions)

For more information, see *Using the Dispense Restart function on page 23*.

Stop sequences

The stop sequence when a `Program Stop` occur during the dispensing process:

	Action
1	The dispensing process is active during the stop phase. If TCP speed proportional analog signals are used the analog flow values will be automatically reduced during the deceleration.
2	The robot is halted on the path.
3	The digital gun signals (for gun on/off) are set to zero.
4	An information text is printed out on the FlexPendant telling if it is possible to restart the process or not.
5	If it not is possible to restart the process the digital Equipment Error signal is activated.

The stop sequence when a halt with motors off (for example `Emergency Stop`) occur during the dispensing process:

	Action
1	The robot is halted as soon as possible, probably with a deviation from the programmed path.
2	The gun signals are set to zero as soon as possible during the deceleration.
3	An information text is printed out on the FlexPendant telling if it is possible to restart the process or not.
4	If it is not possible to restart the process the digital Equipment Error signal is activated.

Continues on next page

Restart sequences

The Restart sequence with Restart type 0 (Default):

	Action
1	A regain motion back to the interrupted position on the path is done (after a halt with motors off).
2	The remaining instructions in the current set of dispense instructions are performed as normal positioning instructions. The dispensing is restarted in the next `Disp` instruction with an `\On` argument.

The Restart sequence with Restart type 1:

	Action
1	A regain motion back to the interrupted position on the path is done (after a halt with motors off).
2	If it is possible to restart, a text is presented on the display asking if the restart shall be **Wet** or **Dry**. It is also possible to select **Bwd**. • **Dry** means restarting without activating the digital gun signals. If **Dry** is selected the process is restarted in next `Disp\On` instruction. • If **Wet** is pressed the process is restarted directly from current position. • If **BWD** is selected a new dialog is activated with possibility to move the robot backward on the path. This function is useful if the bead is interrupted before the stop position. When the backward movement is ready previous dialog appear again. (**Wet** or **Dry**).
3	When **Wet** is selected the digital gun signals and the analog flow signals are activated user defined times before the robot motion starts. (`restart_on_time` and `restart_fl_time`).
4	Before the signals are activated the user routine `dp_restart_proc` (in DPUSER) is executed with possibilities to add some user actions in the restart sequence.

The Restart sequence with Restart type 2:

	Action
1	A regain motion back to the interrupted position on the path is done (after a halt with motors off).
2	If it is possible to restart, the robot is automatically moved backward on the path a user defined distance. Different distances can be used after program stop (`stop_bwddist`) and halt with motors off (`qstop_bwddist`).
3	The digital gun signals and the analog flow signals are activated user defined times before the robot motion starts. (`restart_on_time` and `restart_fl_time`).
4	Before the signals are activated the user routine `dp_restart_proc` (in DPUSER) is executed with possibilities to add some user actions in the restart sequence.

Note

The bead quality when the process is restarted after a halt with motors off will not be the same as after a program stop, mainly because that the robot deviates from the programmed path during the deceleration and also because during halt with motors off it is not possible to compensate for delays in the dispensing equipment. This sometimes results in too much material in the stop position. However, to avoid these problems safeguarded stops can be configured to be soft which gives a smooth stop on the path in these cases.

Continues on next page

Note

The possibility to move the robot backward on the path is limited to one or a few segments. If it is not possible to move the robot a desired distance backward on the path an error text will appear.

Note

It is not possible to restart current bead if Power Failure occur during the dispensing process. After Power On an error text is written on the display. If the program is restarted the process is restarted in next `Disp\On` instruction.

Syntax

```
DispL
  [['\'On ',']|['\'Off ',']]
  [ToPoint':=']<expression (IN) of robtarget>
  [Speed':=']<expression (IN) of speeddata>','
  [Bead':=']<persistent (PERS) of beaddata>
  ['\'D':='<expression (IN) of num>]
  [Zone':=']<expression (IN) of zonedata>','
  [Tool':=']<persistent (PERS) of tooldata>
  ['\'WObj':='<persistent (PERS) of wobjdata>]
  ['\'Corr]
  ['\'T1':='<variable (VAR) of triggdata>] |
  ['\'TArray':='<array variable {*} (VAR) of triggdata>]
  ['\'TLoad':='<persistent (PERS) of loaddata>]';'
DispC
  [['\'On ',']|['\'Off ',']]
  [CirPoint':=']<expression (IN) of robtarget>','
  [ToPoint':=']<expression (IN) of robtarget>
  [Speed':=']<expression (IN) of speeddata>','
  [Bead':=']<persistent (PERS) of beaddata>
  ['\'D':='<expression (IN) of num >]
  [Zone':=']<expression (IN) of zonedata>','
  [Tool':=']<persistent (PERS) of tooldata>
  ['\'WObj':='<persistent (PERS) of wobjdata>]
  ['\'Corr]
  ['\'T1':='<variable (VAR) of triggdata >] |
  ['\'TArray':='<array variable {*} (VAR) of triggdata>]
  ['\'TLoad':='<persistent (PERS) of loaddata>]';'
```

Related information

	Described in:
Definition of velocity, `speeddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of zone data, `zonedata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of tool, `tooldata`	*Technical reference manual - RAPID Instructions, Functions and Data types*

Continues on next page

	Described in:
Definition of work objects, `wobjdata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of `beaddata`	*beaddata on page 36*
Definition of `equipdata`	*equipdata on page 39*
Customizing the functionality	*Customizing Dispense on page 85*
System module DPUSER	*System modules on page 45*
I/O configuration	*Topic I/O on page 47*
`MoveL`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Definition of load, `loaddata`	*Technical reference manual - RAPID Instructions, Functions and Data types*
Motion in general	*Technical reference manual - RAPID Overview*
Other positioning instructions	*Technical reference manual - RAPID Overview*

3.2 Data types

3.2.1 beaddata

Usage

`beaddata` is used in dispensing applications to hold bead specific data. This is the dispensing data which determines the appearance of the bead and which normally varies between different beads.

Description

`beaddata` is used in `DispL` and `DispC` instructions and has the following default contents:

- Information string (80 bytes)
- Flow rates for flow1 and flow2.
- Flow control types for flow1 and flow2.
- Equipment number.
- Gun number (if multiple guns are used).

Arguments

Info

data information

Data type: `string`

String for external use, maximum 80 bytes.

flow1

flow rate

Data type: `num`

Flow rate for the flow1 signal. Normal values: 0-100

flow2

flow rate

Data type: `num`

Flow rate for the flow2 signal. Normal values: 0-100

fl1_type

type of flow1

Data type: `num`

Flow control type for flow1. Permitted values:

- 0 - Flow1 signal not used
- 1 - Constant, speed independent
- 2 - Speed proportional

fl2_type

type of flow2

Continues on next page

Data type: `num`

Flow control type for flow2. Permitted values:

- 0 - Flow2 signal not used
- 1 - Constant, speed independent
- 2 - Speed proportional

equip_no

equipment number

Used equipment for this bead. Permitted values: 1-4

gun_no

gun number

Gun on/off signal selection. Allowed values:

Value	Description
0	No on/off signal is activated
1	Gun 1 is activated (single gun)
>1	Selected guns when multiple guns are used: • for example 135 means that the guns 1, 3, and 5 are activated. • Max 5 guns can be activated (gun_no = 12345).

Limitations

Max number of guns for each pieces of equipment when multiple guns are used: 5

Max number of pieces of equipment: 4

It is not possible to change equipment number within the same sequence of dispense instructions (from `Disp\On` to `Disp\Off`).

It is not possible to change from flow control type 2 to flow control type 1 within the same sequence of dispense instructions.

Predefined data

```
PERS beaddata bead1 := [" ",100,100,2,2,1,1];
```

Defined in module DPUSER.

The predefined data `bead1` defines flow rate 100 for flow1 and flow2 both of type 2, that is speed proportional flows are used. Only one gun is activated. Equipment number 1 is used. This equipment is defined in equipment data number 1 in the array `equipd` in DPUSER. `Bead1` is used as default in the first programmed `DispL` or `DispC` instruction.

Customizing

The Dispense functionality can be customized to adapt to different types of dispensing equipment. For this data type it is possible to delete components if they are not used. It is also possible to add some components and give components own user defined names. See *Customizing Dispense on page 85*.

Continues on next page

Default structure

```
< dataobject of beaddata >
  <info of string>
  <flow1 of num>
  <flow2 of num>
  <fl1_type of num>
  <fl2_type of num>
  <equip_no of num>
  <gun_no of num>
```

Related information

	Described in:
Dispensing instructions	*DispL/DispC on page 25*
Definition of `equipdata`	*equipdata on page 39*
Customizing the functionality	*Customizing Dispense on page 85*
System module DPUSER	*System modules on page 45*

3.2.2 equipdata

Usage

`equipdata` is used in dispensing applications to hold equipment specific data. This is dispensing data which does not normally vary between different beads.

`equipdata`, for each piece of equipment used, is stored in the array `equipd` in the system module `DPUSER`.

Description

`equipdata` has the following default contents:

- Information string (80 bytes)
- Reference speed for the scale calculations.
- Acceleration and deceleration values.
- Anticipated times for the digital gun on/off signals.
- Anticipated times for the analog flow signals.
- Equipment delay compensations.
- Flow rate correction factors.

Arguments

Info

data information

Data type: `string`

String for external use, maximum 80 bytes.

on_time

gun on anticipate

Data type: `num`

Predicted time in seconds for activation of the digital gun on/off signals.

off_time

gun off anticipate

Data type: `num`

Predicted time in seconds for deactivation of the digital gun on/off signals.

ref_speed

reference speed

Data type: `num`

Reference speed in mm/s. Normally 10 - 20% more than the max. dispensing speed used for the equipment in question. Is used in the scale calculation for speed proportional signals. This value must be > 0. See *Calculation of flow values on page 31*.

acc_max

maximum acceleration

Data type: `num`

Continues on next page

The absolute value of the limitation in m/s^2. The TCP acceleration along the path is limited from the beginning to the end of the bead. If the value is set to -1 this function is deactivated.

decel_max

maximum deceleration

Data type: `num`

The absolute value of the limitation in m/s^2. The TCP deceleration along the path is limited from the beginning to the end of the bead. If the value is set to -1 this function is deactivated.

fl1_on_time

flow1 on anticipate

Data type: `num`

Predicted time in seconds for the activation of the flow1 signal when the dispensing instruction is programmed with the `\On` argument.

fl1_off_time

flow1 off anticipate

Data type: `num`

Predicted time in seconds for deactivation of the flow1 signal when the dispensing instruction is programmed with the `\Off` argument.

If the argument is set to -1 then the automatic reset of the flow signal is deactivated. The signal holds the latest value given from the `beaddata` in the `\Off` instruction until start of next bead.

fl1_inc_time

flow1 increase anticipate

Data type: `num`

Predicted time in seconds for increasing the flow1 signal at positions where the bead is changed to another.

fl1_dec_time

flow1 decrease anticipate

Data type: `num`

Predicted time in seconds for decreasing the flow1 signal at positions where the bead is changed to another.

fl1_delay

equipment delay

Data type: `num`

Time in seconds to compensate the flow1 signal for the lag in the dispensing equipment when the TCP speed dips, for example at corners.

fl1_corr

flow1 correction

Data type: `num`

Continues on next page

Flow rate correction factor used in the scale calculations for flow1. Normal value: 100 (%)

fl2_on_time

flow2 on anticipate

Data type: `num`

Anticipated time in seconds for the activation of the flow2 signal when the dispensing instruction is programmed with the `\On` argument.

fl2_off_time

flow2 off anticipate

Data type: `num`

Anticipated time in seconds for the deactivation of the flow2 signal when the dispensing instruction is programmed with the `\Off` argument.

If the argument is set to -1 then the automatic reset of the flow signal is deactivated. The signal holds the latest value given from the `beaddata` in the `\Off` instruction until start of next bead.

fl2_inc_time

flow2 increase anticipate

Data type: `num`

Anticipated time in seconds for increasing the flow2 signal at positions where the bead is changed to another.

fl2_dec_time

flow2 decrease anticipate

Data type: `num`

Anticipated time in seconds for decreasing the flow2 signal at positions where the bead is changed to another.

fl2_delay

equipment delay

Data type: `num`

Time in seconds to compensate the flow2 signal for the lag in the dispensing equipment when the TCP speed dips, for example at corners.

fl2_corr

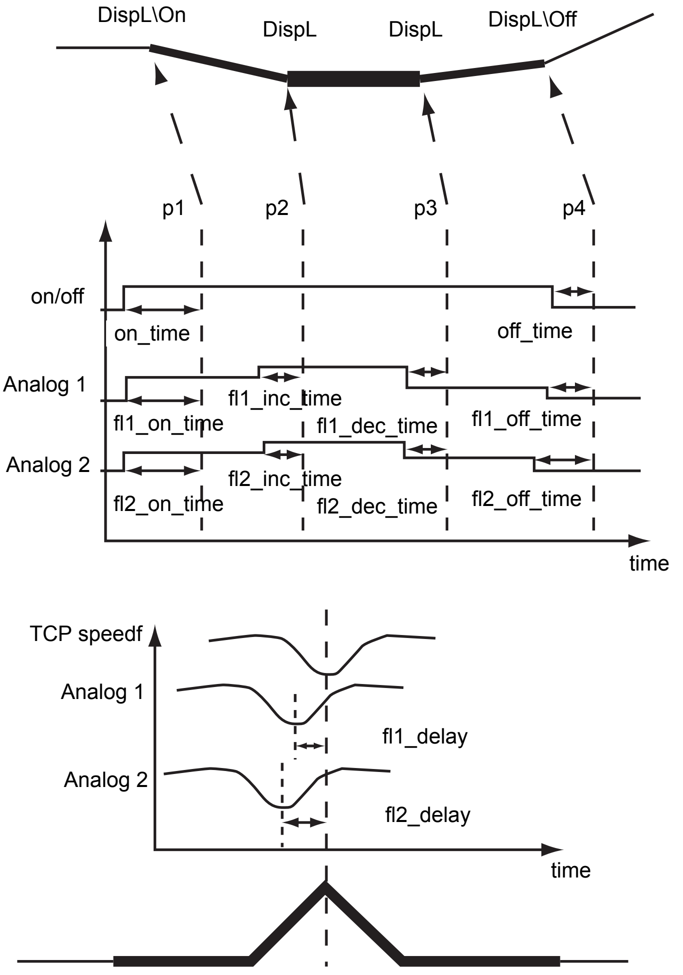
flow2 correction

Data type: `num`

Flow rate correction factor used in the scale calculations for flow2. Normal value: 100 (%)

Continues on next page

Time component description



xx1200000584

Limitations

Typical repeat accuracy values for the digital (gun on/off) dispense signal settings (CAN/DeviceNet): +/- 1 ms.

Max equipment delay compensation (fl1_delay and fl2_delay) is 0.5 seconds but to get a larger compensation than the servo lag, the system parameter *EventPresetTime* must be increased to a corresponding level. This system parameter can be found in the topic *Motion*, type *Motion Planner*. A run time error is generated if the *EventPresetTime* is too small. Max *EventPresetTime* is 0.5 s, but to not lose cycle time, do not use larger *EventPresetTime* than necessary.

Normally all time components have positive values but it is possible to use also negative values for the time components except for the `fl1_delay` and `fl2_delay`. Negative values are limited to 100 ms.

Continues on next page

Recommendations for best accuracy

It can be necessary to change the *EventPresetTime* also for the other time components, to get best accuracy when large predicted times are used. If some predicted times are larger than 50 ms, we recommend setting the *EventPresetTime* to the same value as the maximum used predicted time.

In some special situations it is not possible to activate the gun signals in right position. This can occur for example if the distance between programmed dispense positions are short in combination with high speeds. However, it is possible to supervise these situations during the program test by activating the configuration parameter *TimeEventSupervison*. This system parameter also belongs to the type *Motion Planner*, in the topic *Motion*.

Predefined data

```
PERS equipdata equipd{4} :=
  [[" ",0,0,1000,-1,-1,0,0,0,0,0,100,0,0,0,0,0,100]
  [" ",0,0,1000,-1,-1,0,0,0,0,0,100,0,0,0,0,0,100]
  [" ",0,0,1000,-1,-1,0,0,0,0,0,100,0,0,0,0,0,100]
  [" ",0,0,1000,-1,-1,0,0,0,0,0,100,0,0,0,0,0,100]];
```

This array of `equipdata` is predefined in the module DPUSER but the data components are intended to be changed by the user to define the actual equipment.

Customizing

The Dispense package provides opportunities for the user to customize the functionality to adapt to different types of dispensing equipment. For this data type it is possible to delete components if they are not used. It is also possible to add some components and give components own user defined names.

See *Customizing Dispense on page 85*.

Default structure

```
< dataobject of equipdata >
  <info of string>
  <on_time of num>
  <off_time of num>
  <ref_speed of num>
  <acc_max of num>
  <decel_max of num>
  <fl1_on_time of num>
  <fl1_off_time of num>
  <fl1_inc_time of num>
  <fl1_dec_time of num>
  <fl1_delay of num>
  <fl1_corr of num>
  <fl2_on_time of num>
  <fl2_off_time of num>
  <fl2_inc_time of num>
  <fl2_dec_time of num>
  <fl2_delay of num>
  <fl2_corr of num>
```

Continues on next page

Related information

	Described in:
Dispensing instructions	*DispL/DispC on page 25*
Definition of `beaddata`	*beaddata on page 36*
Customizing the functionality	*Customizing Dispense on page 85*
System module DPUSER	*System modules on page 45*

4 System modules

4.1 DPUSER

Introduction

The system module DPUSER contains predefined data and routines for the dispense application. The module is intended for the purpose of changing and customizing the behavior of the dispense functionality.

Data types

The following data types are predefined in DPUSER:

`beaddata`	*beaddata on page 36*
`equipdata`	*equipdata on page 39*

For `beaddata` and `equipdata` it is possible to reduce the data types by deleting data components that are not used. It is also possible to change the names of the components. See *Customizing Dispense on page 85*.

Note

If the definition of a data type is changed, the routine `dp_set_int_data` must also be changed. See *Routines on page 46*.

Global data

The following global data is predefined in DPUSER. The names of the described data are used internally when a DispL or DispC instruction is used. Therefore the names must not be changed.

Name	Declaration	Description
dp_fl1_corr	PERS num dp_fl1_corr:= 100	Global override for flow1 signal. All equipment are influenced. Range: 0-200%
dp_fl2_corr	PERS num dp_fl2_corr:= 100	Global override for flow2 signal. All equipment are influenced. Range: 0-200%
dp_dry	PERS bool dp_dry :=FALSE	Used for test in dry mode. If TRUE: No dispense signals are activated.
bead1	PERS beaddata *beaddata on page 36*	Predefined beaddata with default values. Used as default in the first programmed `DispL`/`DispC` instruction.
equipd	PERS equipdata *equipdata on page 39*	Predefined array of equipdata with equipdata for equipment 1 - 4. Used when `DispL`/`DispC` instructions are executed.
dp_fl_corr_en	CONSTdp_fl_corr_en:= TRUE	If this data is set to FALSE then the possibility to change the global flow correction factors from the Dispense MMI is disabled.

4.2 Routines

Description

There are some predefined routines installed with the application. They are all called from the dispense kernel and therefore the names must not be changed. Some of the routines have no default functionality.

Note

Many of the routines described below are executed during the motion phase. If there is too much time consuming code in these routines, the maximum possible dispensing speed will be reduced.

Data setup routine

PROC dp_set_int_data	This routine is used to connect user defined data component names to names used internally. The routine is called when a DispL or DispC instruction is executed. The routine must be changed if the definition of the dispensing data types is changed.

Flow calculation routines

FUNC dp_calcf1_type1	This routine is used to calculate the flow1 logical value when flow type = 1.
FUNC dp_calcf1_type2	This routine is used to calculate the flow1 logical value when flow type = 2.
FUNC dp_calcf2_type1	This routine is used to calculate the flow2 logical value when flow type = 1.
FUNC dp_calcf2_type2	This routine is used to calculate the flow2 logical value when flow type = 2.

For more information about the flow calculations, see *Calculation of flow values on page 31*.

Event routines

PROC dp_err_actions	This routine is executed when an error detected by the Dispense occurs.
PROC dp_power_on	This routine is executed each time the system is switched on.
PROC dp_start	The routine is executed each time the execution of a program is started from the beginning. (From Main)
PROC dp_restart	The routine is executed each time the execution of a stopped program is continued.
PROC dp_restart_proc	The routine is executed during restart of an interrupted dispense sequence, just before the process signals are activated.
PROC dp_stop	This routine is executed when a normal stop is performed during program execution.
PROC dp_qstop	This routine is executed when an emergency stop is performed during program execution.

5 System parameters

5.1 Topic I/O

Introduction

The Dispense package can be configured to suit different types of process equipment. This chapter describes the default defined signals used by Dispense and their dependency on the equipment. This signal configuration is enough for running the default version of the Dispense package.

Dispense signals for four equipment are predefined. During installation the predefined signals that are used must be connected to the corresponding physical equipment. It is not required to use all signals.

Note

The digital gun on/off signals for each equipment are internally activated as a signal group. This group must be connected to the physical signal (or signals) for the gun and the signals must follow the rules for group signals, see *Operating manual - OmniCore*.

Output signals controlling dispensing equipment 1

Signal	Type	Description
doEqu1Gun1	digital output	Signal to open/close gun 1.
doEqu1Gun2	digital output	Signal to open/close gun 2 (if used).
doEqu1Gun3	digital output	Signal to open/close gun 3 (if used).
doEqu1Gun4	digital output	Signal to open/close gun 4 (if used).
doEqu1Gun5	digital output	Signal to open/close gun 5 (if used).
goEqu1Guns	output group	Group used to activate the digital gun signals for equipment 1. Internally activated. This output group must be connected to the physical signals for the used guns.
doEqu1Active	digital output	A high signal indicates that the dispensing process is active.
doEqu1Err	digital output	This signal is activated when an internal dispense error is generated.
doEqu1OvrSpd	digital output	A high signal indicates that the calculated value for one analog output signal exceeds the logical maximum value.
aoEqu1F1	analog output	Analog signal for flow1.
aoEqu1F2	analog output	Analog signal for flow2.
goEqu1SwitchNo	output group	This group can be used to set a program number or switch point number to the dispense equipment, see *Customizing Dispense on page 85*.

Output signals controlling dispensing equipment 2-4

Same as above but the equipment numbers are changed in the signal names.

Continues on next page

Other signals

Signal	Type	Description
doBwdOnPath	digital output	Used for the bwd on path function if restart type 1 is activated and dp_bwd_key > 0. This signal is cross connected to the digital input *diBwdOnPath*.
diBwdOnPath	digital input	Used for the bwd on path function if restart type 1 is activated and dp_bwd_key > 0. Internally used.

Analog output scaling

The analog output signals above are defined with the following default values:

- Logical Min: 0
- Logical Max: 100
- Physical Min: 0 V
- Physical Max: 10 V

Normally, you only need to adapt the Physical Min and Max to the equipment in use. See *Calculation of flow values on page 31*.

5.2 Topic Process

5.2.1 Type Dispense GUI

5.2.1.1 The Dispense GUI type

Overview

This section describes the type *Dispense GUI* which belongs to the topic *Process*. Each parameter of this type is described in a separate information topic in this section.

Cfg name

DISPENSE_GUI

Type description

The *Dispense GUI* type contains a number of parameters that defines the characteristics for the graphical user interface. There is one set of parameters of the type *Dispense GUI* for each equipment.

5.2.1.2 Equipment

Parent

Equipment belongs to the type *Dispense GUI*, in the topic *Process*.

Cfg name

name

Description

Equipment is used to define the equipment name in the system parameters. To change the equipment name that is visible to the user, see *Equipment name on page 52*.

Limitations

The parameter is visible but not editable in the software configuration tools.

It is not allowed to change the name from the configuration file.

Allowed values

Equipment_1

Equipment_2

Equipment_3

Equipment_4

5.2.1.3 Equipment visible

Parent

Equipment visible belongs to the type *Dispense GUI*, in the topic *Process*.

Cfg name

equipment_visible

Description

Equipment visible defines if the equipment should be visible from the graphical user interface.

Usage

Set the value to No to hide the equipment in the graphical user interface.

Allowed values

Yes or No.

Default value is Yes.

5.2.1.4 Equipment name

Parent

Equipment name belongs to the type *Dispense GUI*, in the topic *Process*.

Cfg name

equipment_name

Description

Equipment name defines the equipment name in the graphical user interface.

Allowed values

A string with maximum 80 characters.

5.2.1.5 Flow1 visible

Parent

Flow1 visible belongs to the type *Dispense GUI*, in the topic *Process*.

Cfg name

flow1_visible

Description

Flow1 visible defines if the flow meter 1 should be visible from the graphical user interface.

Allowed values

Yes or No.

Default value is Yes.

5.2.1.6 Flow1 name

Parent

Flow1 name belongs to the type *Dispense GUI*, in the topic *Process*.

Cfg name

flow1_name

Description

Flow1 name defines the name of flow meter 1 in the graphical user interface.

Allowed values

A string with maximum 80 characters.

5.2.1.7 Flow2 visible

Parent

Flow2 visible belongs to the type *Dispense GUI*, in the topic *Process*.

Cfg name

flow2_visible

Description

Flow2 visible defines if the flow meter 2 should be visible from the graphical user interface.

Allowed values

Yes or No.

Default value is Yes.

5.2.1.8 Flow2 name

Parent

Flow2 name belongs to the type *Dispense GUI*, in the topic *Process*.

Cfg name

flow2_name

Description

Flow2 name defines the name of flow meter 2 in the graphical user interface.

Allowed values

A string with maximum 80 characters.

5.2.2 Type Dispense Restart

5.2.2.1 The Dispense Restart type

Overview

This section describes the type *Dispense Restart* which belongs to the topic *Process*. Each parameter of this type is described in a separate information topic in this section.

Cfg name

DISPENSE_RESTART

Type description

The *Dispense Restart* type contains a number of parameters that defines the characteristics for restarting an interrupted dispense process. There is one set of parameters of the type *Dispense Restart* for each equipment.

Related information

For more information, see *Using the Dispense Restart function on page 23*.

5.2.2.2 Equipment

Parent

Equipment belongs to the type *Dispense Restart*, in the topic *Process*.

Cfg name

name

Description

Equipment is used to define the equipment name in the system parameters. To change the equipment name that is visible to the user, see *Equipment name on page 52*.

Limitations

The parameter is visible but not editable in the software configuration tools.

It is not allowed to change the name from the configuration file.

Allowed values

Equipment_1

Equipment_2

Equipment_3

Equipment_4

5.2.2.3 Restart type

Parent

Restart type belongs to the type *Dispense Restart*, in the topic *Process*.

Cfg name

restart_type

Description

Restart type is used to set the restart behavior for the equipment. It is possible to set three different restart levels for each equipment.

0	Restart is disconnected. (Default). The process is started in the next bead, i.e. in the next `Disp\On` instruction.
1	Restart of current bead with possibilities to select wet or dry.
2	Restart of current bead in a no query mode (without questions)

Allowed values

A value between 0 and 2.

The default value is 0.

5.2.2.4 Gun on anticipate

Parent

Gun on anticipate belongs to the type *Dispense Restart*, in the topic *Process*.

Cfg name

gun_on_anticipate

Description

Gun on anticipate is used to set the preaction time for the digital gun signal related to the start of motion.

Allowed values

A value between -1 and 1, specifying the preaction time in seconds.

The default value is 0.

5.2.2.5 Flow anticipate

Parent

Flow anticipate belongs to the type *Dispense Restart*, in the topic *Process*.

Cfg name

flow_anticipate

Description

Flow anticipate is used to set the preflow time for the analog signals related to the start of motion.

Allowed values

A value between 0 and 1, specifying the preflow time in seconds.

The default value is 0.5.

5.2.2.6 Preflow value flow1

Parent

Preflow value flow1 belongs to the type *Dispense Restart*, in the topic *Process*.

Cfg name

preflow_value_flow1

Description

Preflow value flow1 is used to set the preflow value for the analog signal 1.

This value is used when the signal is TCP speed proportional and activated before start of motion.

The value is a percentage value related to the logical max value for the used signal.

Allowed values

A value between 0 and 100%.

The default value is 50%.

5.2.2.7 Preflow value flow2

Parent

Preflow value flow2 belongs to the type *Dispense Restart*, in the topic *Process*.

Cfg name

preflow_value_flow2

Description

Preflow value flow2 is used to set the preflow value for the analog signal 2.

This value is used when the signal is TCP speed proportional and activated before start of motion.

The value is a percentage value related to the logical max value for the used signal.

Allowed values

A value between 0 and 100%.

The default value is 50%.

5.2.2.8 Backward key

Parent

Backward key belongs to the type *Dispense Restart*, in the topic *Process*.

Cfg name

bwd_key

Description

Backward key is used to activate a programmable key for the backward on path function.

Allowed values

A value between 0 and 4.

The default value is 0 (not used).

5.2.2.9 Backward distance after stop

Parent

Backward distance after stop belongs to the type *Dispense Restart*, in the topic *Process*.

Cfg name

bwd_distance_after_stop

Description

Backward distance after stop is used to set the backward distance before restart after a program stop.

Allowed values

A value between 0 and 100, specifying the backward distance in mm.

The default value is 0.

5.2.2.10 Backward distance after halt

Parent

Backward distance after halt belongs to the type *Dispense Restart*, in the topic *Process*.

Cfg name

bwd_distance_after_qstop

Description

Backward distance after halt is used to set the backward distance before restart after a halt with motors off.

Allowed values

A value between 0 and 100, specifying the backward distance in mm.

The default value is 0.

5.2.3 Type Dispense Signals

5.2.3.1 The Dispense Signals type

Overview

This section describes the type *Dispense Signals* which belongs to the topic *Process*. Each parameter of this type is described in a separate information topic in this section.

Cfg name

DISPENSE_SIGNALS

Type description

The *Dispense Signals* type contains a number of parameters that defines which signals, configured in topic I/O, that are used as dispense signals. There is one set of parameters of the type *Dispense Signals* for each equipment.

5.2.3.2 Equipment

Parent

Equipment belongs to the type *Dispense Signals*, in the topic *Process*.

Cfg name

name

Description

Equipment is used to define the equipment name in the system parameters. To change the equipment name that is visible to the user, see *Equipment name on page 52*.

Limitations

The parameter is visible but not editable in the software configuration tools.

It is not allowed to change the name from the configuration file.

Allowed values

Equipment_1

Equipment_2

Equipment_3

Equipment_4

5.2.3.3 Flow1

Parent

Flow1 belongs to the type *Dispense Signals*, in the topic *Process*.

Cfg name

ao_flow1

Description

Analog signal for flow 1.

Allowed values

All configured analog output signals.

5.2.3.4 Flow2

Parent

Flow2 belongs to the type *Dispense Signals*, in the topic *Process*.

Cfg name

ao_flow2

Description

Analog signal for flow 2.

Allowed values

All configured analog output signals.

5.2.3.5 OnOff

Parent

OnOff belongs to the type *Dispense Signals*, in the topic *Process*.

Cfg name

go_onoff

Description

Signal group used to activate the digital gun signal or signals.

Allowed values

All configured digital output signal groups.

5.2.3.6 Dispense active

Parent

Dispense active belongs to the type *Dispense Signals*, in the topic *Process*.

Cfg name

do_active

Description

A digital signal that indicates that the dispensing process is active.

Allowed values

All configured digital output signals.

5.2.3.7 Dispense error

Parent

Dispense error belongs to the type *Dispense Signals*, in the topic *Process*.

Cfg name

do_error

Description

A digital signal that indicates that an internal dispense error is generated.

Allowed values

All configured digital output signals.

5.2.3.8 Overspeed

Parent

Overspeed belongs to the type *Dispense Signals*, in the topic *Process*.

Cfg name

do_overspeed

Description

A digital signal that indicates that the calculated value for one of the analog flow signals reached it's maximum value.

Allowed values

All configured digital output signals.

5.2.3.9 Switch value

Parent

Switch value belongs to the type *Dispense Signals*, in the topic *Process*.

Cfg name

go_switch

Description

Signal group that can be used to set a program number or switch point number to the dispense equipment.

For more information, see *How to use an extra signal group with bead information to the process equipment on page 89*.

Allowed values

All configured digital output signal groups.

This page is intentionally left blank

6 FlexPendant Interface

6.1 Application overview

Introduction

The Dispense operator interface (in this chapter referred to as *Dispense GUI*), is available as a web app from the FlexPendant home screen. It is a graphical interface addressing dispense users, designed to fulfill their specific needs. Dispense related information is presented in an instructing way, enabling operators to easily and quickly get their everyday tasks done.

The FlexPendant offers considerable flexibility regarding how to do things. It needs to be pointed out that advanced dispense users will have to use other FlexPendant applications and/or RobotStudio to perform certain tasks. For further information on using the FlexPendant, see *Operating manual - OmniCore*.

The views

The Dispense GUI contains the following primary views:

- **Main**
- **Configuration**
- **Signal**
- **Global flow correction**

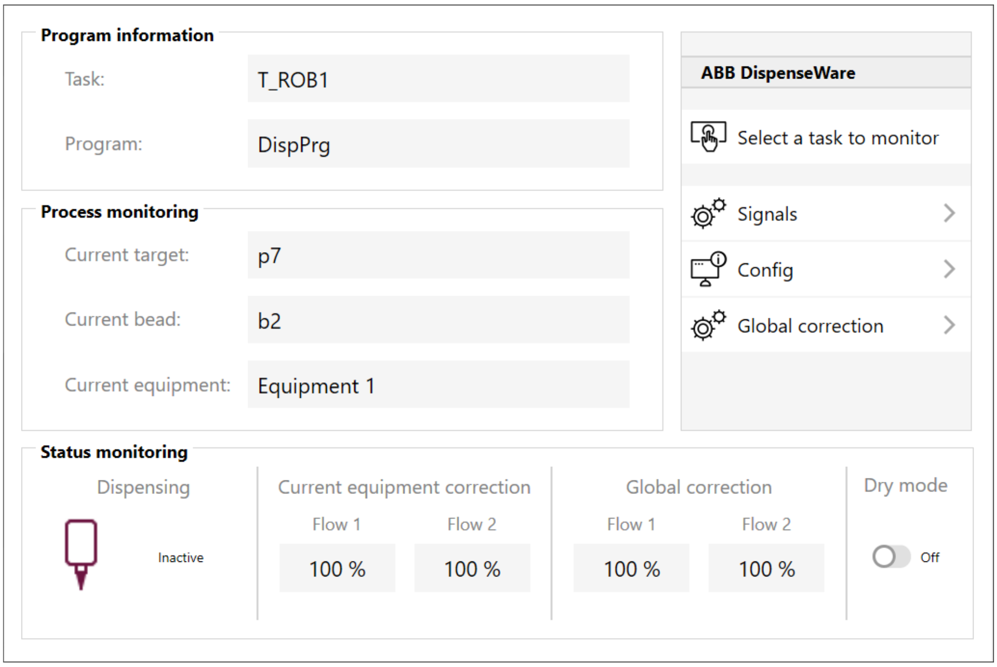
All views (except for **Main**) provide navigation for going back to the previous view, in the upper left corner.

6.2 Main view

Introduction

The **Main** view provides information and states about the currently executing RAPID dispense program. In addition, it provides possibilities to reach other views and sub views.

The **Main** view is the first view to appear when starting the application. It consists of the sections **Program information**, **Process monitoring**, and **Status monitoring**, and a menu section.



Program information

Task: T_ROB1

Program: DispPrg

Process monitoring

Current target: p7

Current bead: b2

Current equipment: Equipment 1

Status monitoring

Dispensing: Inactive

Current equipment correction		Global correction	
Flow 1	Flow 2	Flow 1	Flow 2
100 %	100 %	100 %	100 %

Dry mode: Off

xx2300001825

Program information

In the **Program information** section, the current program name and RAPID task name is displayed. The RAPID task can be changed at any time from the menu section.

Process monitoring

In the **Process monitoring** section, the current bead target name, bead data name, and equipment name are displayed. These values are continuously updated, as the RAPID dispense program executes.

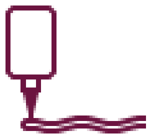
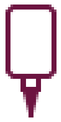
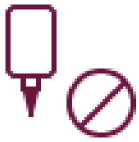
Continues on next page

Status monitoring

In the **Status monitoring** section several process statuses are displayed, such as dispensing status (active, inactive, dry mode), equipment-specific flow rate correction factors, global flow rate correction factor, and dry mode.

Dispensing status

The currently executing dispense instruction is illustrated by a dispense gun icon. When the RAPID dispense program reaches a dispense instruction, the **Active** icon is displayed. When program execution reaches a `DispL\OFF` instruction, the icon will shift to the **Inactive** icon. If the system is running in simulated mode, the **Dry mode** icon will appear and remain during program execution, as dispensing is never physically activated.

	On	The gun is dispensing.
	Off	The robot has stopped or is moving to a position where dispensing will start. The gun is not dispensing.
	Dry mode	Dry mode/simulated mode, dispensing functionality is blocked. The RAPID variable `dp_dry` (defined in RAPID module `DPUSER.sys`) is set. No physical dispensing will occur.

Flow corrections status

This section displays if any flow corrections are active or not, equipment-specific and global corrections.

The equipment-specific flow correction cannot be modified from the Dispense GUI but can be edited by modifying equipment data records in RAPID.

The global flow correction shows correction for flow 1 and flow 2, regardless of equipment. The RAPID variables `dp_fl1_corr` and `dp_fl2_corr` (defined in RAPID module `DPUSER.sys`) can also be modified through the Dispense GUI.

The possibility to change the global flow corrections from the Dispense GUI can in turn be enabled/disabled by setting RAPID variable `dp_fl_corr_en` (defined in RAPID module `DPUSER.sys`).

Dry mode

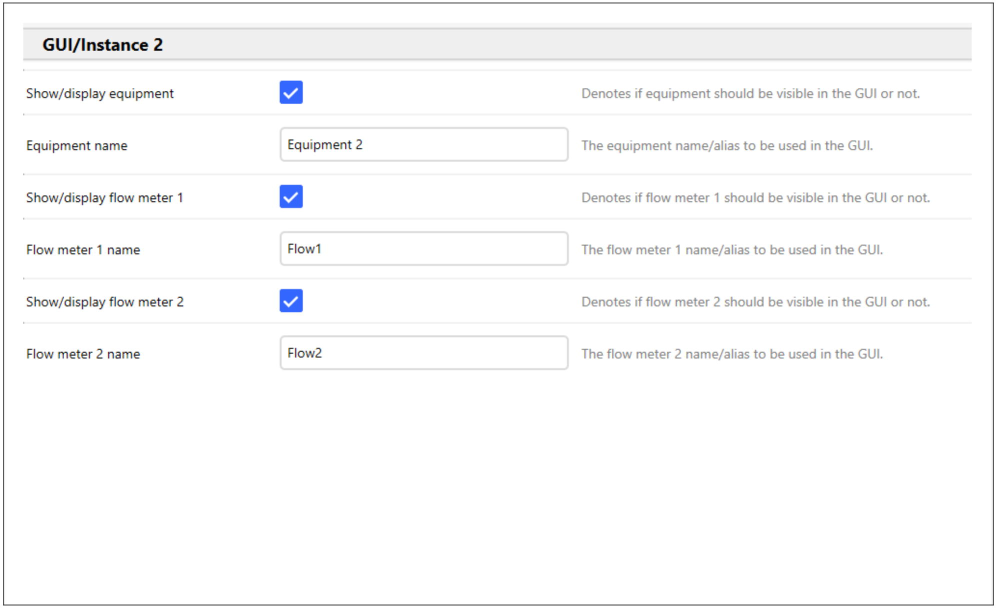
The dry mode can be toggled (on/off), when the controller is in manual mode. If dry mode is activated, no dispensing signals will trigger in program execution, and no physical dispensing will occur.

6.3 Configuration view

Introduction

The **Configuration** view is used to configure process configuration parameters for DispenseWare. Configurable types include GUI-specific configuration (*Dispense GUI*), restart behavior when dispensing (*Dispense Restart*), and custom signal naming (*Dispense Signals*).

The **Configuration** view has three sub-views, to navigating between process configuration types, or between configuration instances (one for each equipment, 1 to 4), or to edit the configuration parameters.



GUI/Instance 2

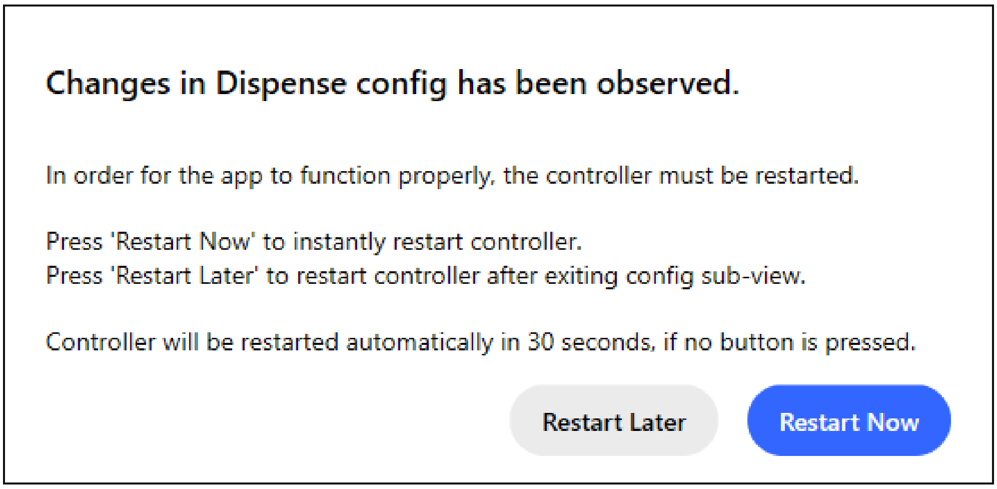
Show/display equipment	☑	Denotes if equipment should be visible in the GUI or not.
Equipment name	Equipment 2	The equipment name/alias to be used in the GUI.
Show/display flow meter 1	☑	Denotes if flow meter 1 should be visible in the GUI or not.
Flow meter 1 name	Flow1	The flow meter 1 name/alias to be used in the GUI.
Show/display flow meter 2	☑	Denotes if flow meter 2 should be visible in the GUI or not.
Flow meter 2 name	Flow2	The flow meter 2 name/alias to be used in the GUI.

xx2300001829

Continues on next page

App refresh/restart

The DispenseWare configuration can change, either triggered from the DispenseWare GUI, or from an external client (such as RobotStudio). When this happens the Dispense GUI may become unsynchronized with the updated DispenseWare configuration. If this happens, a warning is displayed that the application is running in an unsynchronized state, and recommended actions will be suggested. The recommended action can be to restart the application, reload configuration parameters, or to restart the robot controller.



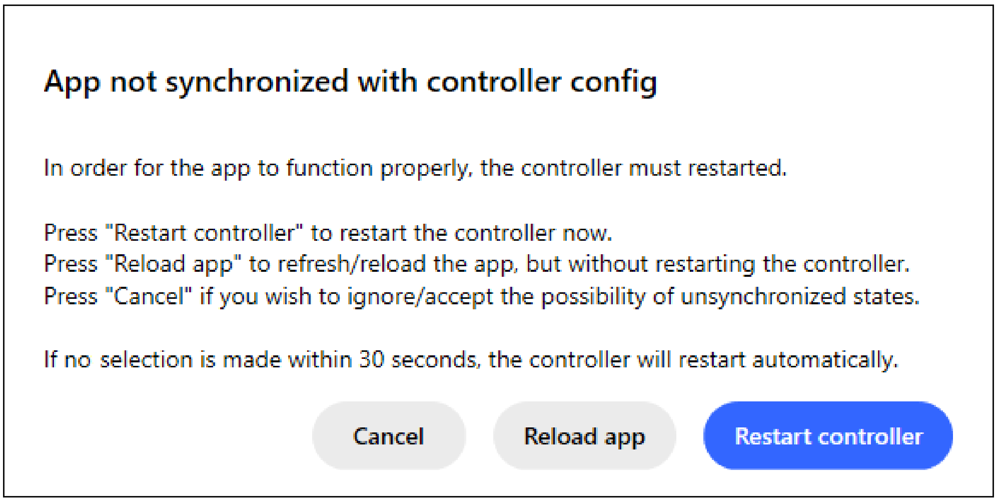
Changes in Dispense config has been observed.

In order for the app to function properly, the controller must be restarted.

Press 'Restart Now' to instantly restart controller.
Press 'Restart Later' to restart controller after exiting config sub-view.

Controller will be restarted automatically in 30 seconds, if no button is pressed.

xx2300001830



App not synchronized with controller config

In order for the app to function properly, the controller must restarted.

Press "Restart controller" to restart the controller now.
Press "Reload app" to refresh/reload the app, but without restarting the controller.
Press "Cancel" if you wish to ignore/accept the possibility of unsynchronized states.

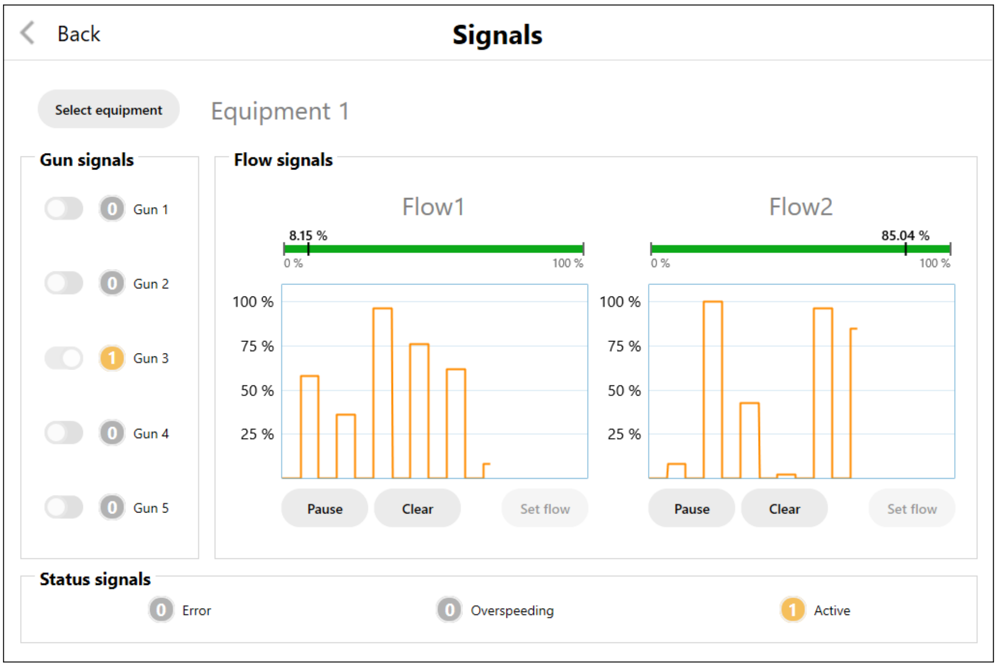
If no selection is made within 30 seconds, the controller will restart automatically.

xx2300001831

6.4 Signal view

Introduction

The **Signal** view displays the output signals controlling the dispense process equipment. Digital and analog signals are shown in an illustrative way, reflecting the real equipment of the specific dispense system in use. When testing the system in manual mode, it is possible to view and to activate the process signals. In other words, this view has capabilities to be used for both signal overview, and to some extent, signal testing and debugging.



xx2300001832

Equipment selection

Tap **Select equipment** to view signals for another equipment. The signals in the view reflects the selected equipment, which can be changed at any time. Note that equipment visibility in this view can be edited in the **Configuration** view.

Gun signals

The **Gun signals** section displays the gun signals for the selected equipment. In manual operating mode, the gun signals can be toggled. This can be useful when testing/debugging a dispensing system. In any other operating mode, the gun signals simply act as signal value preview, and the signals change state according to the executing RAPID program.

Flow signals

The **Flow signals** section shows the current value of flow 1 and 2, and line charts plotting the flow values over elapsed time.

Tap **Pause** to pause/resume the plotting of points.

Tap **Clear** to remove all plotted points.

Continues on next page

Tap **Set flow** to manually set a flow value (can only be set in manual operation mode). Note that the line charts plotting does not provide a perfect reliable real time representation of signal value over time due to natural delays. The charts are intended to visualize differences in flow over time.

Status signals

The **Status signals** section displays three important status signal interfaces.

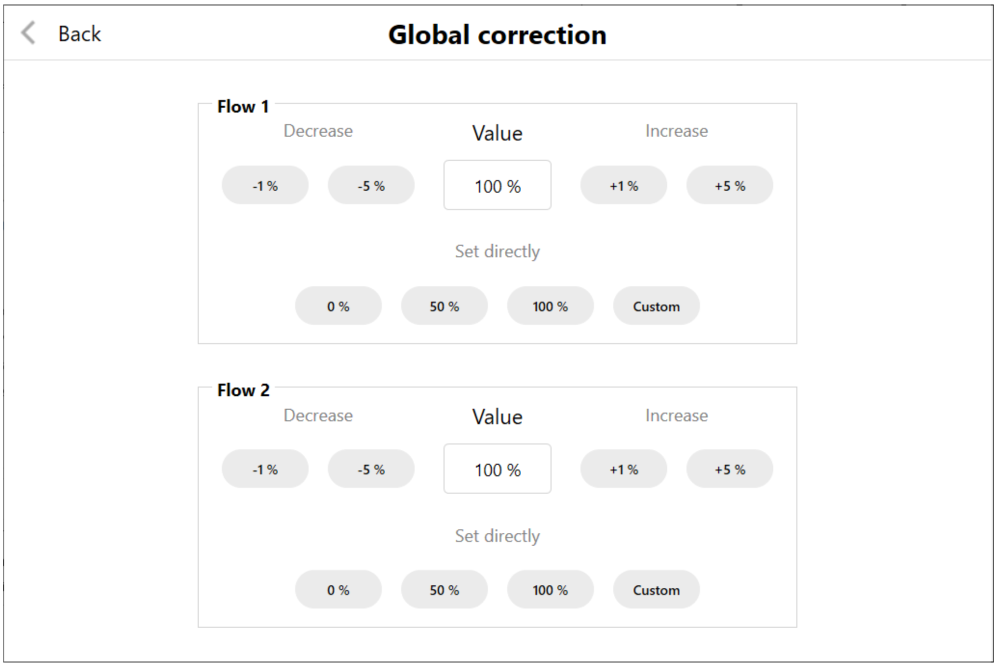
Signal interface name	Included signals	Situations when activated
Error	*doEqu1Err* *doEqu2Err* *doEqu3Err* *doEqu4Err*	An internal dispense error occurs when selected equipment is active. The program may have been stopped in the middle of a bead for example.
Overspeeding	*doEqu1OvrSpd* etcetera	The calculated value for a TCP speed dependent flow signal exceeds the maximum value, which means that the robot speed is temporarily too high to produce the programmed bead.
Active	*doEqu1Active* etcetera	Selected equipment is dispensing.

6.5 Global flow correction view

Introduction

The **Global flow correction** view is used for setting/overriding global flow rate correction factors for flow 1 and flow 2. This is useful when flow needs to be equally mended for both flow 1 and 2 over the whole dispensing process. Each flow correction can be set using either incremental buttons, direct values buttons, or custom value button (setting custom value using a keypad).

Note that this view only can be accessed while the controller is in manual operating mode, and when the RAPID variable `dp_fl_corr_en` (defined in the RAPID module `DPUSER.sys`) is enabled.



xx2300001833

7 Customizing Dispense

7.1 Introduction

Customizing possibilities

The Dispense functionality can be customized to adapt to different dispensing equipment. One of the purposes of this customizing process is to reduce the amount of data and number of variables presented to the operator.

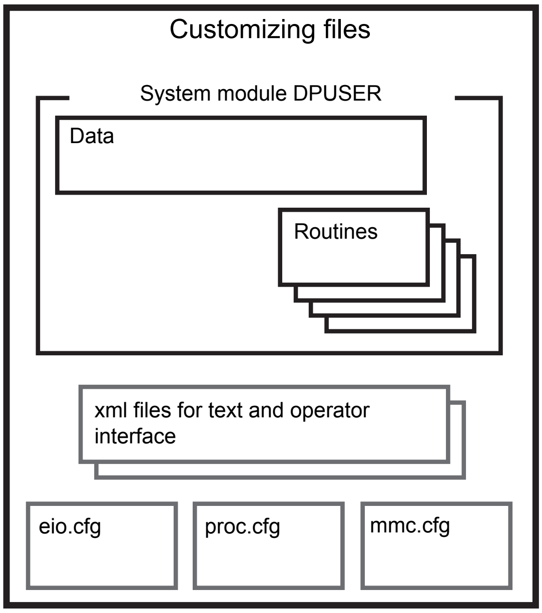
The following customizing is described in this manual:

- *How to redefine the data types beaddata and equipdata on page 88*.
- *How to use your own signal names for internal dispense signals on page 88*.
- *How to add functionality in the process sequence on page 88*.
- *How to create other equipment specific programming instructions on page 89*.
- *How to create service routines for manual functions on page 89*.
- *How to use an extra signal group with bead information to the process equipment on page 89*.
- *How to make it possible to program the process speed in beaddata on page 90*.
- *How to make it possible to program a z offset in beaddata. on page 91*.
- *How to change the flow scale calculation algorithm on page 91*.
- *How to customize the graphical user interface for Dispense on page 92*.
- *How to reduce the number of handled and visible equipment on page 92*.

7.2 Files to be changed during the customizing

Description

The customizing can be done by changing a number of predefined files. The following RAPID module and configuration files can be changed when customizing DispenseWare.



xx1400001081

DPUSER

DPUSER contains global data and routines affecting the entire program. The module also contains the definition of the dispensing data types, see *System modules on page 45*.

During installation, and when using the restart mode **Reset system**, the default version of DPUSER is copied to the directory $/HOME/Dispense in the system. This file is then used during start. If desired, this file can be exchanged to a customized version before start.

I/O configuration (eio.cfg)

As default, dispensing output signals are defined for four pieces of equipment but all signals are simulated. The used dispense outputs must be connected to physical signals. Signals not used must as simulated signals (see *Topic I/O on page 47*).

MMC configuration (mmc.cfg)

This configuration file contains for example information about which instructions are included in the different instruction pick lists.

Continues on next page

Process configuration (proc.cfg)

For dispense the process parameters are divided into three types:

Type	Description
Dispense GUI	Parameters for customizing the GUI for each equipment.
Dispense Restart	Parameters to set the start behavior for each equipment.
Dispense Signals	Parameters for connecting the dispense signals for each equipment.

7.3 Customizing guides

How to redefine the data types beaddata and equipdata

It is possible to:

- Add or delete data components.
- Move data components from for example `equipdata` to `beaddata`.
- Change the names of the data components.

Change the definition of the data types in DPUSER to desired.	
Change corresponding instructions in the data setup routine `dp_set_int_data` in DPUSER.	This routine is used to connect the user defined data components to internal data.

Change the structure and the default values of following data in DPUSER (if corresponding data type is changed):	`- PERS beaddata bead1`
	`- PERS equipdata equipd{4}`

Change text strings in gld_elogtext.xml if necessary.	

How to use your own signal names for internal dispense signals

It is possible to change the names of the predefined dispense outputs and output groups.

Change the signal names in the I/O configuration file.	
Change the corresponding signal in the Process configuration file.	

How to add functionality in the process sequence

It is possible to add own code to the different routines in DPUSER. Following routine is executed each time a dispense instruction is executed.

dp_set_intdata	Will be executed in the beginning of each Dispense instruction. The main task for this routine is to move the process data to the kernel, but in some cases it can be appropriate to add some user code to this routine.

Note

The routine is executed during the motion phase. If there is too much time consuming code in this routine, the maximum possible dispensing speed will be reduced. See description of the event routines in *System modules on page 45*.

When code is added to `dp_set_intdata` it is sometimes desirable to use information about current instruction. To get information following data can be used:

current_bead	beaddata	currently used beaddata
int_dp_on	bool	TRUE if `/On` argument in current instruction.
int_dp_off	bool	TRUE if `/Off` argument in current instruction.
int_dp_finep	bool	TRUE if fine point in current instruction.

Continues on next page

int_dp_posname	string	The name of the currently used `robtarget`. If the position is stored in the instruction (*) the string is empty.

Example: To print log information about current instruction (`robtarget` name and corresponding `beaddata`) in a logfile, the following instruction can be added to `dp_set_intdata`:

```
Write logfile," position: "+int_dp_posname+" beaddata:
     "+Argname(current_bead)+" ";
```

Note

The time when the routine `dp_set_int_data` is running is not coordinated with the robot movement. The routine is normally executed before the robot is passing the programmed position before current `Disp` instruction.

If it is important to have a more coordinated routine this can be done by creating a trap routine. Use for example the signal `doEqu1Active` for the activation, if the routine shall be executed in the beginning or end of the bead.

How to create other equipment specific programming instructions

Create global routines with desired name, syntax and functionality. Use a system module with the attribute NOVIEW or NOSTEPIN.

The MMC configuration has to be changed to define the instruction syntax and to get the new instructions in an instruction pick list.

How to create service routines for manual functions

Create RAPID routines for **Manual Functions** like for example *Move to Purge*, *Move to Home*, *Purge*, *Reload*...

The MMC configuration has to be changed. Add the routine names under `MMC_SERV_ROUT_STRUCT`. These functions are then available from the **Service Routines** menu in the **Operate** app.

How to use an extra signal group with bead information to the process equipment

The dispense kernel is prepared for activation of an extra signal group a user defined time before the robot is passing a programmed `Disp\On` or `Disp` instruction. This signal group can for example be used to give switch point information or program number information to the process equipment. The group is cleared in `Disp\Off` instructions.

If this functionality is used it is suitable to add a new data component (for example `switch_number`) to `beaddata` and an other data component (for example `switch_time`) to `equipdata`. See *How to redefine the data types beaddata and equipdata on page 88*.

In the DPUSER routine dp_set_int_data the internal data int_dp_data.switch_no and int_dp_data.switch_time has to be updated with information from current `beaddata` and `equipdata`.

Continues on next page

An output group for each equipment (for example goEqu1SwitchNo) is already predefined in the default eio.cfg, and handled in the *Dispense Signals* type in the proc.cfg.

How to make it possible to program the process speed in beaddata

It is possible to increase the beaddata with a speed data component. This speed data will then be used during the process instead of using the speed data programmed in the dispense instruction.

To be able to use this possibility, add a new data component (for example tcp_speed) to the beaddata RECORD in DPUSER. See *How to redefine the data types beaddata and equipdata on page 88*.

In the DPUSER routine dp_set_int_data the internal data int_dp_data.tcp_speed has to be updated with information from current beaddata, in a similar way as other data components from beaddata are updated.

For example add following instruction:

```
int_dp_data.tcp_speed := current_bead.tcp_speed;
```

Add the new data component to all already programmed beaddata to be used. Remember to add the new component also to the predefined beaddata *bead1* in DPUSER.

More details:

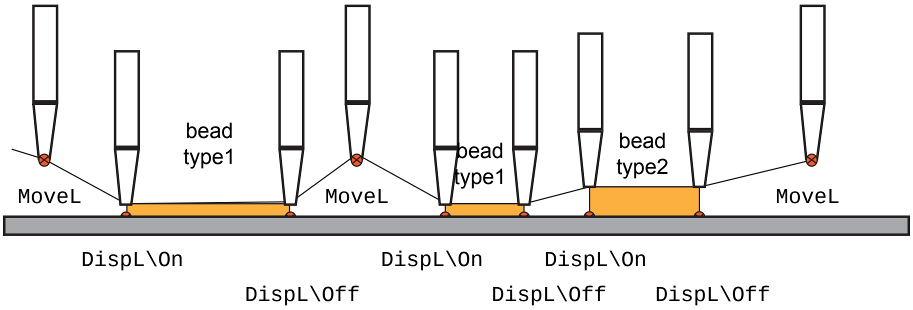
- When the program is executed step wise forward or backward the programmed speed in the instruction is used.
- To reduce unnecessary accelerations or decelerations during the process, the speed data from current beaddata will be used also during the DispL\On or DispC\On segment, before the process is started.
- All speed changes will be affected from the middle of the corner zone, also when the optional argument \D is used. It is for that reason recommended to not use the optional instruction argument \D when the speed is programmed in beaddata.
- The speed from current beaddata is also used during test in dry mode.
- If the speed in the beaddata is set to -1 this function is disabled. The programmed speed in the instruction will be used in the normal way.

Continues on next page

How to make it possible to program a z offset in beaddata.

It is possible to increase the `beaddata` with a z offset component. This offset data will then be used during the process as shown in example below.

Example:



xx1200000640

During programming all dispense instructions are programmed with the TCP on the sheet or with a constant distance to the sheet. During program execution the tool is automatically lifted (in negative z direction) to correct level for the bead according to the offset value in current `beaddata`.

To be able to use this possibility, add a new data component (for example `z_offset`) to the `beaddata` RECORD in DPUSER. See *How to redefine the data types beaddata and equipdata on page 88*.

In the DPUSER routine dp_set_int_data the internal data int_dp_data.z_offset has to be updated with information from current `beaddata`, in a similar way as other data components from `beaddata` are updated.

If the name of the new component is z_offset, add following instruction:

```
int_dp_data.z_offset := current_bead.z_offset;
```

Add the new data component to all already programmed `beaddata` to be used. Remember to add the new component also to the predefined `beaddata` *bead1* in DPUSER.

More details:

- When dispense positions are changed/modified this shall be done without offset as during programming. Therefore the z offset is omitted when the program is executed step wise forward or backward in Manual mode.
- Minimum and and maximum permissible offset is internally limilted to 0-20 mm. It is however possible to change these limits using following prepared DPUSER data.

```
CONST num dp_z_offset_max := xx;
CONST num dp_z_offset_min := xx;
```

How to change the flow scale calculation algorithm

Change the predefined calculation algorithm in following event routines in DPUSER:

dp_calcf1_type1:	Used to calculate the flow1 logical value when flow type = 1
dp_calcf1_type2:	Used to calculate the flow1 logical value when flow type = 2
dp_calcf2_type1:	Used to calculate the flow2 logical value when flow type = 1

Continues on next page

dp_calcf2_type2:	Used to calculate the flow2 logical value when flow type = 2

How to customize the graphical user interface for Dispense

It is possible to change the names for the equipment and flow signals. It is also possible to change the number of visible digital and analog gun signals.

The **Process Signals** view should mirror the real dispense equipment in use. It is possible to customize the GUI by editing the system parameters in type *Dispense GUI* which belongs to the topic *Process*.

The predefined parameters should be edited to reflect the real equipment. This improves both system performance and ease of use since the FlexPendant only displays information that is relevant for a specific installation.

How to reduce the number of handled and visible equipment

By default four equipment are visible in data and configuration. This is also the maximum number of equipment to be handled. If less than four equipment is used, it is possible to reduce the number of handled and visible equipment.

	Action
1	Reduce the number of elements in the `equipd` array in `DPUSER`. The dispense kernel handles the same number of equipment as the size of this array.
2	Delete all I/O signals for the unused equipment in the system parameters, topic *I/O*.
3	Delete all system parameters for the unused equipment of the types *Dispense GUI*, *Dispense Restart*, and *Dispense Signals*, which belongs to the topic *Process*.

Index

ABB AB
Robotics & Discrete Automation
S-721 68 VÄSTERÅS, Sweden
Telephone +46 10-732 50 00

ABB AS
Robotics & Discrete Automation
Nordlysvegen 7, N-4340 BRYNE, Norway
Box 265, N-4349 BRYNE, Norway
Telephone: +47 22 87 2000

ABB Engineering (Shanghai) Ltd.
Robotics & Discrete Automation
No. 4528 Kangxin Highway
PuDong New District
SHANGHAI 201319, China
Telephone: +86 21 6105 6666

ABB Inc.
Robotics & Discrete Automation
1250 Brown Road
Auburn Hills, MI 48326
USA
Telephone: +1 248 391 9000

abb.com/robotics