Electric Drives and Controls | Hydraulics | Linear Motion and Assembly Technologies | Pneumatics | Service

Rexroth Bosch Group

Rexroth IndraMotion for Plastics Software - Open Designable Cycle

R911334673
Edition 01

Application Manual

Title
Rexroth IndraMotion for Plastics
Software - Open Designable Cycle

Type of Documentation
Application Manual

Document Typecode
DOK-IM*PL*-ODC*PROG***-AP01-EN-P

Internal File Reference
RS-280e9707243e8ea70a6846a0013c5b42-1-en-US-3

Purpose of Documentation
This documentation describes the components and their interaction when creating flexible machine procedures for injection molding machines using the Open Designable Cycle (ODC).

Record of Revision

Edition	Release Date	Notes
Edition 01	02.2011	First edition

Published by
Bosch Rexroth AG
Bgm.-Dr.-Nebel-Str. 2 ■ D-97816 Lohr a. Main, Germany
Phone+49 (0)93 52/ 40-0 ■ Fax +49 (0)93 52/ 40-48 85
http://www.boschrexroth.com/
System Development Automation Motion Logic Control, DaSc (FrWe/MePe)

Note
This document has been printed on chlorine-free bleached paper.

Table of Contents

Table of Contents

1 Introduction

1.1 Purpose of the Documentation

This documentation describes the components and how they interact in the creation of flexible machine sequences for injection molding machines when using the Open Designable Cycle (ODC).

1.2 Target Group, Prerequisites

This documentation is intended for the following target audience with the respective prerequisites:

Target group	Prerequisites on knowledge and ability
Programmer	• Technical training (University of Applied Science, engineering degree or corresponding professional experience). • Knowledge about: – the method of operation of a PLC, – safety instructions, – programming a PLC (IEC61131), – object-oriented programming (Java).

1.3 Documentation Notes

1.3.1 General

This manual is part of the product. It is to be retained over the entire life cycle of the product and should be forwarded to any subsequent owners or users of the product.

1.3.2 Documentation Contents

- General Product Description
- Explanation of the ODC components
- Installing and using the editors
- Language description
- Adapting the machine sequence
- Diagnosis possibilities
- Commissioning a modified machine sequence
- Errors and warnings that can occur on the control during operation

1.4 Additional Documentation

Doc. type code	Name	Target group
R911334666	Rexroth IndraMotion for Plastics Version 1.30VRS Graphical User Interface KVB	Machine setter Machine operator
R911334667	Rexroth IndraMotion for Plastics Version 1.30VRS Graphical User Interface KVS	Machine setter Machine operator

All manuals referred to here can be obtained at Bosch Rexroth.

2 Safety Instructions

2.1 Description

At various points in this manual you will see notes and precautionary warnings regarding possible hazards. The symbols used have the following meaning:

⚠ DANGER	• indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.
⚠ WARNING	• indicates a potentially hazardous situation which, if not avoided, can result in death or serious injury.
⚠ CAUTION	• indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury.
NOTICE	• CAUTION used without the safety alert symbol indicates a potentially hazardous situation which, if not avoided, may result in damage to property • This symbol also indicates you of the possible consequences of touching electrostatically sensitive components.

☞ Tips on usage and useful information are indicated with the "tip" symbol. Notes do not contain any information that draws attention to potentially dangerous or harmful functions.

3 General Product Description

3.1 General

The purpose of the Open Designable Cycle (ODC) is to configure variable movement sequences for injection molding machines without having to change the IEC application of the PLC (Programmable Logic Controller).

In this way, the base machine sequence supplied by Bosch Rexroth can be extended and modified with variable movement sequences using the editor (notepad++). This makes it possible to adapt the machine sequence to a variety of machine types (e.g. parallel movements, flexible cores, etc.).

3.2 Overview

3.2.1 General

The entire sequence or machine sequence consists of a series of movements that are performed in one production cycle of an injection molding machine. This is configured in the ODC using a base sequence file and one or more optional description files which have the file extension .odc.

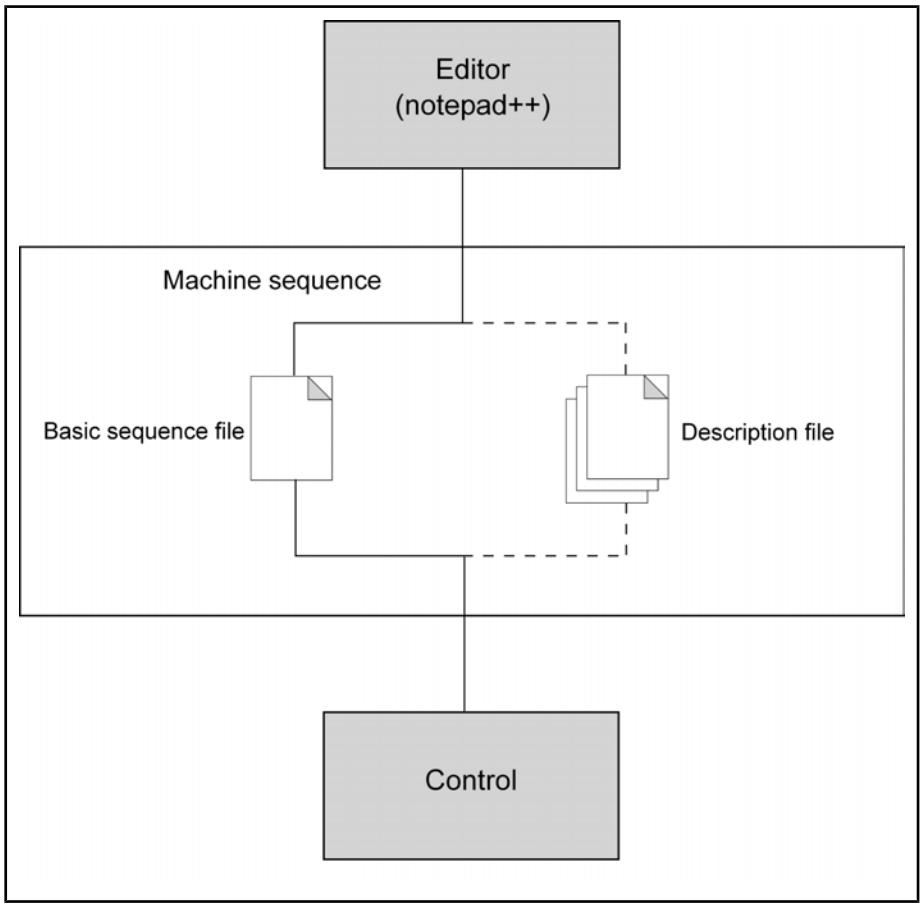


Fig.3-1: Overview of the ODC elements

Sequence files (base sequence and description files) are modified or created and checked with the editor supplied, notepad++ (see chapter 5.5 Syntax-Checker, page 25 and chapter PriorityChecker, page 28).

General Product Description

Then, the files must be copied to the control's Compact Flash to a defined directory, which is created based on these files when the machine sequence starts up.

3.2.2 Base Sequence File

The base sequence file describes the base machine sequence.

In the following example, the default base sequence supplied by Bosch Rexroth is shown:

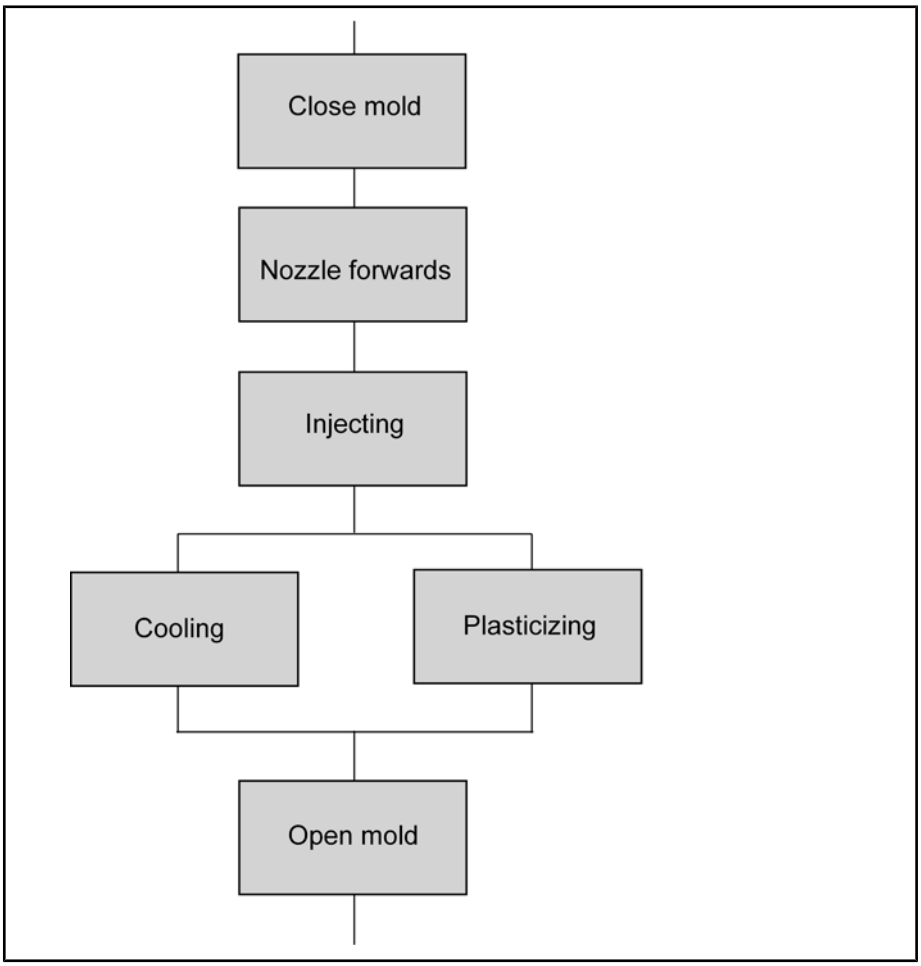


Fig.3-2: Example of a base sequence as a flow chart

3.2.3 Description File

With description files, additional movements or movement sequences that depend on program switches and their values can be added to the sequence.

Program Switch

A program switch consists of the system variables and switches in the visualization that affect the machine cycle. The ODC Builder checks the state of all program switches before the machine sequence is created. Changes to the program switches take effect when the next machine cycle begins.

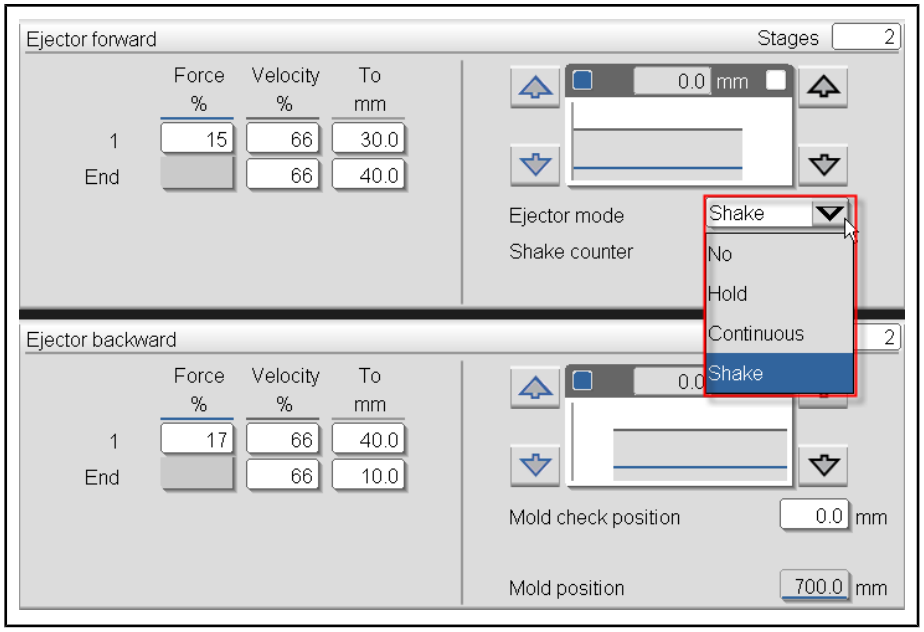


Fig.3-3: Example of program switches based on an "ejector"

It is recommended that for each program switch a description file be created, which specifies the effects of the various program switch positions on the sequence. Also refer to the chapter 3.2.3 Description File, page 8.

4 Editor

4.1 General

The editor notepad ++ is used to modify or create machine sequences.

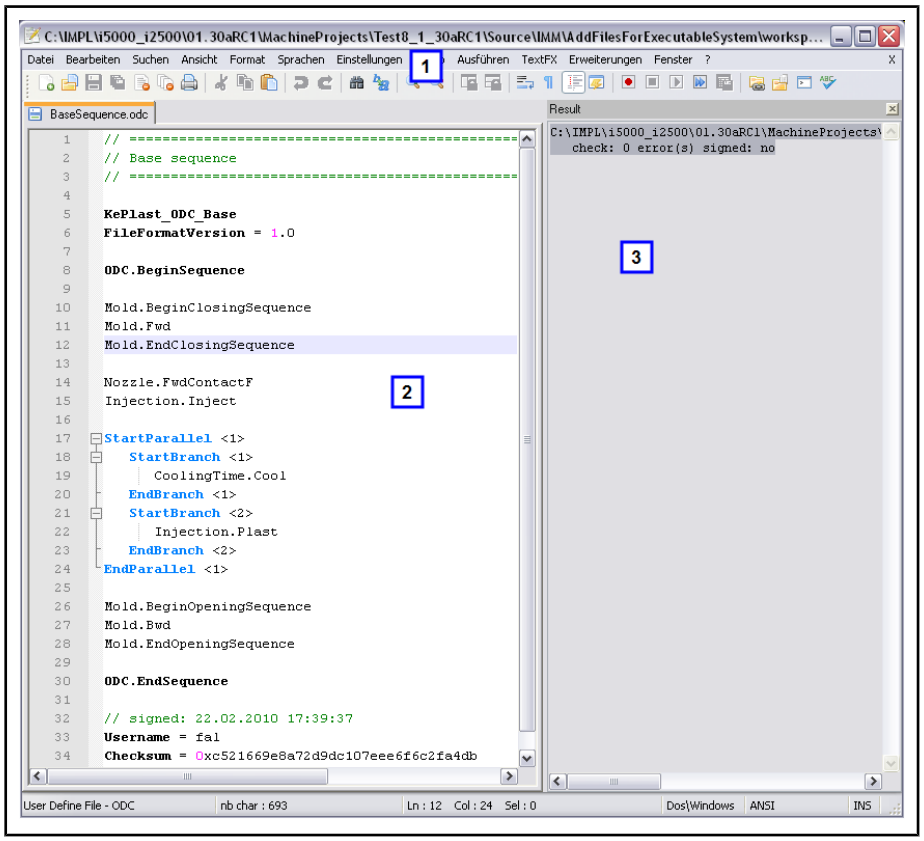
Using notepad++ as an editor is recommended because Bosch Rexroth supplies specialized ODC extensions for it.

4.2 Installation

IMPL sector-specific setup is used to install both the open source editor notepad ++, which is used for the ODC, and the related plug-in for checking ODC files.

Furthermore, files with the file extension .odc are automatically linked with notepad++.

4.3 Interface



1 ... Menu
2 ... Work area
3 ... Output window for diagnostic programs - optional
Fig.4-1: Interface for the "notepad++" editor with installed ODC plug-ins

The output window for diagnostic programs is only displayed after the first test with the SyntaxChecker or PriorityChecker (see chapter 5.5 SyntaxChecker, page 25 and chapter PriorityChecker, page 28) and by default it is deactivated. If can be moved freely within the program window.

Editor

4.4 ODC Plug-in

When the ODC plug-in is installed, notepad++ is extended to include SyntaxChecker and PriorityChecker to check ODC sequences. These programs are described in detail in the chapters 5.5 SyntaxChecker, page 25 and PriorityChecker, page 28.

SyntaxChecker and PriorityChecker can be executed via the menu **Extensions ▸ ODC**.

5 Creating/Adapting a Machine Sequence

5.1 General

To create or adapt a machine sequence, the following chapters provide information regarding

- Design and syntax of base sequence and description files,
- Commands for individual movements,
- Available program switches,
- SyntaxChecker and PriorityChecker syntax programs

.

All sequence files (both base sequence and description files) must be saved with the file extension "**.odc**". Only files with this ending are recognized by control as sequence files.

On the control's Compact Flash, the sequence files must be stored in the

`\\harddisk0\workspace\data\sequence\std`

directory.

Upon delivery, the default sequence supplied by Bosch Rexroth is found in this directory.

5.2 Base Sequence File

5.2.1 General

```
1  // ================================================
2  // Base sequence
3  // ================================================
4
5  KePlast_ODC_Base
6  FileFormatVersion = 1.0
7
8  ODC.BeginSequence
9
10 Mold.BeginClosingSequence
11 Mold.Fwd
12 Mold.EndClosingSequence
13
14 Injection.BeginInjectSequence
15 Nozzle.Fwd
16 Injection.Inject
17 Injection.EndInjectSequence
18
19 StartParallel <1>
20    StartBranch <1>
21       CoolingTime.Cool
22    EndBranch <1>
23    StartBranch <2>
24       Injection.BeginPlastSequence
25       Injection.Plast
26       Injection.EndPlastSequence
27    EndBranch <2>
28 EndParallel <1>
29
30 Mold.BeginOpeningSequence
31 Mold.Bwd
32 Mold.EndOpeningSequence
33
34 ODC.EndSequence
```

Fig.5-1: Example code for a base sequence file based on the default sequence

`KePlast_ODC_Base`	defines the file as a base sequence file. It is imperative that this element be located at the beginning of the file.
`FileFormatVersion`	indicates which version of ODC is used and must be increased when a more recent version of ODC is used. The FileFormatVersion used must be the same version or previous to the version supported by the package in current use.

`ODC.BeginSequence`	marks the beginning of the sequence description; ODC.EndSequence marks its end. All of the movements required for the machine sequence are between these elements (see chapter 5.2.3 Motions, page 16)
`StartParallel` and `EndParallel`	limit parallel branching (see chapter 5.2.2 Parallel Sequences, page 15).

The area between the element // and the respective end of the line is considered a comment and is not taken into consideration in the creation of the sequence.

A detailed listing of all of the keywords can be found in the chapter 5.4 Keywords, page 23.

5.2.2 Parallel Sequences

With the ODC, several movements can be executed in parallel.

```
1  Movement A
2  Movement B
3  StartParallel <1>
4    StartBranch <1>
5      StartParallel <2>
6        StartBranch <1>
7          Movement C
8          Movement D
9        EndBranch <1>
10       StartBranch <2>
11         Movement E
12         Movement F
13       EndBranch <2>
14     EndParallel <2>
15   EndBranch <1>
16   StartBranch <2>
17     Movement G
18   EndBranch <2>
19   Movement H
20 EndParallel <1>
```

Fig.5-2: Example code for parallel movements

Parallel movements can be located in both the base sequence file within the sequence description and in a description file within a movement block (also refer to chapter 5.2.3 Motions, page 16).

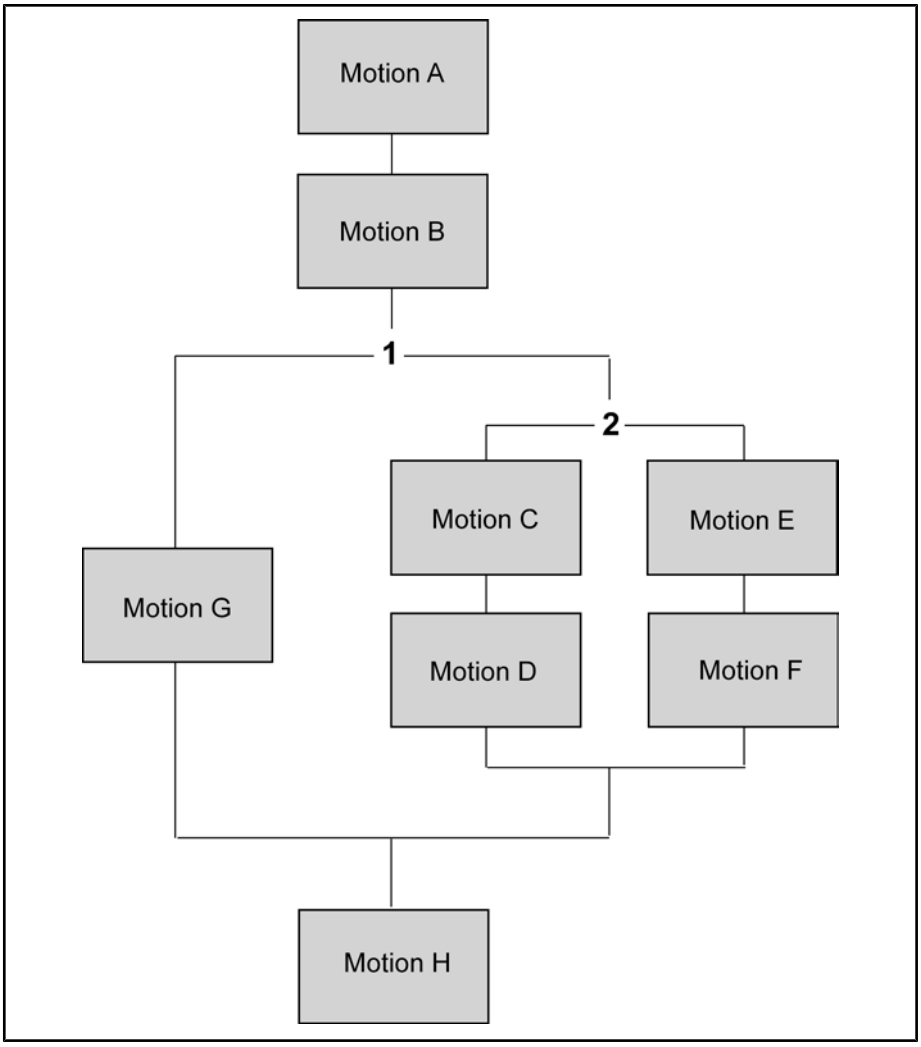
`StartParallel`	generates a branch; an index in pointed brackets is used for uniquely identifying the elements if there are multiple nested branches
`EndParallel`	brings the branches together again.

For the index <x>, only values from 1 to 6 are valid because the ODC allows a maximum of 6 parallel branches.

`StartBranch`	marks the beginning of the movement sequence within a branch.
`EndBranch`	The movement sequence comes to an end. An index is specified here as well.

The following figure shows the code above as a flow chart.

Creating/Adapting a Machine Sequence



1 Represents the beginning of the first parallel branch

2 In the right hand branch "2" indicates the second branch. The vertical lines within the branches each represent "one branch".

Fig.5-3: Flow chart to illustrate the example

☞ Parallel sequences can be used in the same way in both the base sequence file and in the description files.

5.2.3 Movements

Commands for executing movements consists of the following components:

- FU (Function Unit, e.g. Core)
- Index in pointed brackets (e.g. <1>). This index is used to discriminate among multiple function units (multiple cores, air valves, etc.). This specification is optional; if no index is specified, "<1>" is used by default.
- Movement ID (e.g. `.In`)

☞ Movements are used in the same way in both the base sequence file and the description files.

Example:

The command for entering Core1 is represented as follows:
`Core<1>.In`

☞ In a movement block, movements can only be inserted between the tags `{Sequ}` and `{Sequ_End}`.

The following FUs and motion IDs are available:

FU	Motion IDs	Description
AirValve	Start StartAndWait	Starts the air valve and in the sequence, the next movement is executed immediately. Starts the air valve and the next movement is executed, but only after the air valve "movement" is finished.
CoolingTime	Cool	Starts the cooling.
Core	In Out	Core in. Core out.
Ejector	Fwd Bwd Shake	Ejector forward. Ejector backward. Shake ejector.
Injection	Inject Plast Intrusion BeginInjectSequence EndInjectSequence BeginPlastSequence EndPlastSequence ColdDrop	Injection. Charging and decompressing. Intrusion. Start of injection sequence. End of injection sequence. Start of charging sequence. End of charging sequence. Time-dependent injection movement.
Mold	Fwd FwdInterPos Bwd BwdInterPos BeginClosingSequence EndClosingSequence BeginOpeningSequence EndOpeningSequence	Close mold. Close mold till intermediate position Open mold. Open mold till intermediate position Start of closing sequence. End of closing sequence. Start of opening sequence. End of opening sequence.
Nozzle	Fwd Bwd	Nozzle forward. Nozzle backward.

Creating/Adapting a Machine Sequence

The commands for starting and ending movement sequences (e.g. Mold.BeginClosingSequence) are required to perform these sequences individually with key presses in set-up and manual mode.

5.3 Description File

5.3.1 General

```
KePlast_ODC_Option
FileFormatVersion = 1.0

SelectorVar = Core<index>.sv_CoreOutMode

// ------------------------------------------
// part for before/after mold open
// ------------------------------------------
{Sequ}
   // movement to insert
   Core<index>.Out
{Sequ_End}

// insert conditions

// before mold open
{Insert}
   SelectorValue  = 1
   Insert         = Before Mold.Bwd
   InsertPriority = 20
{Insert_End}

// after mold open
{Insert}
   SelectorValue  = 3
   Insert         = After Mold.Bwd
   InsertPriority = 20
{Insert_End}
// ------------------------------------------

// ------------------------------------------
// part for during mold open
// ------------------------------------------
{Sequ}
   // movement to insert
   Mold.BwdInterPos
   Core<index>.Out
{Sequ_End}

// insert conditions

// during mold open
{Insert}
   SelectorValue     = 2
   Insert            = Before Mold.Bwd
   InsertPriority    = 40
   DominantMovement  = Core<index>.Out
{Insert_End}
```

Fig.5-4: Example of a description file based on the default file for Core x Out

`KePlast_ODC_Option`	defines the file as a description file. It is imperative that this element be located at the beginning of the file.
`SelectorVar`	specifies the name of the program switch (in this case, the ejector mode); see also 5.3.2 Program Switch, page 23.
`<index>`	is a placeholder, which is necessary here because there are several cores (`Core`) referenced using one index. For the `<index>` placeholder, the respective numeric value used in the base sequence file is used when the movement sequence is executed.
`{Sequ}`and `{Sequ_End}`	limit a movement block that is to be inserted at a certain position in the base sequence when a certain program switch position is used.

The insertion conditions for the preceding movement sequence are located between `{Insert}` and `{Insert_End}`.

The following insertion conditions must be entered between the elements `{Insert}` and `{Insert_End}`:

- `SelectorValue:` indicates the program switch position of `SelectorVar` for which this movement block is to be inserted.
- `Insert:` specifies the insertion mode (`Before, After`) and for the `Before` and `After` modes for a movement it specifies the location in the base sequence where the insertion will be performed.

  As an alternative to `Insert` the elements `ParallelBegin` and `ParallelEnd` can be used to insert a movement sequence. The keywords in this case define two movements in the base sequence at which the inserted parallel branch begins and ends.

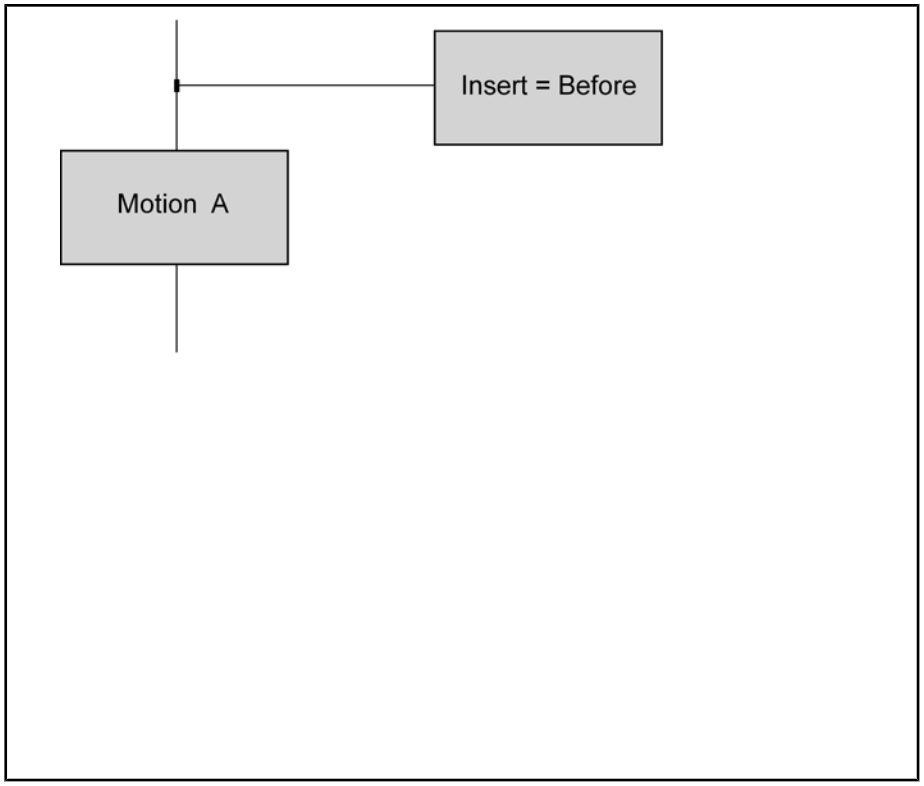


Fig.5-5: Before insertion mode

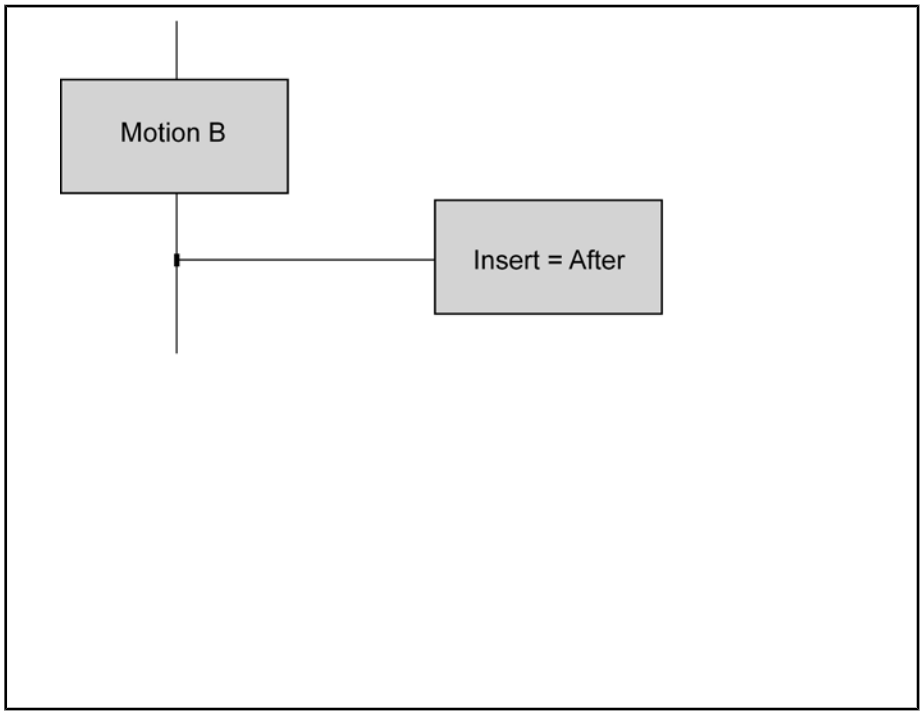


Fig.5-6: After insertion mode

Creating/Adapting a Machine Sequence

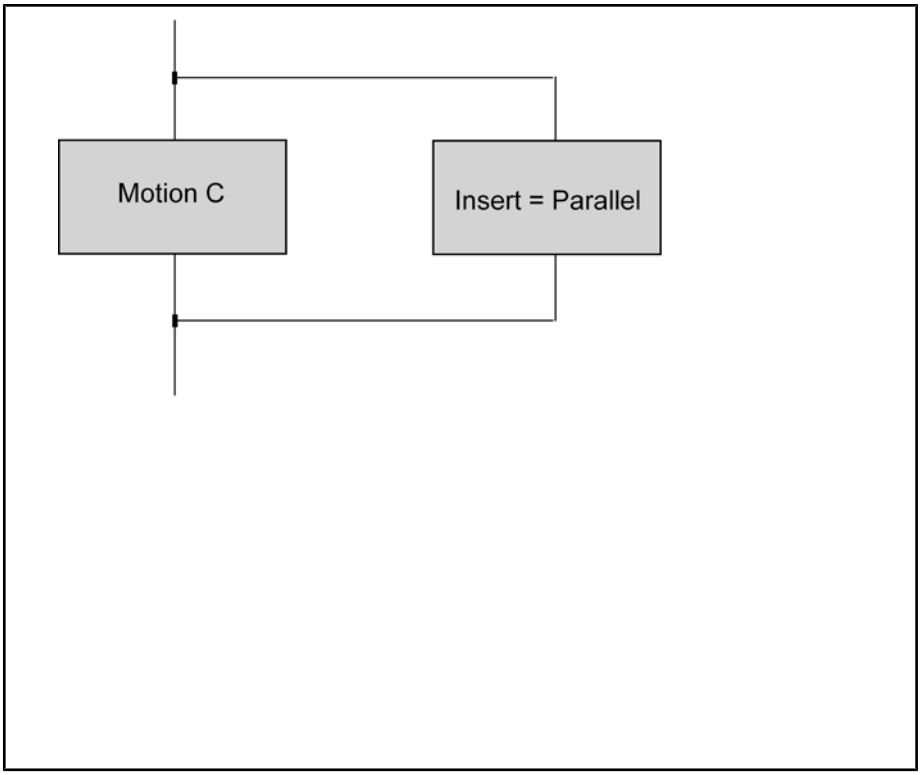


Fig.5-7: Parallel insertion mode

- `InsertPriority:` is used to set priorities if several movements are inserted at the same position. The greater the value of `InsertPriority`, the higher the priority.

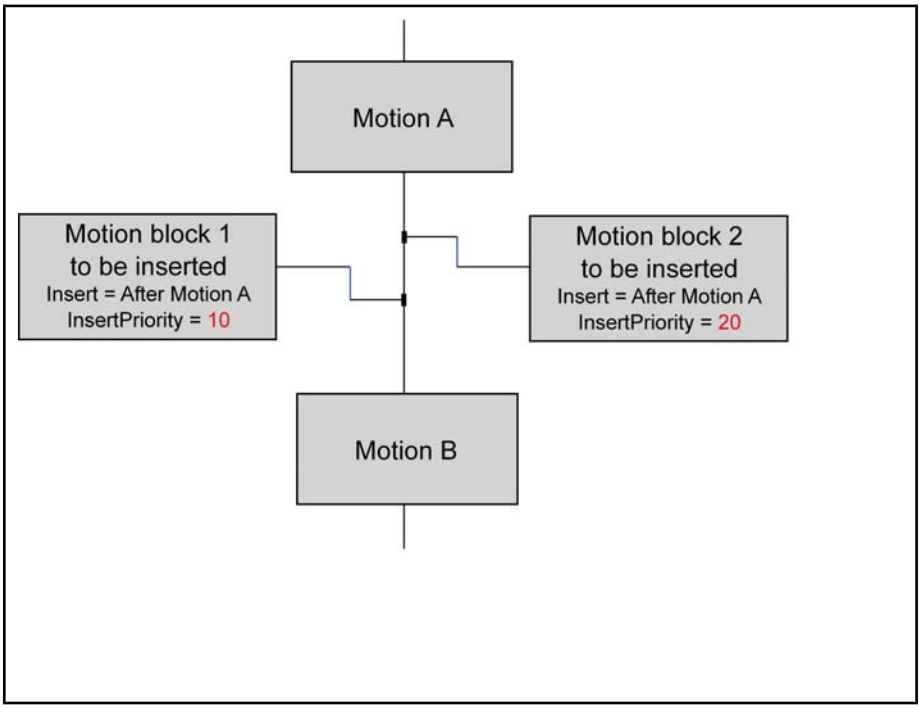


Fig.5-8: Graphical example for InsertPriority

Optional condition:

- `DominantMovement:` if there are several movements in movement blocks in different description files that are inserted at the same position in the base sequence, `DominantMovement` can be used to assign absolute priority to a certain movement.

If, depending on program switch position (`SelectorValue`), a movement block is to be inserted at another position or with another priority, insertion conditions for the movement block (for the respective program switch positions) can be used to specify multiple blocks.

5.3.2 Program Switch

This chapter includes a list of the program switches already in use in the default sequence. In principle, other IMPL application system variables can also be used as program switches.

Program switch	System variable name	Values
Air valve	`AirValve<index>.sv_AirValveMode`	0: Off 1: After loading 2: After mold open 3: After ejector 4: During mold open 5: Before loading
Core in	`Core<index>.sv_CoreInMode`	0: Before close 1: During close 2: After close
Core out	`Core<index>.sv_CoreOutMode`	1: Before open 2: During open 3: After open
Ejector mode	`Ejector<index>.sv_EjectorMode`	0: Off 1: Hold 2: Continuous 3: Shake
Intrusion	`Injection<index>.sv_IntrusionSettings.Mode`	0: Off 1: Time
Nozzle backward mode	`Nozzle<index>.sv_NozzleBwdMode`	0: Off 1: After plasticize 2: Before mold open 3: After inject

5.4 Keywords

Alphabetical list of all of the keywords used in ODC:

☞ If there is no entry in the "Used in" column, this keyword is used in both base sequence and description files.

Creating/Adapting a Machine Sequence

Keyword	Used in	Description
`//`		The area between `//` and the end of the line is considered a comment and is not taken into consideration in the creation of the sequence.
`{Insert} … {Insert_End}`	Description file	Block for insertion conditions; insertion conditions are specified between these keywords.
`{Sequ} … {Sequ_End}`	Description file	These elements define a movement block that can be inserted into the sequence.
`Checksum`		Checksums (unique recognition feature for the program code that has been checked). This element is automatically generated by the SyntaxChecker.
`DominantMovement`	Description file	Assigns unique priority to a movement within a movement block if several movement blocks with the same priority are inserted at the same position.
`FileFormatVersion`		Specifies the ODC file format version (required by ODC Builder).
`Insert`	Description file	Specifies the insert mode, i.e. if a movement block is to be inserted before (`Before`) or after (`Aftera` certain movement.
`InsertPriority`	Description file	This value is used to set priorities for various movement blocks that are to be inserted at the same position. It applies: The greater the value, the higher the priority.
`KePlast_ODC_Base`	Base sequence file	Specifies that this file is a base sequence file.
`KePlast_ODC_Option`	Description file	Specifies that this file is a description file.
`ODC.BeginSequence … ODC.EndSequence`	Base sequence file	These elements limit the sequence description in the base sequence file. Movements can only be located between these elements.
`ParallelBegin, ParallelEnd`	Description file	These keywords can be used as an alternative to Insert in order to insert a movement block parallel to base sequence movements. `ParallelBegin` defines the starting point of the parallel branch to be inserted; `ParallelEnd`defines the end. As with `Insert`, the keywords `Before` and `After` are used here to specify the exact position for insertion.
`SelectorValue`	Description file	Specifies the program switch position for which a movement sequence is to be inserted.
`SelectorVar`	Description file	Specifies the program switch upon which the insertion is dependent.
`StartBranch … EndBranch<x>`		The movement sequence of a parallel branch is specified between these elements.
`StartParallel … EndParallel<x>`		These elements start and end a parallel branch, i.e. multiple movements can be executed next to each other at the same time.
`Username`		User name of the user who signed a file This element is automatically generated by the SyntaxChecker.

5.5 SyntaxChecker

5.5.1 General

After completion, the SyntaxChecker must be used to check the `.odc` files for correct syntax. In addition, the SyntaxChecker adds a unique signature to the end of the checked files. This is used as another safety feature for execution on the control and for protection from undesired manipulation of the sequence files.

5.5.2 Signature

A signature consists of an name abbreviation and a checksum and ensures that the syntax of the sequence file is correct and that the file was not manipulated following the syntax check.

Example signature:

```
// signed: 01.08.2008 09:48:31
Username = user
Checksum = 0x500caaba59eac762da3610450aled735
```

Fig.5-9: Example of a signature

- If there is **no** signature present when a sequence file is loaded, a warning is output (for tracing purposes, an entry is also made in the Infolog), but the control still creates a movement sequence.
- If the **incorrect** signature is present when a sequence file is loaded, an error is output and the control cannot create a movement sequence.

  The incorrect signature is present if sequence file that has already been checked and signed was modified and no new check was performed afterward.

5.5.3 Checking an Individual File

Select the menu item **Extensions ▸ ODC ▸ Check and sign current file** to subsequently check the ODC file opened by notepad++ in the current tab.

Check with no errors: The signature is automatically inserted. If there is already a signature present and the file has not been changed, this existing signature is retained.

Creating/Adapting a Machine Sequence

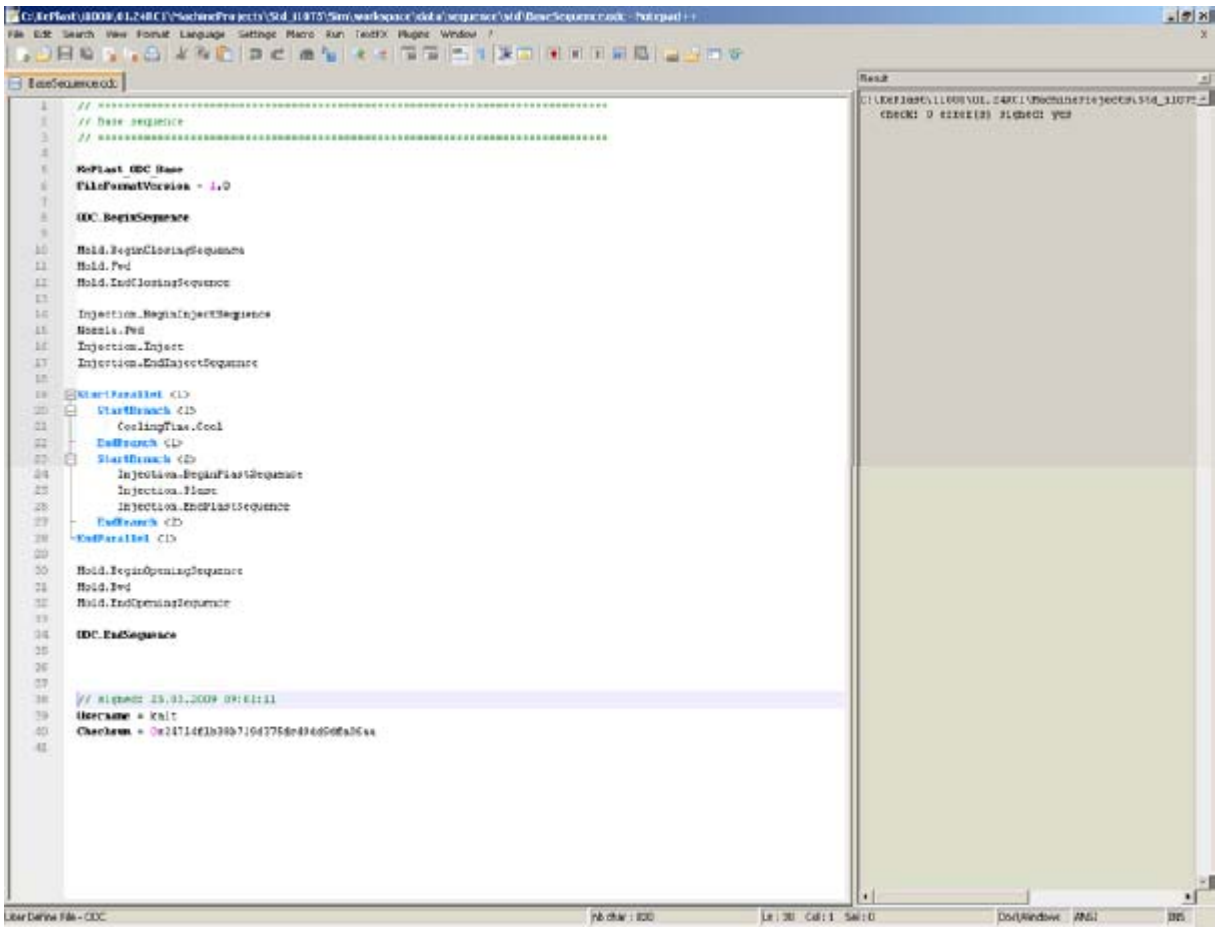


Fig.5-10: Screenshot of notepad++; syntax check of the selected file with no errors

Check yields syntax errors:

- A window is opened in which all of the errors recognized are displayed (type of error and line specification).
- In addition, the erroneous area in the sequence file is highlighted in red (error in the screenshot: StartsParalel instead of StartParallel)

Fig.5-11: Screenshot of notepad++; error found in the syntax check of the selected file

After the error has been corrected, a new check must be carried out in order to provide the file with a signature.

5.5.4 Checking all Files in a Specific Directory

Select the menu item **Extensions ▸ ODC ▸ Check and sign all files in current directory**to subsequently check all sequence files located in the directory of the currently opened sequence file.

Check with no errors: A signature (see also chapter 5.5.2 Signature, page 25) consisting of a name abbreviation and a checksum is appended to the end of all checked files.

Creating/Adapting a Machine Sequence

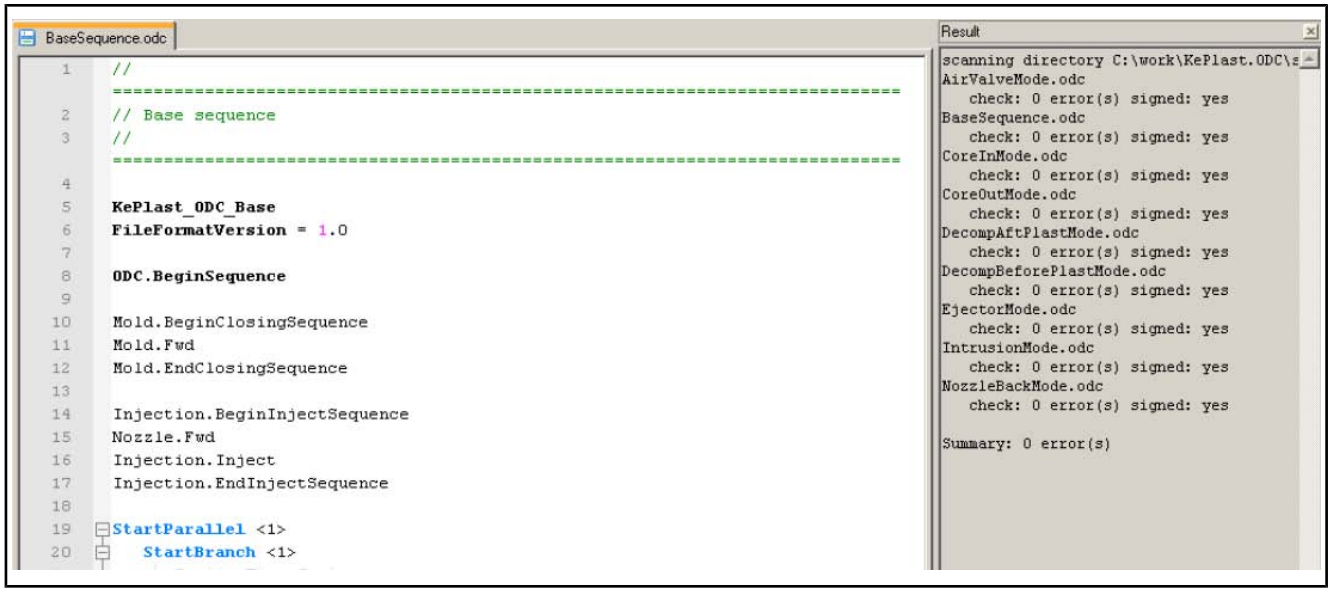


Fig.5-12: Screenshot of notepad++; syntax check of all of the files located in the directory; no errors found

Check yields syntax errors:

- A window is opened in which all of the errors recognized are displayed (type of error and line specification).
- All files that include errors are opened and displayed in order.
- In addition, all lines in the sequence files that include errors are highlighted in red.

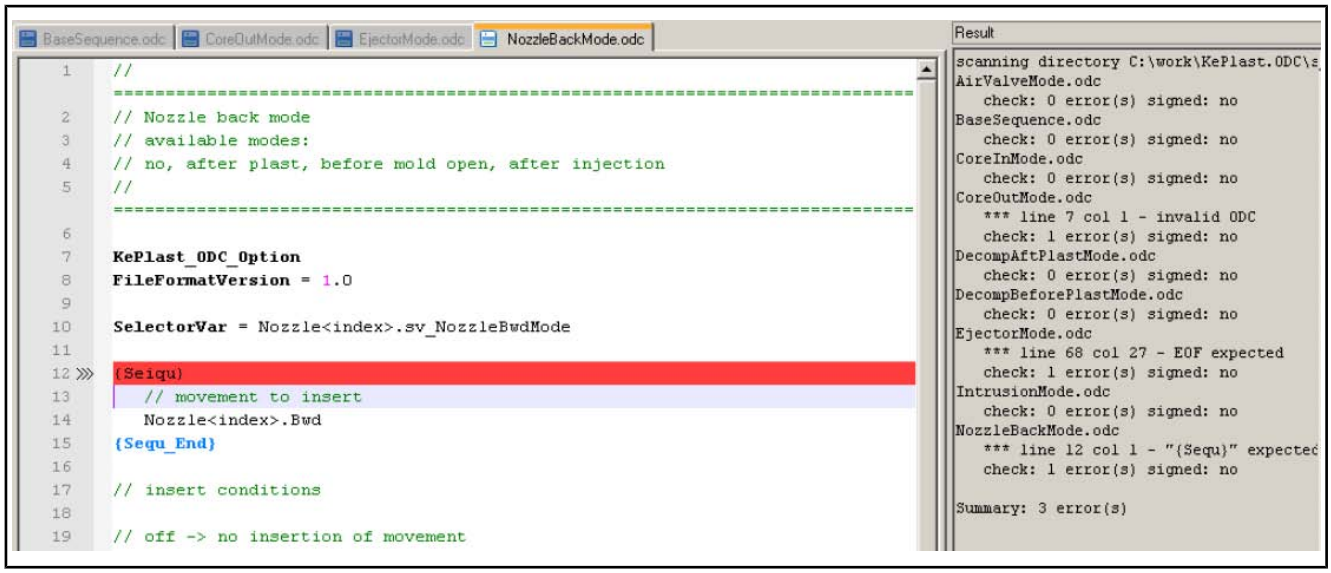


Fig.5-13: Screenshot of notepad++; at least one error found in the syntax check of the files

After the error is corrected, a new check must be carried out.

5.6 PriorityChecker

5.6.1 General

The PriorityChecker has two functions:

- Checking all sequence files in the current working directory for collisions among insertion priorities and
- Listing the priorities of all insertion conditions in the current working directory.

A prerequisite for using the PriorityChecker is correct syntax for all sequence files (.odc) in the directory of the file that is currently open. Otherwise the PriorityChecker aborts the check and outputs an error message.

5.6.2 Checking Priorities

Select the menu item **Extensions ▸ ODC ▸ Priority Check** to subsequently check all of the insertion conditions located in the directory of the currently open sequence file for collisions among insertion priorities.

Collisions occur if movements of two or more different FUs (components), each dependent on different program switches, can be inserted at the same position in the sequence. PriorityChecker checks all of the sequence files in a directory to see if this could occur.

Check did not yield any collisions:

PriorityChecker issues the short message "No priority collisions found".

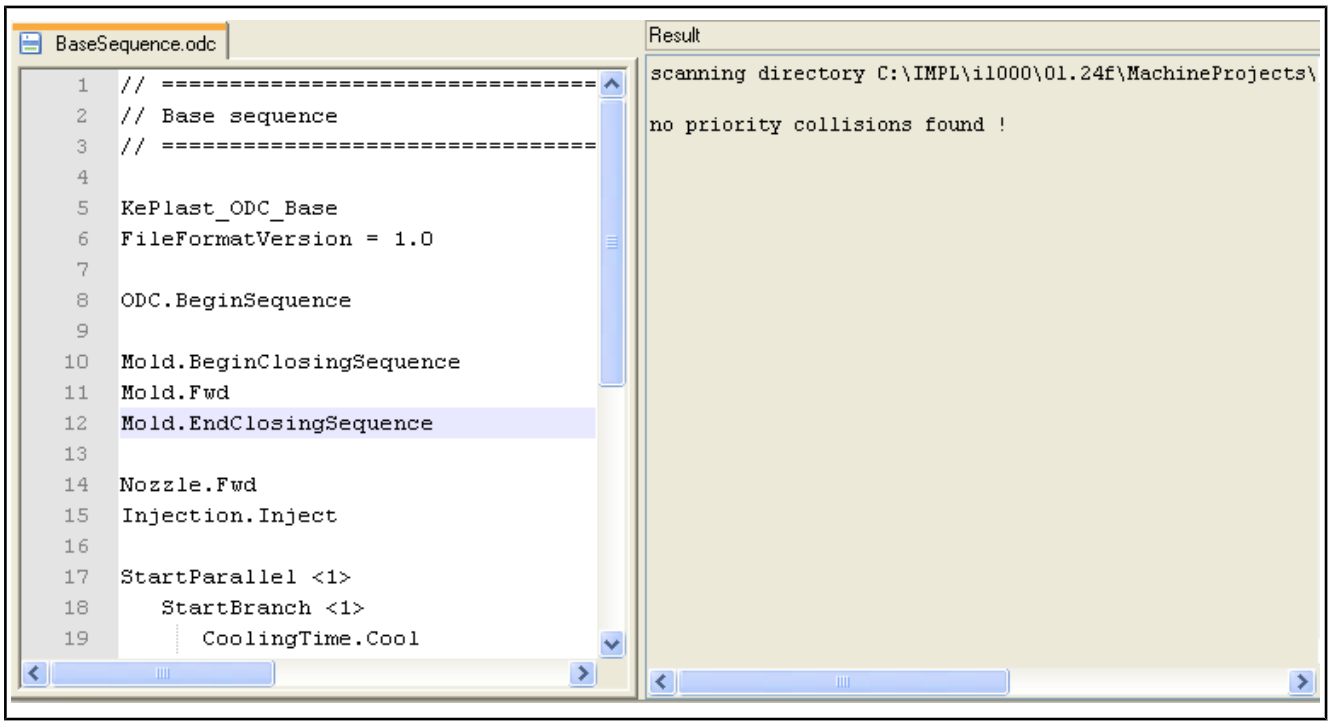


Fig.5-14: Screenshot of notepad++; PriorityCheck did not yield any collisions

Check yields collisions:

- A window opens, which displays all possible collisions (files, line numbers and insertion conditions of the insertion blocks that could collide).
- All of the files affected are opened and displayed in order.
- In addition, `{Insert}` elements for the affected insertion conditions are highlighted in red.

Creating/Adapting a Machine Sequence

Fig.5-15: Screenshot of notepad++; PriorityCheck found collisions

- Collisions can be removed by changing the priorities or by inserted the movement block at another position in the sequence.

5.6.3 Displaying All Priorities

Select the menu item **Extensions ▸ ODC ▸ Show all Priorities** to subsequently list all of the insertion priorities located in the directory of the currently open sequence file.

The following data are displayed for the insertion conditions:

- `InsertPriority`
- File in which the insertion condition is located
- `SelectorVar`
- `SelectorValue`
- `Insert`

The list is sorted according to decreasing priority or by file name.

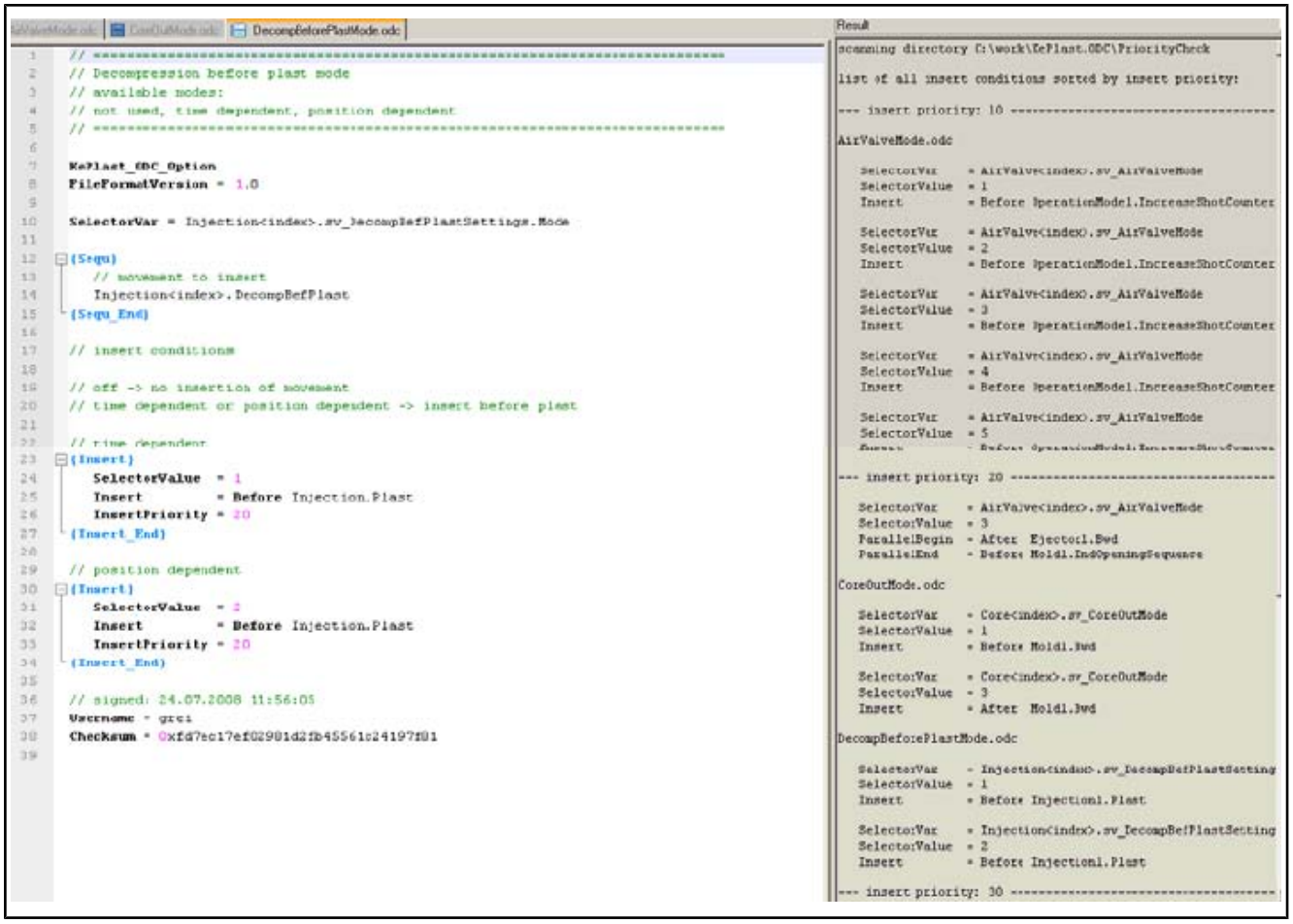


Fig.5-16: Screenshot of notepad++; Display all priorities

6 Commissioning a New Sequence

Load sequence files:

1. Switch off the machine.
2. Copy the new or modified files to the Compact Flash into the
   `\\harddisk0\workspace\data\sequence\std`
   directory.
3. Restart the control (changes to the sequence files only take effect after a restart).
4. Switch on the machine.
5. Set the respective program switch and test the sequence.
   The debug mask for the visualization you are using (KePlast.HMI.KVS or KePlast.HMI.KVB) offers support for testing.
   Now the machine sequence is ready to be used.

7 Errors and Warnings

7.1 General

This chapter contains descriptions of the errors and warnings that cannot be recognized with the SyntaxChecker and PriorityChecker. They are only recognized during the generation of the sequence or during runtime on the control.

Errors and warnings from the ODC are output in the visualization alarm mask.

Errors cause the sequence creation or the running machine sequence to be aborted immediately. Warnings can subsequently result in errors, but do not prevent the creation of the sequence or the sequence itself.

7.2 Errors

Alarm text	Description
BaseSequenceFile does not exist in <<Directory>>	No base sequence file could be found in the directory specified. Create a base sequence file or copy an existing base file into the directory specified.
ODCBuilder Parser Syntax Error: <<Filename>> <<Linenumber>> <<Characters>>	In the file specified, a syntax error was found at the specified position. This error can only occur in unsigned sequence files that yielded errors when previously checked with the SyntaxChecker. Open the file specified and repair the error.
ODCBuilder Parser Error <<Filename>> <<Linenumber>> <<Characters>>	In the file specified, an error that cannot be exactly defined was found at the position specified. Open the file and repair the error at the position specified.
ODCBuilder: fileformat version x.x not supported	The specified `FileFormatVersion` is not supported by the package installed on the control. Change your sequence files to a version supported by the package.
ODCBuilder: Error ODC.BeginSequence movement	In the base sequence file, a movement was inserted before the keyword `ODC.BeginSequence`. Open the base sequence file and make sure that no movements are located before the keyword.
ODCBuilder: Error ODC.EndSequence movement	In the base sequence file, a movement was inserted after the keyword `ODC.EndSequence`. Open the base sequence file and make sure that no movements are located after the keyword.
ODCBuilder: File <<Filename>> has an invalid signature	The signature for the specified file is incorrect. Check the file with the SyntaxChecker in order to create a correct signature.
ODCBuilder: baseSequence is already defined - <<Filename>>	Several base sequences are present. Make sure that only one base sequence is located in the directory.

Errors and Warnings

Alarm text	Description
ODCBuilder: cannot get value for selector variable <<Variable-name>>	The value of the specified program switch could not be read in. This is caused by an internal KePlast application software error. Restarting the control can help remedy the situation.
ODCBuilder: sysvar <<Variablename>> is not available	The specified program switch (system variable) could not be found. Check the ODC file; the name of the system variable was either written incorrectly or the system variable does not exist.
ODCBuilder: movement <<Movement>> not available in movement-list	The specified movement was not recognized. The movement was either written incorrectly or does not exist in the list of the available movements.
ODCBuilder: illegal dominant movement	The movement specified for `DominantMovement` could not be found in the related movement block. The movement was either incorrectly written or it does not exist in the movement block.
ODCBuilder: insert position (<<Movement>>) not found in base sequence	The movement specified (insertion position) could not be found in the base sequence file. Check both the base sequence file and the insertion condition in the description file.
ODCBuilder: insert of movement before ODC.BeginSequence not allowed	A movement is supposed to be inserted before the keyword `ODC.BeginSequence`; this is not permitted. Check the insertion conditions in the description files.
ODCBuilder: insert of movement after ODC.EndSequence not allowed	A movement is supposed to be inserted after the keyword `ODC.EndSequence`; this is not permitted. Check the insertion conditions in the description files.
ODCBuilder: invalid InsertParallel sequence (insert position: <<Movement>>)	Parallel insertion of a movement block at the specified position is not possible. Check the description files.
ODCBuilder Parser Exception <<Description>>	An internal error has occurred. More detailed information can be found in the description of the exception.
ODCBuilder cannot create logfile <<File>>	The specified logfile could not be created. Check the configuration.
ODCBuilder: insertpriorities and instancepriorities are equal <<Movement1>> <<Movement2>>	Both of the movements specified are to be inserted at the same position and have the same insertion and instance priorities. For this reason, the ODCBuilder cannot decide which movement to insert first. Check the description files.
ODCBuilder: instance priority for <<Movement>> necessary	No instance priority is available for the specified movement, which was specified as a `DominantMovement`. Check the description file.
ODCBuilder: internal Error <<Errormessage>>	An internal error has occurred. More detailed information can be found in the error message specified.

Alarm text	Description
ODCBuilder: invalid call of CreateSequence – init error <<Errormessage>> occurred	The sequence could not be created because an error occurred during initialization. More detailed information can be found in the error message. Check the sequence files.
ODCBuilder: sequence array (<<Height>> <<Width>>) is too small (<<Line>> <<Column>>)	The application movement array is too small. This error must be repaired by adjusting the movement array in the KePlast application.
ODCBuilder: sequence array element (<<Line>> <<Column>>) is already used	Internal error.

7.3 Warnings

Alarm – ID	Description
ODCBuilder <<File>> has no signature	The file specified does not have a signature. Check the file with the SyntaxChecker in order to create a signature.
ODCBuilder: Parser warning <<Warning>> - <<File>> <<Linenumber>> <<Characters>>	A warning has occurred at the position specified in the file specified. Check the file.
ODCBuilder: Parser warning: <<Warning>>	The warning specified has occurred. Check the sequence files.

8 Service and Support

Our service helpdesk at our headquarters in Lohr, Germany and our worldwide service will assist you with all kinds of enquiries. You can reach us **around the clock - even on weekend and on holidays**.

	Helpdesk	Service Hotline Worldwide
Phone	**+49 (0) 9352 40 50 60**	Outwith Germany please contact our sales/service office in your area first. For hotline numbers refer to the sales office addresses on the Internet.
Fax	**+49 (0) 9352 40 49 41**	
E-mail	service.svc@boschrexroth.de	
Internet	http://www.boschrexroth.com You will also find additional notes regarding service, maintenance (e.g. delivery addresses) and training.	

Preparing Information

For quick and efficient help please have the following information ready:

- Detailed description of the fault and the circumstances
- Information on the type plate of the affected products, especially type codes and serial numbers
- Your phone, fax numbers and e-mail address so we can contact you in case of questions.

Index

Notes

Rexroth
Bosch Group

Bosch Rexroth AG
Electric Drives and Controls
P.O. Box 13 57
97803 Lohr, Germany
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr, Germany
Tel. +49 (0)93 52-40-0
Fax +49 (0)93 52-48 85
www.boschrexroth.com/electrics

Printed in Germany
DOK-IM*PL*-ODC*PROG***-AP01-EN-P