



Version 5.10

Label Printing

Manual

Document: **Manual - Label Printing**

Created: **2019-10-02**

Last changes: **2019-10-31**

Author: **STernes**



Table of Contents

1	Introduction	3
2	Configuration Elements	4
2.1	Label Printing Template.....	4
2.1.1	Static Elements of a Label Printing Template	4
2.1.2	Dynamic Data Elements in a Label Printing Template.....	5
2.1.3	Sample of a Label Printing Template	7
2.2	Configuring an Activity Step for Label Printing	8
2.2.1	Shop Floor Terminal Configurator	8
2.3	Assignment of Terminal Templates and Printers	11
3	Barcode Types and Scan Methods.....	12
3.1	Barcode Types.....	12
3.2	Barcode Scan Methods	13
4	Annex	14

1 Introduction

The "Label Printing" function is generally used to output static and dynamic data from FORCAM FORCE™ software modules. This data can be either produced as an electronic document or printed out on a printer.

The following general configuration functions are available to the system operator and user within this functional component:

- Label document template
- Embedding dynamic data into this template
- Configuration of events that should cause labels to be printed
- Printer definition

The label printing function can be used by the producer and system operator to print dynamic data based on specific events in the production run directly on the shop floor and make them available for further use (for example, for the logistics and materials process).

2 Configuration Elements

The following configuration elements are available in FORCAM FORCE™ for enabling the initiation of label printing dynamically from the process, triggered by specific events:

- Label printing template:
 - o Placeholder variables for dynamic content
 - o Support for bar code
- Configuration and assignment of trigger events:
 - o Integration into the system template definitions (Shop Floor Terminal configurator and terminal templates)
- Printer definition

2.1 Label Printing Template

It is possible to create an individual label template document for each application situation so that many different label variants can be produced.

The following document type is supported in FORCAM FORCE™ for the definition of label printing templates (label printing forms):

Table 1: Document type that can be used for label printing forms

Document type	Description
*.RTF	Rich Text Format (RTF) is a text file format. It can be used as an exchange format between text processing software of different manufacturers on different operating system platforms. File extension: .rtf MIME type: text/rtf, application/rtf[1]

2.1.1 Static Elements of a Label Printing Template

All of the static design elements available for documents in *.RTF format can be used for the definition of a label form.

A sample of a label printing form with plain static elements is shown below:

QS-FO-383_00	Check List	Processed:
		Date:
Customer	Part Description	
Part Number Customer	Revision Index Part:	
Production Date:	Order Number:	

Fig. 1: Example of a static label printing form in *.RTF

2.1.2 Dynamic Data Elements in a Label Printing Template

The most frequent application situation for a label is that a static form is completed with dynamic data for a defined event from the currently running process and printed out.

Dynamic data content can be arranged at any position within a label template (*.RTF). The content is positioned via defined placeholders within the template.

Table 2: Placeholder definition for dynamic data elements

Name	Description
@Common_Place_Holder@	General placeholder definition: Placeholders begin and end with the character declaration "@".

The following elements provide dynamic content for label printing in FORCAM FORCE™:

Table 3: Data types supported for dynamic data elements

Type	Description
Numeric	Any numeric value can be assigned to a placeholder to represent dynamic content and appear as character(s) on the label.
Alphanumeric	Any alphanumeric value can be assigned to a placeholder to represent dynamic content and appear as character(s) on the label.
String	Any letter (string) can be assigned to a placeholder to represent dynamic content and appear as character(s) on the label.
Bar code	<p>Plain numeric or alphanumeric text data can be assigned to a placeholder to represent dynamic content and appear as a bar code on the label.</p> <p>For this purpose, the bar code 128 must be used as FFT font in the *.RTF document. It is available on the FORCAM FORCE™ application server.</p> <p>The bar code 128 is fully described in the international standard ISO/IEC 15417.</p> <p>In FORCAM FORCE™, the bar code specification 128B is implemented. (https://www3.hi-tier.de/Entwicklung/Technik/bar-code_Code128.html)</p>

The following elements are available in FORCAM FORCE™ for label printing using dynamic placeholders with the associated database field to be used as a source:

Table 4: List of dynamic placeholders available

Placeholder	Description
@material\$number_number@	This placeholder serves as a dynamic parameter and is replaced by the material number at trigger time (it depends on the operation and whether the data is actually available).
@material\$description_description@	This placeholder serves as a dynamic parameter and is replaced by the material description at trigger time (it depends on the operation and whether the data is actually available).
@material\$type_type@	This placeholder serves as a dynamic parameter and is replaced by the material type at trigger time (it depends on the operation and whether the data is actually available).
@erpYieldQuantity@	This placeholder serves as a dynamic parameter and is replaced by the yield quantity at trigger time (it depends on the operation and whether the data is actually available. The yield quantity is made available to the event by the preceding activity "Quantity Booking").
@erpScrapQuantity@	This placeholder serves as a dynamic parameter and is replaced by the scrap quantity at trigger time (it depends on the operation and whether the data is actually available. The scrap quantity is made available to the event by the preceding activity "Quantity Booking").
@erpReworkQuantity@	This placeholder serves as a dynamic parameter and is replaced by the rework quantity at trigger time (it depends on the operation and whether the data is actually available. The rework quantity is made available to the event by the preceding activity "Quantity Booking").

Process instructions can be used to supplement placeholders with functions in templates. In general, several process instructions for a placeholder can be defined and integrated. The process instructions currently available are shown in the following list:

Table 5: List of process instructions available

Placeholder	Description
<p> prefix="</p> <p>Example:</p> <p>@material\$number_number prefix="@</p>	<p>Use this process instruction to add a static prefix (prefix=") to a dynamic value.</p> <p>A process instruction always begins with the character and the process instruction, followed by the process value (here, prefix='Process value'). A process instruction ends either by the beginning of another instruction denoted by or the end of the placeholder definition denoted by @. The process value of the process instruction must be defined within the single quote (') characters.</p> <p>Example with the 'Prefixtest' string: @material\$number_number prefix='Prefixtest'@</p> <p>Example of a dynamic material number: 123456789 Result: Prefixtest123456789</p>

<p> formatter=format_name</p> <p>Example:</p> <p>@material\$number_number formatter=barcode128B@</p>	<p>Use this process instruction for different preprocessing operations for a dynamic value.</p> <p>A process instruction always begins with the character and the process instruction, followed by the process value (here, formatter="format_name"). A process instruction ends either by the beginning of another instruction denoted by or the end of the placeholder definition denoted by @. The process value of the process instruction must be defined in place of format_name.</p> <p>Example: The presentation of the dynamic content in Barcode128 is formatted to the specific Barcode128B format.</p> <p>@material\$number_number formatter=barcode128B@</p> <p>Example with multiple process instructions: @material\$number_number prefix='Prefixtest' formatter=barcode128B@</p>
--	---

2.1.3 Sample of a Label Printing Template

Sample of a label for a quantity message relating to a currently running order. The label should contain the following information (placeholders are marked in red):

- Numeric/alphanumeric material number
- Numeric/alphanumeric material number as a bar code
- Yield quantity reported

The following figure shows the associated *.RTF document:

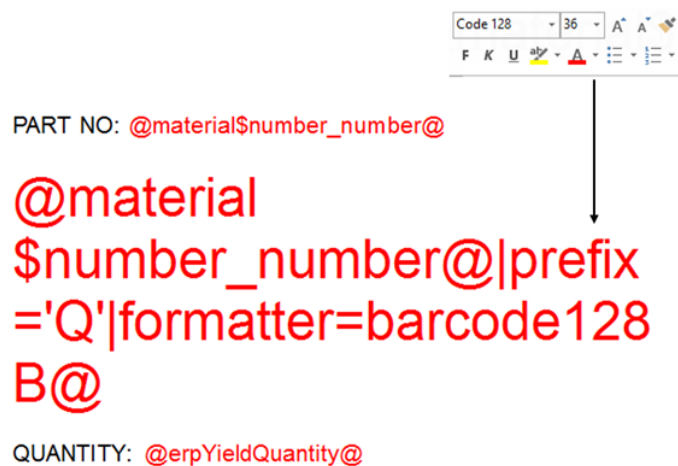


Fig. 2: Sample of a *.RTF template with placeholder for dynamic content (marked in red) and a barcode 128 element in FFT font, formatted as "barcode128B")

PART NO: 299033205



QUANTITY: 90

Fig. 3: Result of label printing showing the dynamic content at the time of the trigger event

2.2 Configuring an Activity Step for Label Printing

To produce a label printout, an activity step must first be defined for the created label printing template (*.RTF).

In FORCAM FORCE™, label printing can only be generated by actions on the Shop Floor Terminal. The Shop Floor Terminal is configured in the System Administration module of FORCAM FORCE™ using the Shop Floor Terminal configurator which generates the required terminal templates.

This configurator can be used to define label printing as an activity step. This is illustrated in a realistic example in the following section.

2.2.1 Shop Floor Terminal Configurator

Path (Workbench): Configurations > Shop Floor Terminal > Template > Edit

For a detailed description of the Shop Floor Terminal configurator for users, refer to the "Master Data and System Configuration" manual of FORCAM FORCE™.

Example

In this example, label printing is added to the "Quantity Booking" activity in an existing terminal template configuration.

Configuration Elements

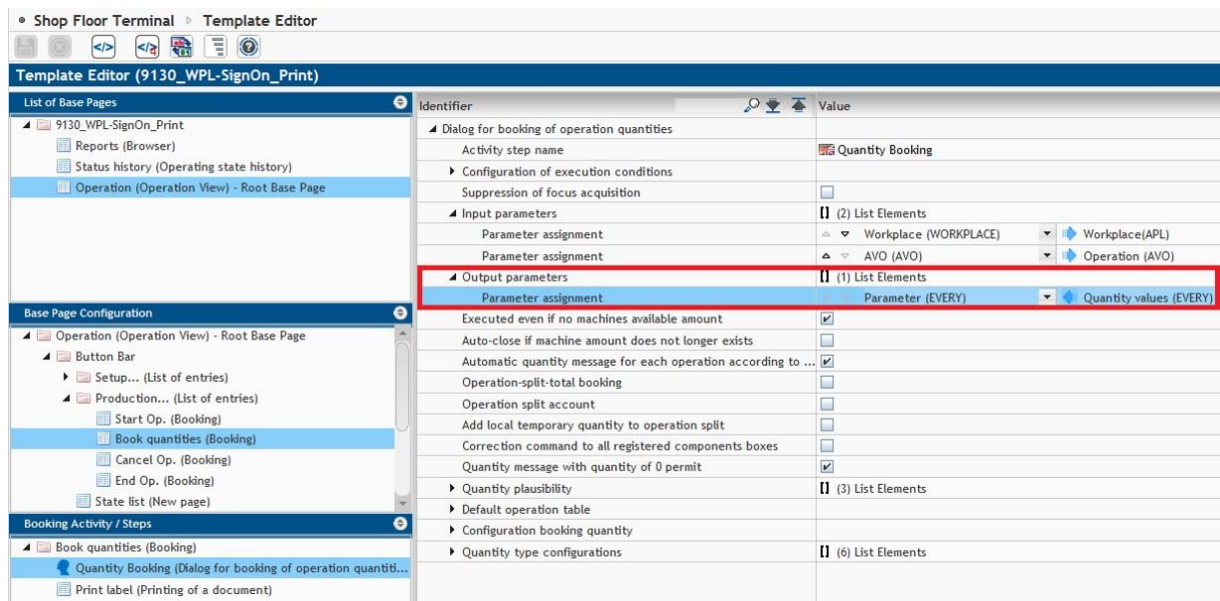


Fig. 4: Quantity booking activity step in the Shop Floor Terminal configurator

In this label printing example, a label should be produced when a quantity booking occurs for an order. Both material data and quantity data are required for the label.

The logical execution steps of the process structure for a Shop Floor Terminal configuration are shown below:

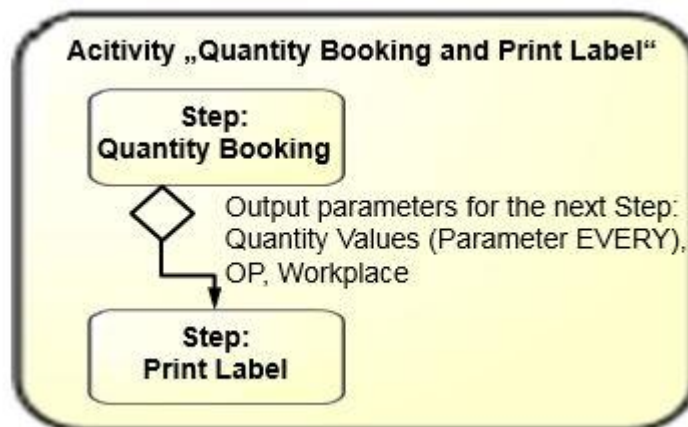


Fig. 5: Logical execution steps for the configuration of a template for "Quantity Booking and Print Label"

The first process step is quantity booking via the corresponding quantity dialog in the Shop Floor Terminal. The worker enters the appropriate booking data for dedicated orders into this dialog. These data are used as the output data of the Quantity Booking step and as input data for a defined subsequent step.

If a Quantity Booking step should be followed by a Print Label activity step, just add another Print Label step.

Configuration Elements

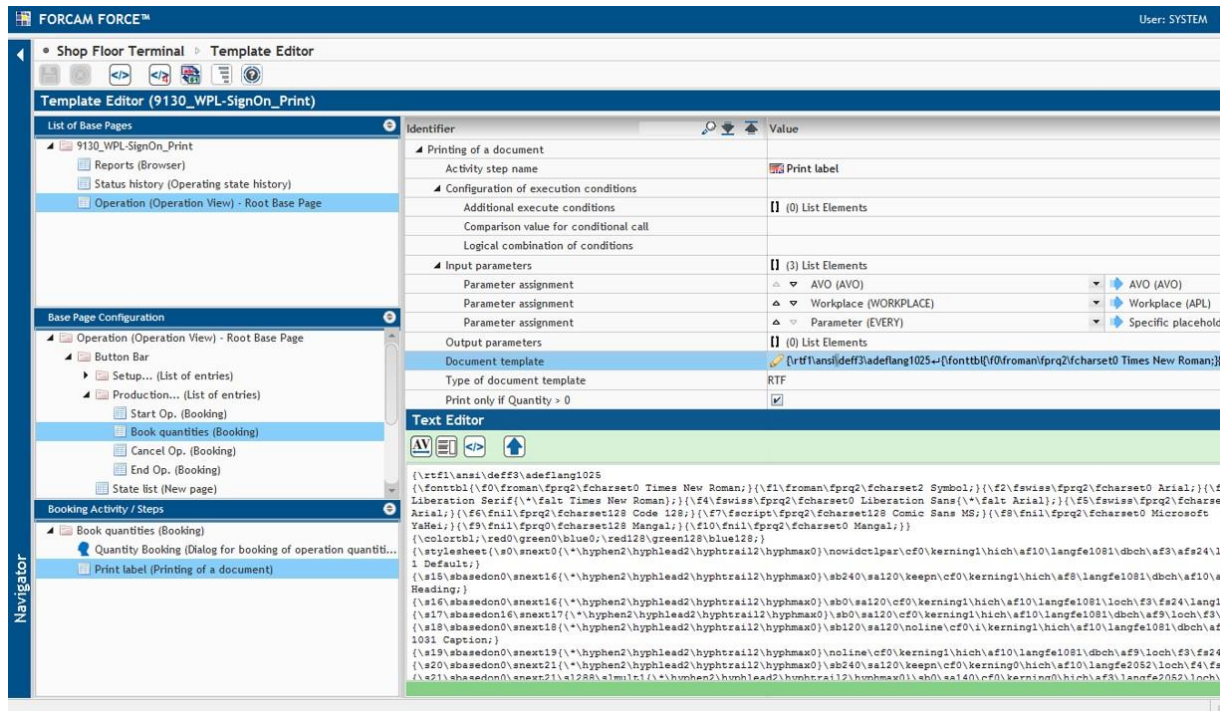


Fig. 6: Print label (Printing of a document) in the Shop Floor Terminal configurator

The following elements can be configured for the "Print label (Printing of a document)" step:

Table 6: Configuration elements for the Print Label step in the Shop Floor Terminal configurator

Configuration elements	Description
Name	Name of step
Input parameters	<p>The most important element. This defines the data that are made available from a previous step as input for the current step for processing.</p> <p>In this example:</p> <ul style="list-style-type: none"> — Operation — Workplace <p>Parameter EVERY (specific placeholder values) → Definition of output values of the Quantity Booking step (Quantity values (EVERY), see Fig. 4)</p>
Output parameters	<p>Also an important element. This defines the data made available as output from the current step. Output data can be made available as input data for a subsequent step.</p> <p>Specify/upload the label printing template (document) that should be printed out when executing the step.</p>

Configuration elements	Description
Document template	The label printing templates to be used at different workplaces can be mapped using different Shop Floor Terminal templates (assignment of workplace -> Shop Floor Terminal template). When selecting the attribute, the template document is opened in a text editor (in plain data format).
Type of document template	Definition of the document type of the label printing template
Print only if Quantity > 0	Active/inactive: Label printing is only executed for quantity bookings with yield quantity > 0.

2.3 Assignment of Terminal Templates and Printers

Path (Workbench): Configurations > Shop Floor Terminal

To be able to use a terminal template created with the Shop Floor Terminal configurator, the relevant workplaces must be assigned to the terminal.

It is possible to assign a printer for label printing explicitly to a specific workplace. The name of the printer must be the precise Windows printer name.

If no printer is specified, the default printer of the computer will be used for printing. In case there is no default printer either, the printer selection dialog will be displayed for manual selection.

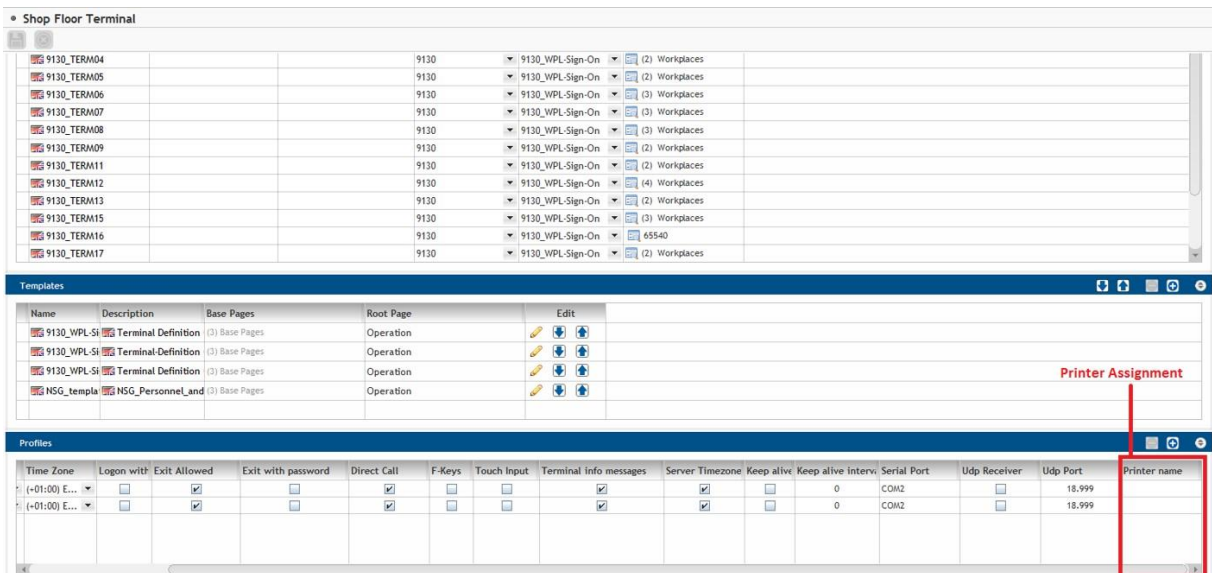


Fig. 7: Explicit or default printer definition in Shop Floor Terminal configuration

3 Barcode Types and Scan Methods

FORCAM FORCE™ supports several methods for scanning (reading) barcodes and printing labels or documents at the terminal. These methods are gathered in the barcode library. The barcode library also contains many different types of barcodes, such as Linear, 2D, QR, PDF417, Data Matrix and so on.

3.1 Barcode Types

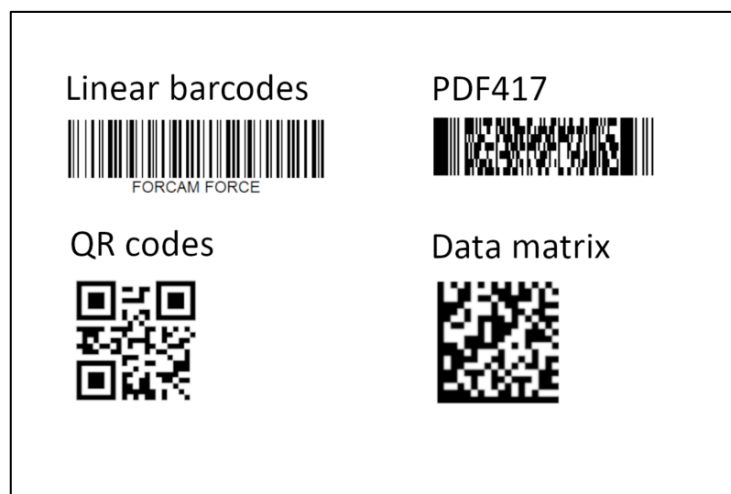



Fig. 8: Example of different types of barcodes

 The barcode library replaces the previous method of using fonts for barcodes.

3.2 Barcode Scan Methods

There are various barcode scanning methods that can be configured according to individual requirements. The configuration is done under the category **Shop Floor Terminal - Template Editor - Mask Configuration - Operation View (OP Mask)**. Further information on the configuration of the barcode scan method is found in the Manual - Master Data and System Configuration.

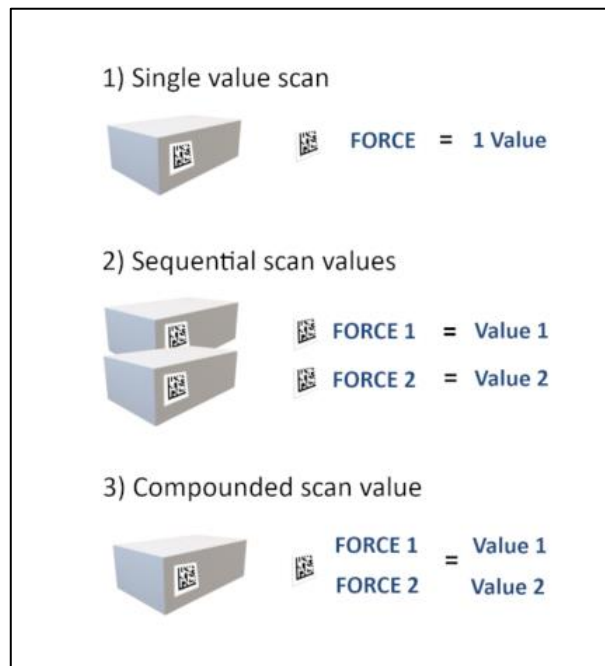


Bild 9: Barcode scan methods

1) Single value scan	Reads a barcode and outputs one value per scan. This means that one barcode scan returns one value and can trigger a background activity, depending on the configuration.
2) Sequential scan values	Several scan values are scanned sequentially via a dialog. This means that several barcodes are scanned one after the other with one value output each. For this purpose, one value after the other is automatically intercepted. The sequence is defined in the Shop Floor Terminal.
3) Compounded scan value	Barcodes with more than one value each can be split and distributed to different variables. This is useful, for example, when several different materials are mounted on one work center. Since a barcode itself does not contain the information as to whether it is a main part or a component, it can be defined here.

4 Annex

The *.RTF data format presents the content as follows (example in Fig. 2):

```
{\rtf1\ansi\deff3\adeflang1025
{\fonttbl{\f0\froman\fprq2\fcharset0 Times New Roman;}{\f1\froman\fprq2\fcharset2 Sym-
bol;}{\f2\fswiss\fprq2\fcharset0 Arial;}{\f3\froman\fprq2\fcharset0 Liberation Serif{\*\falt Times
New Roman};}{\f4\fswiss\fprq2\fcharset0 Liberation Sans{\*\falt Ari-al};}{\f5\fswiss\fprq2\fcharset1
Arial;}{\f6\fnil\fprq2\fcharset1 Code 128;}{\f7\fnil\fprq2\fcharset0 Microsoft
YaHei;}{\f8\fnil\fprq2\fcharset0 Man-gal;}{\f9\fnil\fprq0\fcharset1 Mangal;}}
{\colortbl;\red0\green0\blue0;\red128\green128\blue128;}
{\stylesheet{\s0\next0\nowidctlpar{\*\hyphen2\hyphlead2\hyphtrail2\hyphmax0}\cf0\kern-
ing1\dbch\af10\langfe2052\dbch\af8\afs24\alang1081\loch\af3\fs24\lang1031 Normal;}
{\s15\sbasedon0\next16\sb240\sa120\keepn\dbch\af7\dbch\af8\afs28\loch\af4\fs28
\u220'dcberschrift;}
{\s16\sbasedon0\next16\sl288\smult1\sb0\sa140 Textk\u246'f6rper;}
{\s17\sbasedon16\next17\sl288\smult1\sb0\sa140\dbch\af9 Liste;}
{\s18\sbasedon0\next18\sb120\sa120\noline\i\dbch\af9\afs24\ai\fs24 Beschriftung;}
{\s19\sbasedon0\next19\noline\dbch\af9 Verzeichnis;}
}{\info{\creatim\yr2015\mo8\dy15\hr8\min40}{\rev-
tim\yr0\mo0\dy0\hr0\min0}{\printim\yr0\mo0\dy0\hr0\min0}{\comment Li-
breOffice}{\vern67241218}}\deftab709
\viewscale90
{\*\pgdsctbl
{\pgdsc0\pgdscuse451\pgwsxn11906\pghsxn16838\mar-
glsxn1134\margrsxn1134\margtsxn1134\marginbxn1134\pgdscnxt0 Standard;}}
\formshade\paperh16838\pa-
perw11906\margl1134\margr1134\margt1134\marginb1134\sectd\sbknone\sectun-
locked1\pgndec\pgwsxn11906\pghsxn16838\mar-
glsxn1134\margrsxn1134\margtsxn1134\marginbxn1134\ftnbj\ftnstart1\ftnrstcont\ftnnar\ae-
ndoc\aftnrstcont\aftnstart1\aftnrlc
\pgndec\pard\plain \s0\nowidctlpar{\*\hyphen2\hyphlead2\hyphtrail2\hyphmax0}\cf0\kern-
ing1\dbch\af10\langfe2052\dbch\af8\afs24\alang1081\loch\af3\fs24\lang1031{\afs32\rtlch
\ltrch\loch\fs32\loch\fs5
PART NO: @material$number_number@}
\par \pard\plain \s0\nowidctlpar{\*\hyphen2\hyphlead2\hyphtrail2\hyphmax0}\cf0\kern-
ing1\dbch\af10\langfe2052\dbch\af8\afs24\alang1081\loch\af3\fs24\lang1031\afs32\rtlch
\ltrch\loch\fs32\loch\fs5

\par \pard\plain \s0\nowidctlpar{\*\hyphen2\hyphlead2\hyphtrail2\hyphmax0}\cf0\kern-
ing1\dbch\af10\langfe2052\dbch\af8\afs24\alang1081\loch\af3\fs24\lang1031{\afs72\rtlch
\ltrch\loch\fs72\loch\fs6
@material$number_number|prefix='Q'|formatter=barcode128B@}
\par \pard\plain \s0\nowidctlpar{\*\hyphen2\hyphlead2\hyphtrail2\hyphmax0}\cf0\kern-
ing1\dbch\af10\langfe2052\dbch\af8\afs24\alang1081\loch\af3\fs24\lang1031\afs32\rtlch
\ltrch\loch\fs32\loch\fs5
```

Annex

\par \pard\plain \s0\nowidctlpar{*\hyphen2\hyphlead2\hyphtrail2\hyphmax0}\cf0\kern-
ing1\dbch\af10\langfe2052\dbch\af8\afs24\alang1081\loch\f3\fs24\lang1031{\afs32\rtlch
\ltrch\loch\fs32\loch\f5

QUANTITY: **@erpYieldQuantity@**

\par \pard\plain \s0\nowidctlpar{*\hyphen2\hyphlead2\hyphtrail2\hyphmax0}\cf0\kern-
ing1\dbch\af10\langfe2052\dbch\af8\afs24\alang1081\loch\f3\fs24\lang1031\afs32\rtlch
\ltrch\loch\fs32\loch\f5