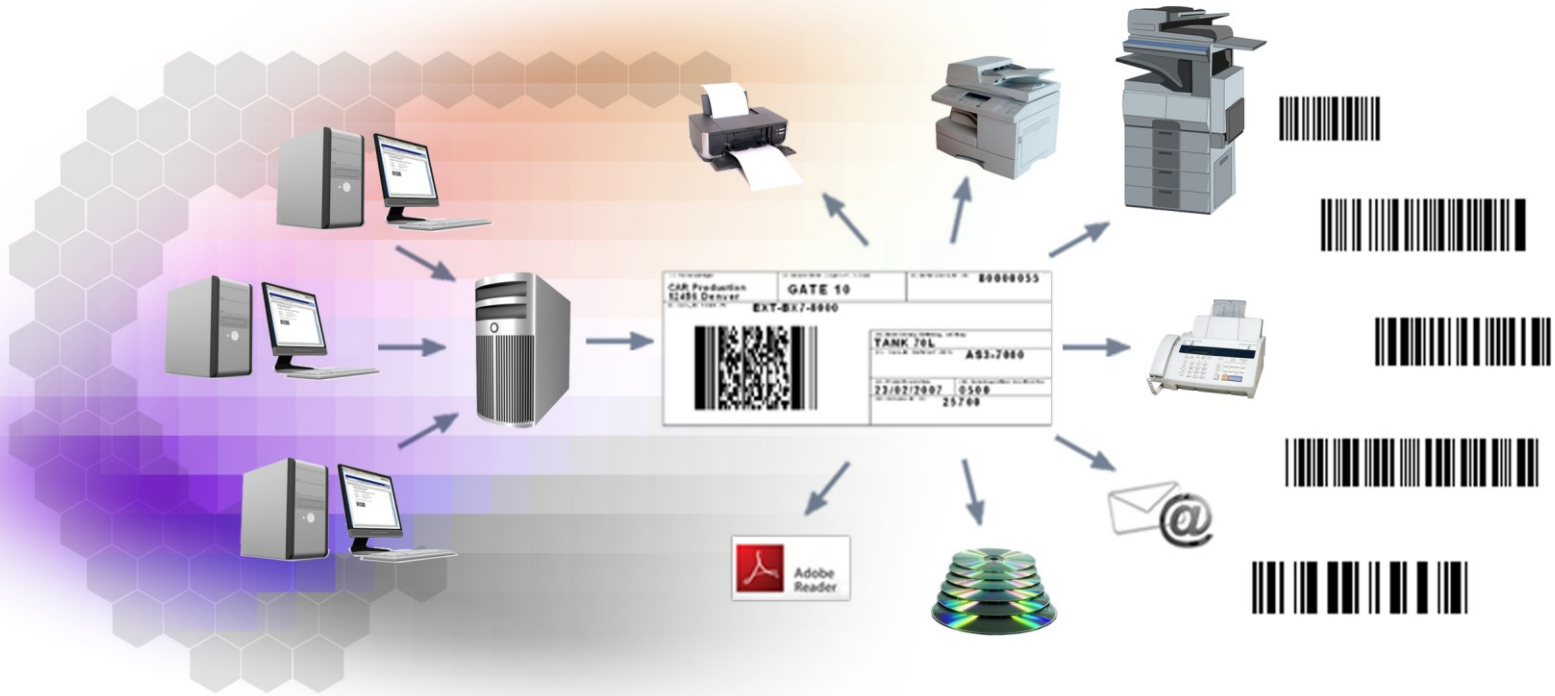
Suchy MIPS

www.suchymips.de



RBarc+

Für SAP® Systeme

Version 12

Handbuch

Copyrights und Warenzeichen

SAP, ABAP, SAPscript, SmartForms sind eingetragene Warenzeichen der SAP AG in Walldorf.

Microsoft Windows ist eingetragenes Warenzeichen der Microsoft Corp.

Alle in diesem Handbuch zusätzlich verwendeten Programmnamen und Bezeichnungen sind u.U. ebenfalls eingetragene Warenzeichen der Herstellerfirmen und dürfen nicht gewerblich oder in sonstiger Weise verwendet werden. Irrtümer vorbehalten.

Bei der Zusammensetzung von Texten und Abbildungen wurde mit großer Sorgfalt gearbeitet. Trotzdem können Fehler nicht vollständig ausgeschlossen werden. Die angegebenen Daten dienen lediglich der Produktbeschreibung und sind nicht als zugesicherte Eigenschaft im Rechtssinne zu verstehen. Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder juristische Verantwortlichkeit noch irgendeine Haftung übernehmen.

Suchy MIPS entwickelt Ihre Produkte ständig weiter, um Ihnen den größtmöglichen Komfort zu bieten. Deshalb behalten wir uns mögliche Abweichungen vom Handbuch zum Produkt vor und bitten um Ihr Verständnis.

Suchy MIPS
Schichtlstraße 32A
81929 München

INHALTSVERZEICHNIS

1 Einführung

RBarc+ ist ein **ABAP** Programm, das zur **ON-THE-FLY** Ausgabe von Barcodes auf Dokumenten, die auf einem SAP Systemen erzeugt wurden, dient. Das Programm wurde so konzipiert, dass die Generierung der Barcodes unabhängig vom Typ der Ausgabe ist, egal ob Sie nun das Dokument mit dem Barcode **drucken**, **faxen**, **mailen**, **archivieren** oder in **PDF umwandeln** möchten. Aus diesem Grund werden wir in diesem Handbuch den Begriff "**drucken**" nur dann verwenden, wenn es sich wirklich nur um das Drucken handelt, ansonsten werden wir den Begriff „**ausgeben**" verwenden.

Barcodes die mit RBarc+ erzeugt wurden, können auf beliebigen Druckern ausgegeben werden, vorausgesetzt ein entsprechender **SAP-Gerätetyp** mit Grafikunterstützung ist vorhanden. Es kommt dabei nicht auf die verwendete Koppelart an, d.h. das Ausgabemedium kann über eine beliebige SAP Koppelart angeschlossen werden.

Wichtig für Sie zu wissen ist, dass die Implementierung einer Barcode Ausgabe mit RBarc+ für Sie KEINE Änderung am SAP-Standard bedeutet. Es sind lediglich Anpassungen in den verwendeten SAPscript-, SmartForms- bzw. AdobeForms-Formularen notwendig. Die dazugehörigen SAP-Ausgabeprogramme unterliegen keinerlei Änderungen. Alle RBarc+ Programme werden dazu im Z-Bereich der ABAP-Workbench installiert, so dass sie von eventuellen System-Updates nicht betroffen sind.

Hier noch einmal eine kurze Zusammenfassung aller wichtigen RBarc+ Eigenschaften:

- Barcodes werden immer originalgetreu ausgegeben, unabhängig vom Ausgabetyp (Drucken, Faxen, Mailen, Archivieren, Umwandeln in PDF).
- Es wird keine Hardwareerweiterung für Drucker und Faxe benötigt.
- Die Ausgabe kann auf beliebigen Drucker- und Faxtypen erfolgen (auch Tintenstrahlgeräte).
- Die Ausgabe kann über beliebige Koppelarten erfolgen.
- Es werden keine SAP-Standards geändert.
- Installation von RBarc+ erfolgt im Z-Raum, so dass es von einem System-Update unberührt bleibt.
- Die Installation erfolgt nur auf dem ABAP Application Server (keine Client-Anpassung notwendig).
- RBarc+ arbeitet unabhängig von dem Betriebssystem, auf dem Ihr SAP-System installiert wurde.

Das Handbuch richtet sich sowohl an System-Administratoren (Installation) als auch an Programmierer (Implementierung von Barcodes in Formularen). Wir gehen von grundlegenden Kenntnissen der ABAP Workbench bei der Installation, als auch von grundlegenden Kenntnissen über ABAP und den verwendeten Formular-Composern (SAPscript, SmartForms bzw. AdobeForms) aus. Wir bitten um Ihr Verständnis dafür, dass wir hier nicht auf Einzelheiten, wie z.B. die Bedienung der ABAP Workbench eingehen können. Das würde den Rahmen des Buches sprengen. Mit einzelnen Fragen können Sie sich jedoch über info@suchymips.de gerne an unseren Support wenden.

Achtung: Alle Anwender, die Dokumente mit von RBarc+ erzeugten Barcodes drucken, müssen folgende Berechtigung für das Objekt **S_BDS_DS** besitzen**:**

ACTVT = 01,02,03 und 06
CLASSNAME = DEVC_STXD_BITMAP
CLASSTYPE = OT

Die entsprechenden Einstellungen können ggf. mit der Transaktion "**PFCG**" durchgeführt werden. Die User Rolle muss die Transaktion SE78 und die Autorisierung für **BC-SRV-KPR-BDS** (Technischer Name **S_BDS_DS**) enthalten. **S_BDS_DS** gehört zu den **Basis-Central-Funktionen**.
Bei fehlenden Rechten des Benutzers erscheint kein Barcode auf dem Ausdruck oder Barcodes werden aus dem System nicht gelöscht.

2 Installation

RBarc+ besteht aus miteinander verbundenen ABAP Programmen vom Typ “Ausführbares Programm“ und „Include“. Die Programme müssen einmalig auf dem ABAP Application Server installiert werden. Nach den Installationen stehen die in RBarc+ implementierten Form Routinen zur Verfügung und können sowohl aus SAPscript als auch SmartForms Formularen aufgerufen werden. Falls auf Ihrem SAP-Testsystem bereits die Demoversion von RBarc+ installiert wurde, reicht es aus, wenn Sie die Inhalte der Programme (Reports) und Includes gegen die der Vollversion austauschen. Die TrueType Fonts (Seite 6) und das Testmaterial (Seite 8) müssen jedoch neu installiert werden.

Gehen Sie wie folgt vor, um **RBarc+** auf Ihrem SAP-System zu installieren. Achten Sie während der Installation sowohl auf die Einhaltung der beschriebenen Reihenfolge als auch auf die genaue Namensgebung der Objekte:

① Neues Paket anlegen:

- Starten Sie die Transaktion **SE80** und legen ein neues Paket an (unser Vorschlag: **ZRBARC)**, in dem die gelieferten Programme und Testformulare installiert werden. Legen Sie dabei einen neuen Auftrag an, damit Sie später die Installation problemlos ins Zielsystem transportieren können. Obwohl für das Paket ein beliebiger Name, der mit dem Buchstaben „**Z**“ oder „**Y**“ beginnt, vergeben werden kann, werden wir uns im weiteren Verlauf des Handbuches auf den vorgeschlagenen Namen **ZRBARC** beziehen.

② Include ZRBARC_12:

- Legen Sie in dem Paket **ZRBARC** ein Objekt von Typ **Include** an und nennen Sie es **ZRBARC_12**.
- Kopieren Sie den Inhalt aus der Datei **ZRBARC_12.INC** in das Quellcodefenster des neu angelegten Objekts.
- Speichern und **aktivieren** Sie das Programm.

③ Include ZSS_12 (für SAPscript-Unterstützung):

- Legen Sie in dem Paket **ZRBARC** ein Objekt von Typ **Include** an und nennen Sie es **ZSS_12**.
- Kopieren Sie den Inhalt aus der Datei **ZSS_12.INC** in das Quellcodefenster des neu angelegten Objekts.
- Speichern und **aktivieren** Sie das Programm.

④ Include ZSF_12 (für SmartForms-Unterstützung):

- Legen Sie in dem Paket **ZRBARC** ein Objekt von Typ **Include** an und nennen Sie es **ZSF_12**.
- Kopieren Sie den Inhalt aus der Datei **ZSF_12.INC** in das Quellcodefenster des neu angelegten Objekts.
- Speichern und **aktivieren** Sie das Programm.

③ Include ZAF_12 (für AdobeForms-Unterstützung):

- Legen Sie in dem Paket **ZRBARC** ein Objekt von Typ **Include** an und nennen Sie es **ZAF_12**.
- Kopieren Sie den Inhalt aus der Datei **ZAF_12.INC** in das Quellcodefenster des neu angelegten Objekts.
- Speichern und **aktivieren** Sie das Programm.

⑤ Programm ZSS_BC_SETTINGS12 (für SAPScript-Unterstützung):

- Legen Sie in dem Paket **ZRBARC_12** ein Objekt von Typ **Programm** an und nennen Sie es **ZSS_BC_SETTINGS12.**

- Kopieren Sie den Inhalt aus der Datei **ZSS_BC_SETTINGS12.PRG** in das Quellcodefenster des neu angelegten Objekts.

- Speichern und **aktivieren** Sie das Programm.

⑥ Programm ZSF_BC_SETTINGS12 (für SmartForms-Unterstützung):

- Legen Sie in dem Paket **ZRBARC** ein Objekt von Typ **Programm** an und nennen Sie es **ZSF_BC_SETTINGS12.**

- Kopieren Sie den Inhalt aus der Datei **ZSF_BC_SETTINGS12.PRG** in das Quellcodefenster des neu angelegten Objekts.

- Speichern und **aktivieren** Sie das Programm.

⑦ Programm ZAF_BC_SETTINGS (für AdobeForms-Unterstützung):

- Legen Sie in dem Paket **ZRBARC_12** ein Objekt von Typ **Programm** an und nennen Sie es **ZAF_BC_SETTINGS12.**

- Kopieren Sie den Inhalt aus der Datei **ZAF_BC_SETTINGS12.PRG** in das Quellcodefenster des neu angelegten Objekts.

- Speichern und **aktivieren** Sie das Programm.

⑧ TrueType Fonts für vertikale Ausgabe der Klarschriftzeile:

Weder SAPscript noch SmartForms bieten die Möglichkeit, dynamische Texte in vertikaler Richtung oder um 180 Grad gedreht auszugeben. Daher bedarf es besonderer Maßnahmen, falls Sie einen Barcode drehen und dazu die Klarschriftzeile gedreht, passend zu dem Barcode, ausgeben möchten, wie z.B. auf dem folgenden Bild dargestellt:

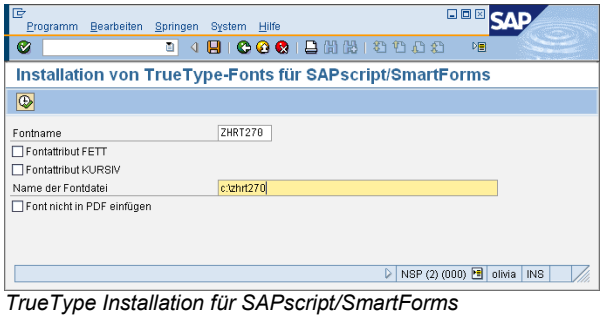
Barcode um 270 Grad gedreht.

Um Ihnen dennoch die Möglichkeit einer solchen Ausgabe zu ermöglichen, liefern wir mit RBarc+ 3 TrueType Schriften aus, deren Zeichen jeweils um 90, 270 bzw. 180 gedreht sind. Wenn Sie diese Schriften auf Ihrem SAP-System installieren und damit die Klarschriftzeile formatieren, werden Sie das

gewünschte Ergebnis erzielen. Die genaue Vorgehensweise wird weiter unten in den Kapiteln 5.3 (für SAPscript) und 6.3 (für SmartForms) detailliert beschrieben.

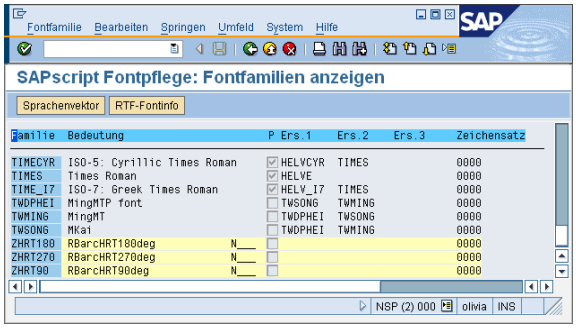
- Mit **RBarc+** werden die folgenden TrueType Schriften ausgeliefert: **ZHRT90.TTF, ZHRT180.TTF** und **ZHRT270.TTF**. Um diese Schriften zu installieren, starten Sie die Transaktion **SE73** und wählen *TrueType Font installieren*.

- Geben Sie im Feld *Fontname* den jeweiligen Schriftnamen ein (ohne die Erweiterung .TTF) und führen das Programm aus.



TrueType Installation für SAPscript/SmartForms

- Prüfen Sie nach der Installation mit der Transaktion **SE73**, ob die Fontfamilien **ZHRT90**, **ZHRT180** und **ZHRT270** angelegt wurden. Das Ergebnis sollte in etwa so aussehen:



Liste der installierten Fontfamilien

Merke: Falls Sie den Gerätetypen SAPWIN nutzen und über das SAP Client Programm SAPLPD drucken, müssen Sie diese Schriften auch auf den Microsoft Windows Clients installieren, sonst erfolgt keine korrekte Ausgabe von gedrehten Klarschriftzeilen.

Zusätzlich zu den Programmen, die für die Ausgaben von Barcodes notwendig sind, liefern wir einige **Testressourcen**, mit denen die Barcode Ausgabe sofort nach der Installation getestet werden kann, ohne dass Sie vorher eine Barcode Implementierung in Ihren Formularen durchführen müssen. Um diese Ressourcen zu installieren, gehen Sie wie folgt vor:

① Programm ZSS_BC_PRINT (für Druck eines SAPscript Formulars):

- Legen Sie in dem Paket **ZRBARC_12** ein Objekt von Typ **Programm** an und nennen Sie es **ZSS_BC_PRINT.**

- Kopieren Sie den Inhalt aus der Datei **ZSS_BC_PRINT.PRG** in das Quellcodefenster des neu angelegten Objekts.

- Speichern und **aktivieren** Sie das Programm.

② SAPscript Formular ZSS_BC_FORM:

- Starten Sie den ABAP Editor (Transaktion **SE38**) und danach das SAP Programm **RSTXSCRP**.

- Tragen Sie im Feld *Objektname* den Formularnamen **ZSS_BC_FORM** und im Feld *Modus* den Wert **IMPORT** ein und führen das Programm anschließend aus. Selektieren Sie im Dateiauswahlmenü die Datei **ZSS_BC_FORM**.**FOR** und klicken auf **<OK>** um den Vorgang abzuschließen.

③ SmartForms Formular ZSF_BC_FORM:

- Starten Sie die Transaktion **SMARTFORMS**

- Tragen Sie im Feld *Formular* den Namen des Formulars **ZSF_BC_FORM** ein

- Wählen Sie aus dem Menü *Hilfsmittel / Formular Hochladen*. Selektieren Sie im Dateiauswahlmenü die Datei **ZSF_BC_FORM.XML** und klicken Sie auf **<OK>** um den Vorgang abzuschließen**.**

④ SmartForms Stil ZSF_BC_STIL (nur Vollversion):

- Starten Sie die Transaktion **SMARTFORMS**

- Tragen Sie im Feld *Stil* den Namen des Stils **ZSF_BC_STIL** ein

- Wählen Sie aus dem Menü *Hilfsmittel / Stil Hochladen*. Selektieren Sie im Dateiauswahlmenü die Datei **ZSF_BC_STIL.XML** und klicken Sie auf **<OK>** um den Vorgang abzuschließen**.**

⑤ Programm ZAF_BC_PRINT (für Druck eines AdobeForms Formulars):

- Legen Sie in dem Paket **ZRBARC_12** ein Objekt von Typ **Programm** an und nennen Sie es **ZAF_BC_PRINT.**

- Kopieren Sie den Inhalt aus der Datei **ZAF_BC_PRINT.PRG** in das Quellcodefenster des neu angelegten Objekts.

- Speichern und **aktivieren** Sie das Programm.

⑥ AdobeForms Schnittstelle ZAF_BC_INTERFACE12:

- Starten Sie die Transaktion **SFP**
- Tragen Sie im Feld *Schnittstelle* den Namen der Schnittstelle **ZAF_BC_INTERFACE12** ein
- Wählen Sie aus dem Menü *Hilfsmittel / Formularobjekt hochladen*. Selektieren Sie im Dateiauswahlmenü die Datei **ZAF_BC_INTERFACE12.XML** und klicken Sie auf **<OK>** um den Vorgang abzuschließen**.**

⑦ AdobeForms Formular ZAF_BC_FORM:

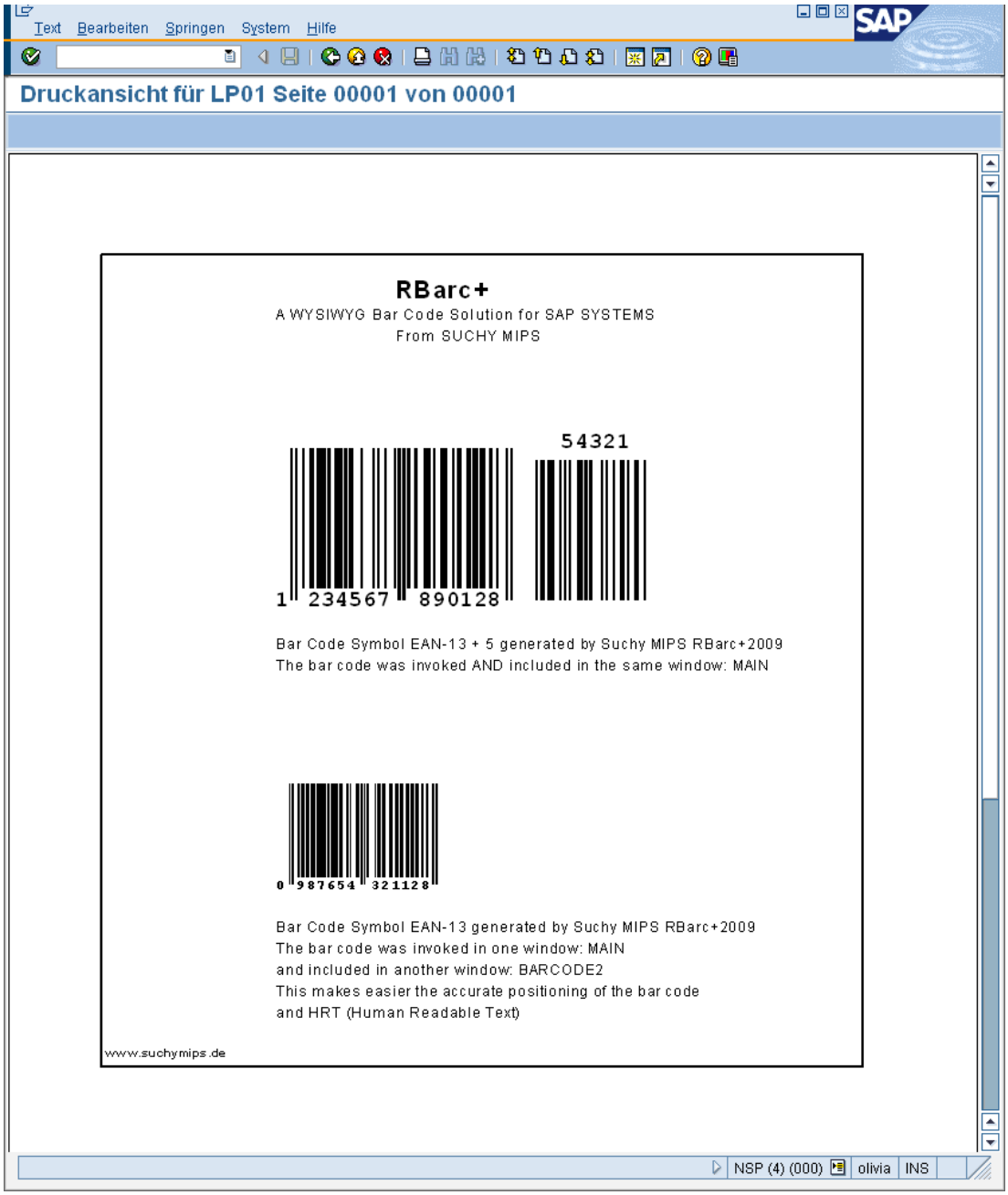
- Starten Sie die Transaktion **SFP**
- Tragen Sie im Feld *Formular* den Namen des Formulars **ZAF_BC_FORM** ein
- Wählen Sie aus dem Menü *Hilfsmittel / Formularobjekt hochladen*. Selektieren Sie im Dateiauswahlmenü die Datei **ZAF_BC_FORM.XML** und klicken Sie auf **<OK>** um den Vorgang abzuschließen**.**

Wenn Sie allen im Kapitel 2 beschriebenen Schritten gefolgt sind, ist die Installation jetzt abgeschlossen. Sie können sich nun dem Test und der Implementierung von Barcodes in Ihren Formularen widmen. Lesen Sie dazu die folgenden Kapitel.

3 Testausdruck mit SAPscript

- Starten Sie den ABAP Editor mit der Transaktion **SE38** und führen Sie das Programm **ZSS_BC_PRINT** aus.

Das Programm **ZSS_BC_PRINT** druckt das mitgelieferte **SAPscript** Formular **ZSS_BC_FORM**. Da die Barcode Ausgabe mit **RBarc+** geräteunabhängig ist, müssen Sie das Formular nicht unbedingt drucken, sondern Sie können das Ergebnis auf dem Bildschirm betrachten, indem Sie im Druckerdialog *Druckansicht* wählen. Das Ergebnis sollte in etwa wie folgt aussehen:



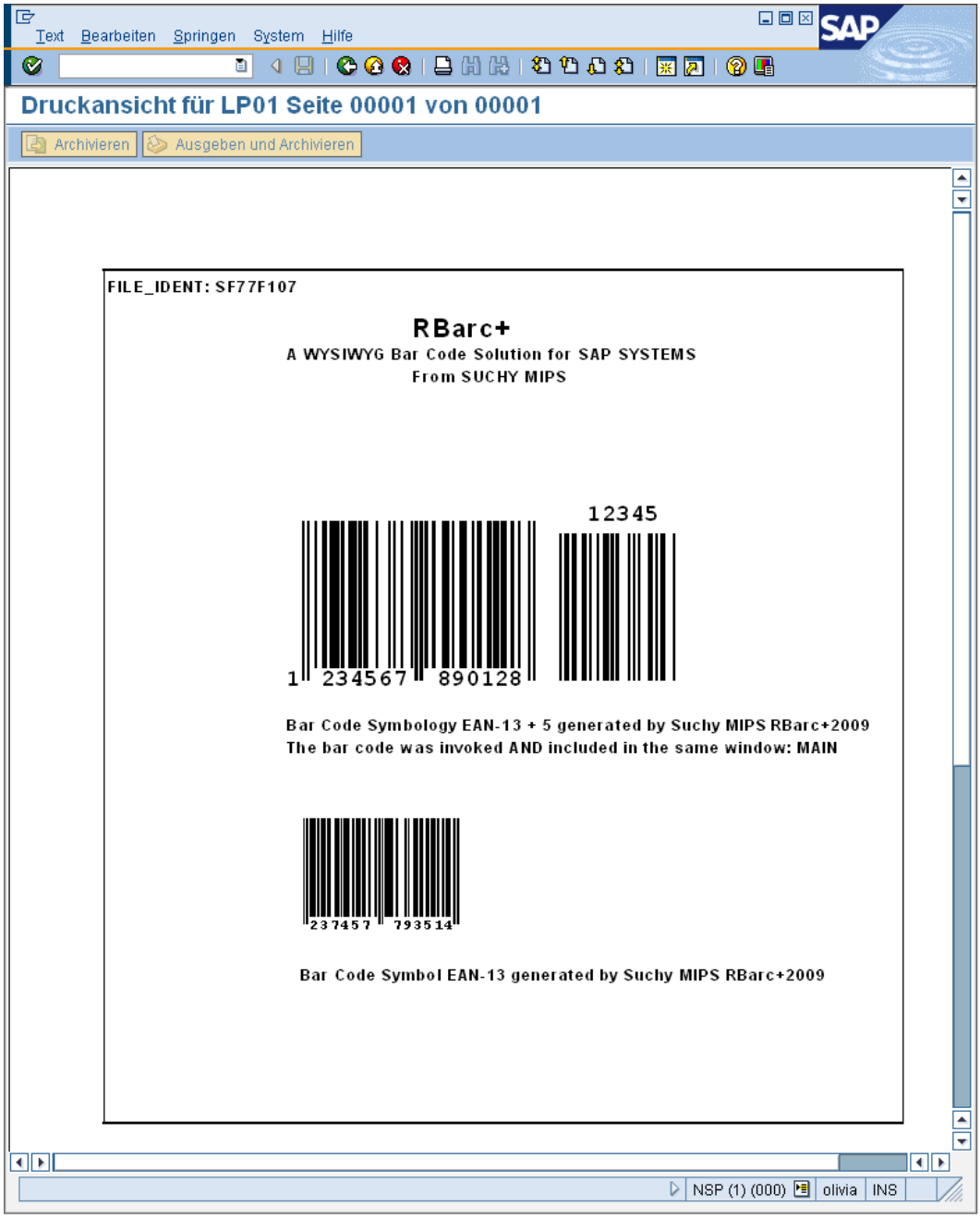
Druckvorschau des Demo SAPscript Formulars ZSS_BC_FORM

Wenn Sie die Barcodes auf der Formularausgabe sehen, dann heißt das, dass die Installation erfolgreich war. Sie können jetzt damit anfangen, Barcodes in Ihren eigenen SAPscript Formularen mit RBarc+ zu implementieren.

4 Testausdruck mit SmartForms

- Starten Sie den die Transaktion **SMARTFORMS**.
- Geben Sie im Feld *Formular* den Formularnamen **ZSF_BC_FORM** ein und klicken auf das Symbol  oder wählen Sie aus dem Menü *SmartForms / Testen.*

Da die Barcode Ausgabe mit **RBarc+** geräteunabhängig ist, müssen Sie das Formular nicht unbedingt drucken, sondern Sie können das Ergebnis auf dem Bildschirm betrachten, indem Sie im Druckerdialog *Druckansicht* wählen. Das Ergebnis sollte dann in etwa wie folgt aussehen:



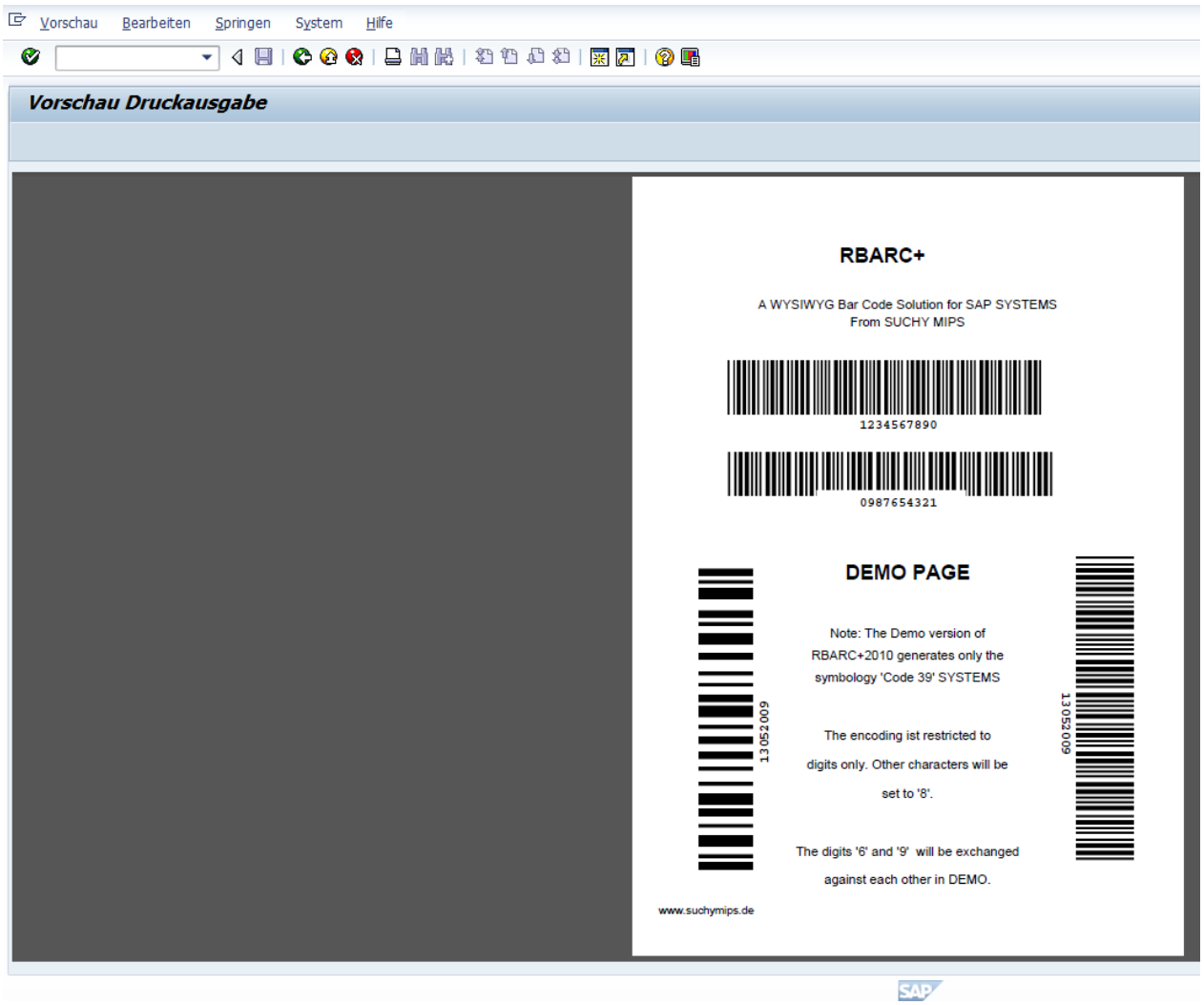
Druckvorschau des Demo SAPscript Formulars ZSS_BC_FORM

Wenn Sie die Barcodes auf der Formularausgabe sehen, dann heißt das, dass die Installation erfolgreich war. Sie können jetzt damit anfangen, Barcodes in Ihren eigenen SmartForms Formularen mit RBarc+ zu implementieren.

5 Testausdruck mit AdobeForms

- Starten Sie den ABAP Editor mit der Transaktion **SE38** und führen Sie das Programm **ZAF_BC_PRINT** aus.

Das Programm **ZAF_BC_PRINT** druckt das mitgelieferte **AdobeForms-**Formular **ZAF_BC_FORM**. Da die Barcode Ausgabe mit **RBarc+** geräteunabhängig ist, müssen Sie das Formular nicht unbedingt drucken, sondern Sie können das Ergebnis auf dem Bildschirm betrachten, indem Sie im Druckerdialog **Druckansicht** wählen. Das Ergebnis sollte in etwa wie folgt aussehen:



Druckvorschau des Demo AdobeForms Formulars ZAF_BC_FORM

Wenn Sie die Barcodes auf der Formularausgabe sehen, dann heißt das, dass die Installation erfolgreich war. Sie können jetzt damit anfangen, Barcodes in Ihren eigenen AdobeForms-Formularen mit RBarc+ zu implementieren.

6 Implementierung einer Barcode Ausgabe mit SAPscript

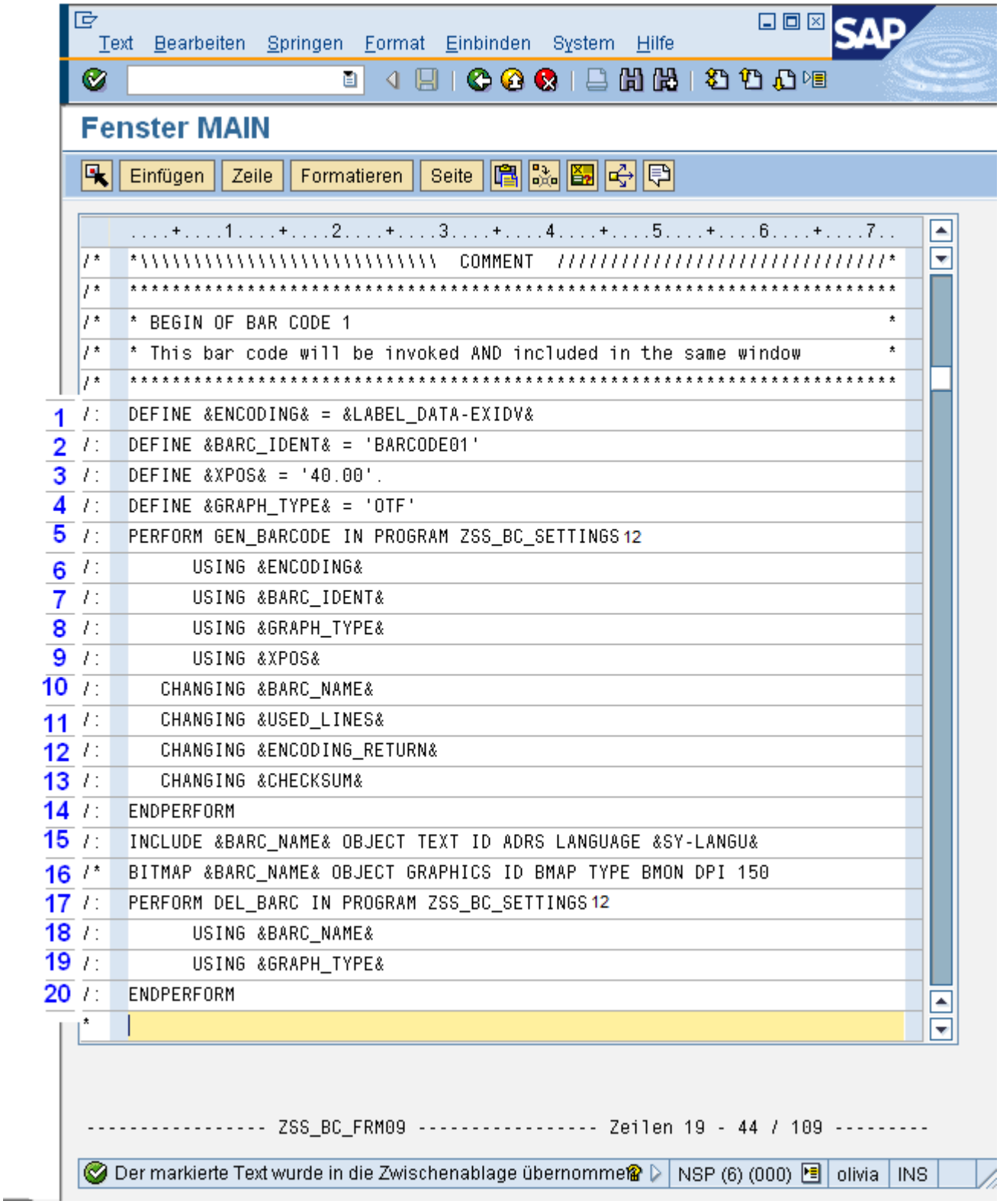
6.1 Einbinden von Barcodes in ein SAPscript Formular

In einem SAPscript Formular können beliebig viele Barcodes eingebunden werden. Obwohl es ratsam ist, für einen Barcode ein separates Fenster zu definieren, kann ein Barcode auch in ein beliebiges, vorhandenes Fenster eingebunden werden. Der Einfachheit halber konzentriert sich unsere Beschreibung auf das Einbinden eines Barcodes in ein bestehendes SAPscript Fenster. Die Barcode Einbindung geschieht ausschließlich über sog. Command Anweisungen, so dass sich dadurch das bestehende Formularlayout nicht ändert (der eingefügte Barcode verdrängt keinen Platz). Bestehende Druckprogramme für ein SAPscript Formular werden **NICHT** geändert. Die Anpassungen betreffen lediglich das Formular selbst (somit ist die RBarc+ Lösung für SAP-Updates geeignet). Damit die Formularanpassungen so gering wie nur möglich ausfallen, werden im SAPscript Formular nur die nötigsten Definitionen vorgenommen. Dazu gehören die Informationen: „**was soll kodiert werden**“? und „**wo soll der Barcode positioniert werden**“?. Alle anderen Barcode Eigenschaften, wie **Symbologie**, **Breite**, **Höhe** usw. werden in dem außen liegenden ABAP Programm **ZSS_BC_SETTINGS12** vorgenommen.

Die Logik der Vorgehensweise ist immer die gleiche und richtet sich nach folgendem Prinzip:

- Einmalige Definition der Barcodeeigenschaften (Symbologie, Höhe, Breite usw.) in einer separaten Form Routine im ABAP Programm **ZSS_BC_SETTINGS12**.
- Definition der Variable(n) im SAPscript Formular, die als Barcode kodiert werden sollen.
- Festlegen einer Barcode Identifikation (jeder Barcode in einem Formular muss eindeutig identifizierbar sein).
- Aufruf der Form Routine **GEN_BARCODE** im Programm **ZSS_BC_SETTINGS12** und Übergabe der Parameter (zu der in erster Linie die zu kodierenden Daten und die Barcode Identifikation gehören).
- Übernahme der von RBarc+ zurück gelieferten Rückgabeparametern (in erster Linie der dynamisch erzeugte Name der einzubindenden Barcodegrafik).
- Dynamisches Einbinden der Barcodegrafik in das Formular
- Löschen der zuvor erzeugten Barcodegrafik aus dem System.

Über die Eigenschaften der Barcodes, wie **Symbologie**, **Höhe**, **Breite** usw. schreiben wir später im **Kapitel 7 „Die Barcode Generierung“**, da dieser Teil identisch für **SAPscript** und **SmartForms** Anwendungen ist. Im Folgenden präsentieren wir ein Listing eines **SAPscript** Formulars mit entsprechenden Erklärungen, die Sie in die Lage versetzen sollte, selbst Barcodes mir RBarc+ in **SAPscript** Formulare einzubinden.



Listing der Barcode Implementierung in einem SAPscript Fenster

Wie schon früher erwähnt, sind alle Zeilen Command Zeilen, die das Layout des Formulars nicht verändern, da die Barcodegrafik keinen Platz verdrängt. Die Erläuterung des Listings finden Sie auf den folgenden Seiten.

Erläuterung der einzelnen Zeilen:

Zeile 1:

DEFINE &ENCODING& = &LABEL_DATA-EXIDV&

Hier wird der Wert der Variablen **EXIDV** aus der Tabelle **LABEL_DATA**, (die hier zur Ablauflogik des SAPscript Formulars gehört) der Variablen **ENCODING** zugewiesen. Die Variable **ENCODING** übergibt den zu verschlüsselnden Wert an RBac+. Der Name „**ENCODING**“ darf deshalb nicht geändert werden. Es kann nur eine **ENCODING** Variable pro RBarc+ Aufruf übergeben werden, aber bei Bedarf können mehrere Variablen aus der Formularablauflogik gleichzeitig in eine **ENCODING** Variable geschrieben werden. Dabei muss vor das Gleichheitszeichen ein Doppelpunkt gesetzt und die Variablen in Hochkommata eingeschlossen werden, z.B.

DEFINE &ENCODING& **:= '**&LABEL_DATA-EXIDV&&LABEL_DATA-EXIDT&**'**

Merke: Die Länge der Variablen **ENCODING** darf insgesamt **70 Zeichen** nicht überschreiten.

Zeile 2:

DEFINE &BARC_IDENT& = 'BARCODE01'

Hier wird eine Barcode Identifikation vergeben, die über die Variable **BARC_IDENT** an RBarc+ übermittelt wird. Die maximale Länge für die Variable beträgt **10 Zeichen**. Jeder Barcode im Formular muss eine eindeutige Identifikation besitzen, deshalb empfiehlt es sich, einen Zähler als diskriminierendes Mittel zu verwenden, wie im obigen Beispiel das „01“.

Merke: Die Barcode Identifikation muss dem Namen der Form Routine im Programm **ZSS_BC_SETTINGS12** entsprechen, in der die Barcode Eigenschaften definiert werden. Ggf. müssen Sie die Form Routine für einen neuen Barcode im Programm **ZSS_BC_SETTINGS12** selbst erstellen. Bei der Auslieferung sind im Programm **ZSS_BC_SETTINGS12** 5 Form Routinen vordefiniert: **BARCODE01, BARCODE02, BARCODE03, BARCODE04 und BARCODE05.** Die dort definierten Barcode Parameter können Sie jederzeit ändern. Eine detaillierte Beschreibung dazu finden Sie in **Kapitel 7 „Die Barcode Generierung“**,.

Zeile 3:

DEFINE &XPOS& = '40.00'.

Mit dieser Definition wird der Barcode um 40 Einheiten nach rechts verschoben. Die Maßeinheit für den Wert entspricht der Maßeinheit für Barcode Abmessungen, wie sie im Programm **ZSS_BC_SETTINGS12** in der entsprechenden Form Routine, in der die Barcode Eigenschaften definiert wurden, festgelegt wird (Parameter **UNIT**). Der Standardwert ist „mm“.

Merke: Mit dem Parameter **XPOS** kann ein Barcode nur nach rechts verschoben werden. Ein verschieben nach links (also außerhalb des linken Fensterrahmens) ist nicht möglich.

Merke: Es ist nicht möglich, einen Barcode mit einem ähnlichen Parameter **YPOS** relativ zur aktuellen Cursorposition nach unten bzw. nach oben zu verschieben. Die Eingabe eines **YPOS** Parameters ist grundsätzlich möglich, führt jedoch im Endeffekt dazu, dass der Barcode nicht im aktuellen Fenster, sondern immer auf der gleichen Höhe der physikalischen Seite erscheinen würde.

Zeile 4:

DEFINE &GRAPH_TYPE& = 'OTF'.

Mit dieser Definition wird der Art des Grafiktyps festgelegt, der von RBarc+ erzeugt wird. Standardmäßig wird **OTF** erzeugt (der Parameter sollte nach Möglichkeit nicht geändert werden). Prinzipiell ist es möglich eine sog. BMP-Bitmap zu erzeugen (DEFINE &GRAPH_TYPE& = 'BMP'). Da diese jedoch nicht relativ zum Fenster nach rechts verschoben werden kann, kommt dieser Grafiktyp unter SAPscript fast nie zur Anwendung.

Zeile 5-9:

```
PERFORM GEN_BARCODE IN PROGRAM ZSS_BC_SETTINGS12.
    USING &ENCODING&
    USING &BARC_IDENT&
    USING &GRAPH_TYPE&
    USING &XPOS&
```

Aufruf des der Form Routine **GEN_BARCODE** im Programm **ZSS_BC_SETTINGS** und Übergabe der Parameter **ENCODING**, **BARC_IDENT**, **GRAPH_TYPE** und **XPOS**.

Merke: Es ist prinzipiell möglich, für jedes Formular ein eigenes ABAP Programm mit dort definierten Barcode Eigenschaften zu verwenden. In diesem Fall kopieren Sie das Programm **ZSS_BC_SETTINGS12** in ein anderes Programm und rufen dieses im SAPscript Formular auf. Achten Sie jedoch dabei darauf, dass Sie die Namen der Variablen nicht verändern, da sonst das Programm nicht einwandfrei funktionieren würde.

Zeile 10-13:

```
CHANGING &BARC_NAME&
CHANGING &USED_LINES&
CHANGING &ENCODING_RETURN&
CHANGING &CHECKSUM&
```

In den Zeilen 10-13 werden die Variablen für Rückgabewerte aus Rbarc+ definiert. Diese Namen dürfen nicht geändert werden. Falls Sie die Werte dieser Variablen zu anderweitiger Verwendung benötigen, empfiehlt es sich, diese in andere, selbst definierte Variablen zu schreiben.

BARC_NAME – Name der von RBarc+ erzeugten Grafik mit dem Barcodebild.
USED_LINES – gibt an, wie viele Zeilen (á 1/6 Zoll) der Barcode einnimmt.
ENCODING_RETURN – gibt an, was wirklich verschlüsselt wurde.
CHECKSUM – die von RBarc+ errechnete Prüfziffer (falls erforderlich).

Merke: Der zurückgegebene Wert für **ENCODING_RETURN** kann manchmal vom übergebenen Wert für **ENCODING** abweichen, z.B. dann, wenn bestimmte Optionen, wie das Löschen führender Nullen im Programm **ZSS_BC_SETTINGS12** eingestellt wurde.

.

Zeile 14:

ENDPERFORM

Ende des in Zeile 5 eingeleiteten PERFORM Kommandos.

Zeile 15:

INCLUDE &BARC_NAME& OBJECT TEXT ID ADRS LANGUAGE &SY-LANGU&

Mit dieser Anweisung wird die von RBarc+ erzeugte Barcodegrafik in das Formular dynamisch, während der Laufzeit eingebunden. Dieses Kommando gilt nur für **OTF** Grafiken (empfohlene Standardeinstellung).

Zeile 16:

BITMAP &BARC_NAME& OBJECT GRAPHICS ID BMAP TYPE BMON DPI 150

Diese Anweisung ist im Formular auskommentiert und dient hier nur dokumentarischen Zwecken. Falls Sie sich für den Grafiktyp **BMP** entschieden haben (durch Angabe DEFINE &GRAPH_TYPE& = 'BMP' in Zeile 4), dann muss die Grafik mit diesem Kommando dynamisch eingebunden werden.

Merke: Im Unterschied zu OTF-Grafiken muss bei BMP-Grafiken die Auflösung (DPI) angegeben werden, die unbedingt mit der Auflösung übereinstimmen muss, die im Programm **ZSS_BC_SETTINGS12** für den einzufügenden Barcode definiert wurde. Ist das nicht der Fall, wird eine andere Barcode Größe dargestellt, als erwartet.

Merke: Eine Grafik vom Typ **BMP** kann im Gegensatz zu einer **OTF** Grafik innerhalb des Fensters nicht horizontal verschoben werden. Der Parameter **XPOS** bleibt ohne Wirkung. Das **BMP** Format wird aus Kompatibilitätsgründen zu früheren RBarc+ Versionen, die noch kein **OTF** Format kannten, unterstützt.

Zeile 17-20:

```
PERFORM DEL_BARC IN PROGRAM ZSS_BC_SETTINGS12
    USING &BARC_NAME&
    USING &GRAPH_TYPE&
ENDPERFORM
```

Mit diesen Zeilen wird der zuvor erzeugte Barcode wieder aus dem System gelöscht.

6.2 Einfügen einer Klarschriftzeile.

Aus technischen Gründen kann die Klarschriftzeile nicht zusammen mit der Barcodegrafik erzeugt werden. Deshalb muss, falls eine Klarschriftzeile gewünscht wird, diese im Formular separat ausgegeben werden. Dabei bieten sich 2 Möglichkeiten an:

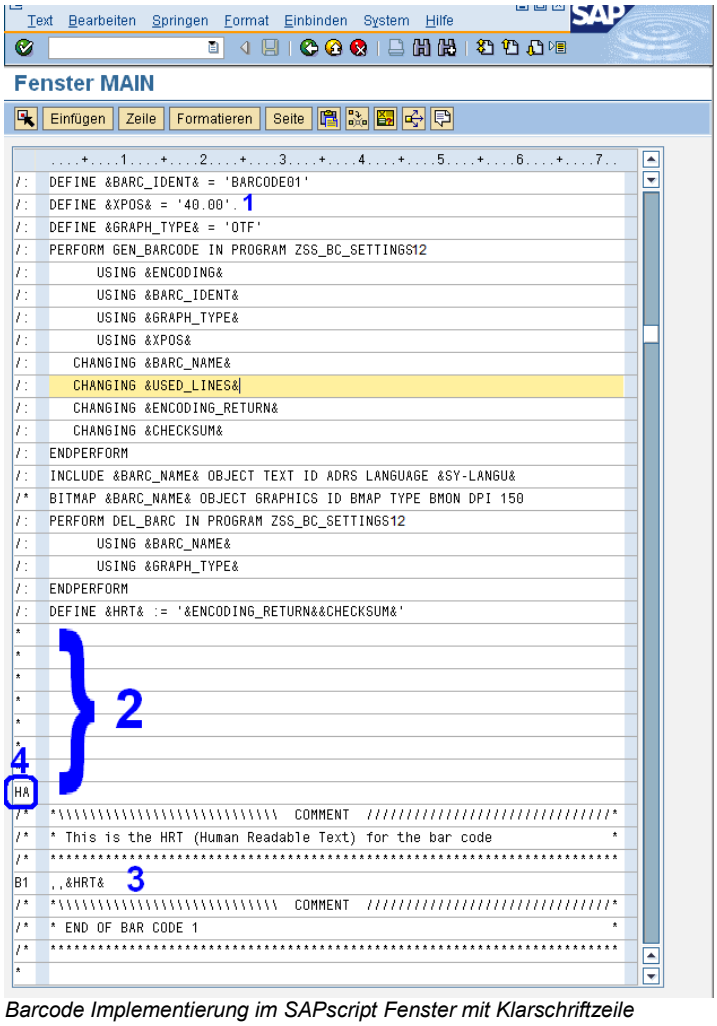
- Klarschriftzeile im gleichen Fenster definieren, in dem der Barcode steht
- Klarschriftzeile in einem eigenen Fenster definieren.

Wir werden zunächst den ersten Ansatz verfolgen.

Für die Klarschriftzeile sollte der Wert der Variablen **ENCODING_RETURN** verwendet werden. Sollte die Prüfziffer zusammen mit der Klarschriftzeile ausgegeben werden, müssen die Variablen zusammengefügt werden, z.B. in einer SAPscript Kommandozeile mit folgendem Inhalt:

DEFINE &HRT& **:=** **'**&ENCODING_RETURN&&CHECKSUM&**'**

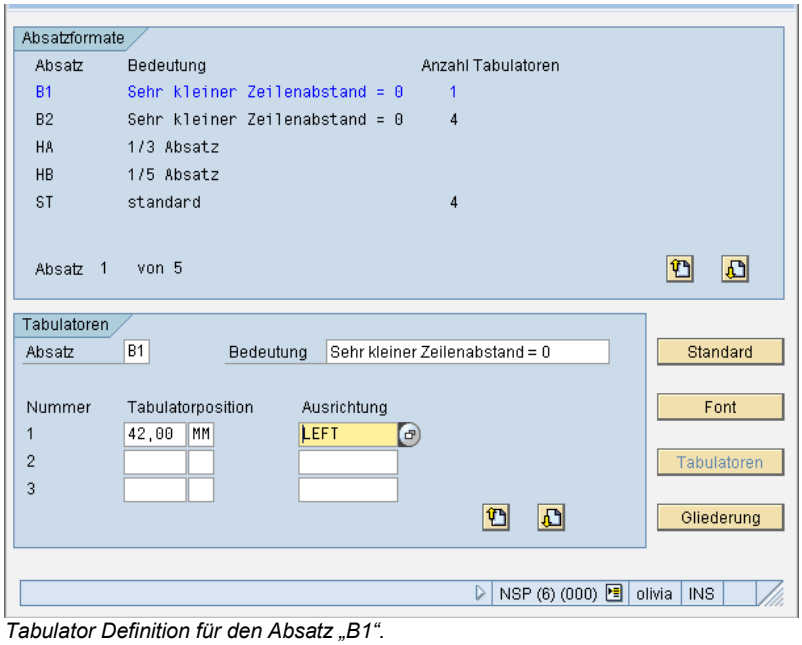
Wenn Sie die Klarschriftzeile im gleichen Fenster einfügen wollen, in dem der Barcode eingefügt wurde, dann müssen Sie bedenken, dass der Barcode keinen Platz verdrängt. Sie müssen also die Klarschriftzeile einige Zeilen weiter unten nach der Include Anweisung für die Barcodegrafik eingeben, um die Klarschriftzeile unter dem Barcode zu positionieren. Die horizontale Position der Klarschriftzeile können Sie mit Leerzeichen oder Tab Stopps vor der Variablen **&HRT&** erreichen:



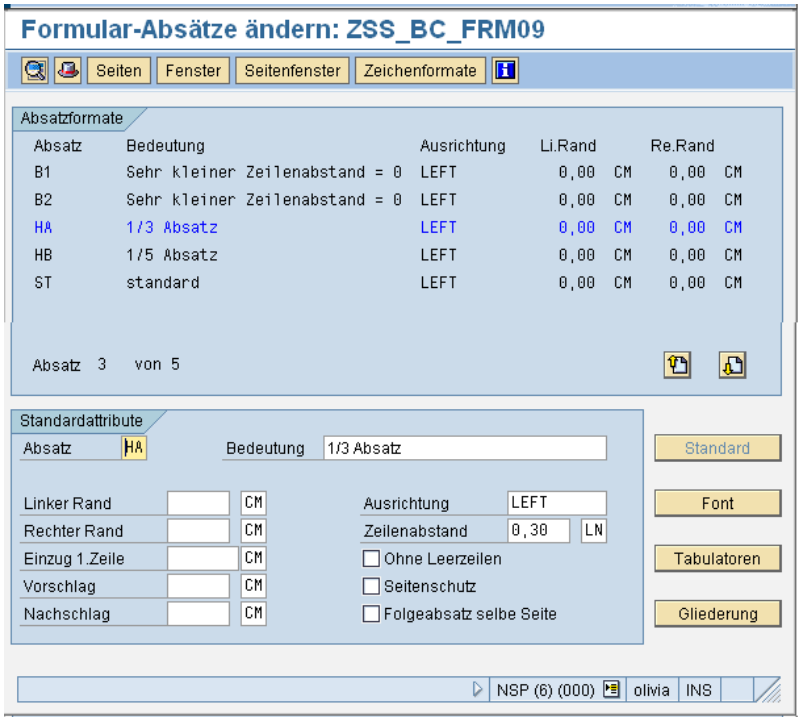
Barcode Implementierung im SAPscript Fenster mit Klarschriftzeile

Um die Position genau unter dem Barcode zu erreichen, wurden Leerzeilen unter die Kommandozeilen mit der Barcode Implementierung eingefügt **(2).** Erst danach folgt die Klarschriftzeile **(3).** Da der Barcode um 40 mm nach rechts verschoben wurde **(1)**, wurde für den Absatz „**B1**“ ein Tab Stopp auf der Position 42mm definiert und vor die Variable **HRT** ein Tabulator eingefügt **(3)**.

Da man nicht immer die exakte, vertikale Position für die Klarschriftzeile mit Standardzeilen (1/6 Zoll) erreicht, muss eventuell ein zusätzliches Absatzformat mit eigens definiertem Zeilenabstand vorgegeben werden. Im obigen Beispiel wurde ein Absatz „HA“ eingefügt **(4)**, der mit einem Zeilenabstand 1/3 LN definiert wurde.

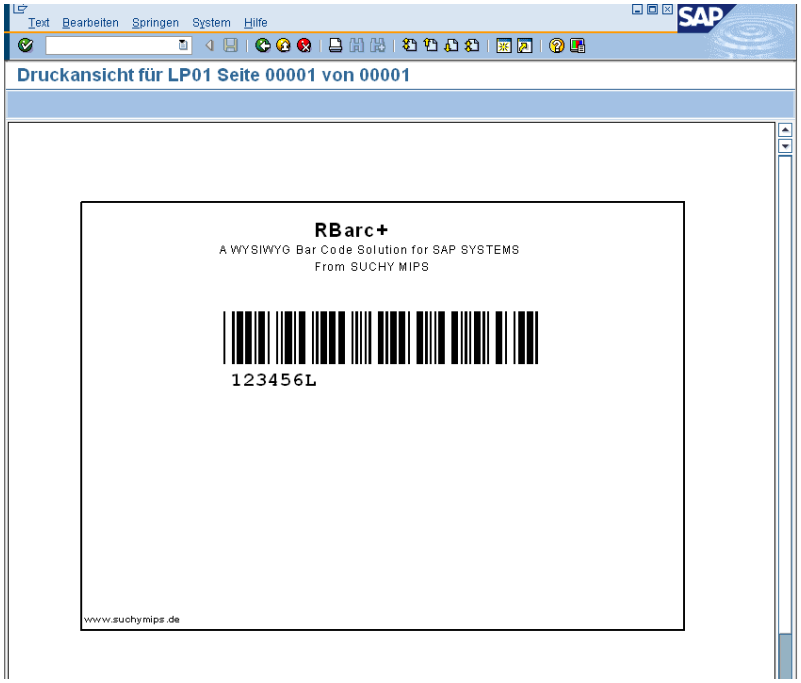


Tabulator Definition für den Absatz „B1“.



Definition des Zeilenabstands 0,3 LN für den Absatz HA.

Das Ergebnis sieht dann in etwa so aus:



Mit den oben beschriebenen Methoden kann die Klarschriftzeile (**HRT**) an jeder beliebigen Stelle ausgegeben werden.

6.3 Einfügen einer gedrehten Klarschriftzeile in ein SAPscript Formular

Wie schon erwähnt, bietet weder SAPscript noch SmartForms die Möglichkeit, einen dynamischen Text zu drehen. Deshalb haben wir für Sie 3 Spezialschriften entwickelt, in der die Zeichen jeweils um 90, 180 bzw. 270 Grad gedreht sind. Wenn Sie alle Installationsschritte durchgeführt haben, sollten diese Schriften bereits auf Ihrem SAP-System installiert sein. Falls nicht, kehren Sie zu den Installationsanweisungen in Kapitel 2 zurück und installieren die mitgelieferten TrueType Schriften **ZHRT90**, **ZHRT180** und **ZHRT270**.

Beispiel HRT90:
1 2 3 4 5

Beispiel HRT180
12345

Beispiel HRT270
1 2 3 4 5

Da der Zeichenvorschub trotz Zeichendrehung nach wie vor vom System in horizontaler Richtung durchgeführt wird, muss für einen vertikal ausgerichteten Text jedes Zeichen aus der Klarschriftzeile in einer separaten Zeile ausgegeben werden:

1
2
3
4
5

Wie Sie sicherlich schon gemerkt haben, ist in manchen Fällen die Reihenfolge der Ziffern umgekehrt, als in der ursprünglichen Zeichenfolge 12345. Deshalb müssen Sie bei der Ausgabe von Text, der 90 bzw. 180 Grad gedreht wurde, die Zeichen der Klarschriftzeile in umgekehrter Reihenfolge ausgeben. SAPscript bietet dafür entsprechende Möglichkeiten, die im Folgenden beschrieben werden.
Nehmen wir an, dass die Klarschriftzeile 10 Zeichen lang ist und in der Variablen **HRT1** gespeichert wurde. Dann muss im Falle eines um 90 oder 180 Grad gedrehten Textes – von oben bzw. von links gesehen – zuerst das zehnte, dann das neunte, dann das achte usw. Zeichen ausgegeben werden. Das können sie mit folgender SAPscript Kodierung erreichen:

Beispiel HRT1 = '0123456789'

Befehlszeilen für 90 Grad	**Ergebnis bei 90 Grad Drehung**
&HRT1+9(1)&	9
&HRT1+8(1)&	8
&HRT1+7(1)&	7
&HRT1+6(1)&	6
&HRT1+5(1)&	6
&HRT1+4(1)&	5
&HRT1+3(1)&	4
&HRT1+2(1)&	3
&HRT1+1(1)&	2
&HRT1(1)&	1

Befehlszeile

&HRT1+9(1)&&HRT1+8(1)&&HRT1+7(1)&&HRT1+6(1)&&HRT1+5(1)&&HRT1+4(1)&&HRT1+3(1)&&HRT1+2(1)&HRT1+1(1)&&HRT1(1)&

Ergebnis bei 180 Grad Drehung

9876543210

Bei einer 270 Grad Drehung muss die Befehlsreihenfolge umgekehrt sein, als bei 90 Grad:

Befehlszeilen für 270 Grad	**Ergebnis bei 270 Grad Drehung**
&HRT1+(1)&	0
&HRT1+1(1)&	1
&HRT1+2(1)&	2
&HRT1+3(1)&	3
&HRT1+4(1)&	4
&HRT1+5(1)&	5
&HRT1+6(1)&	6
&HRT1+7(1)&	7
&HRT1+8(1)&	8
&HRT1+9(1)&	9

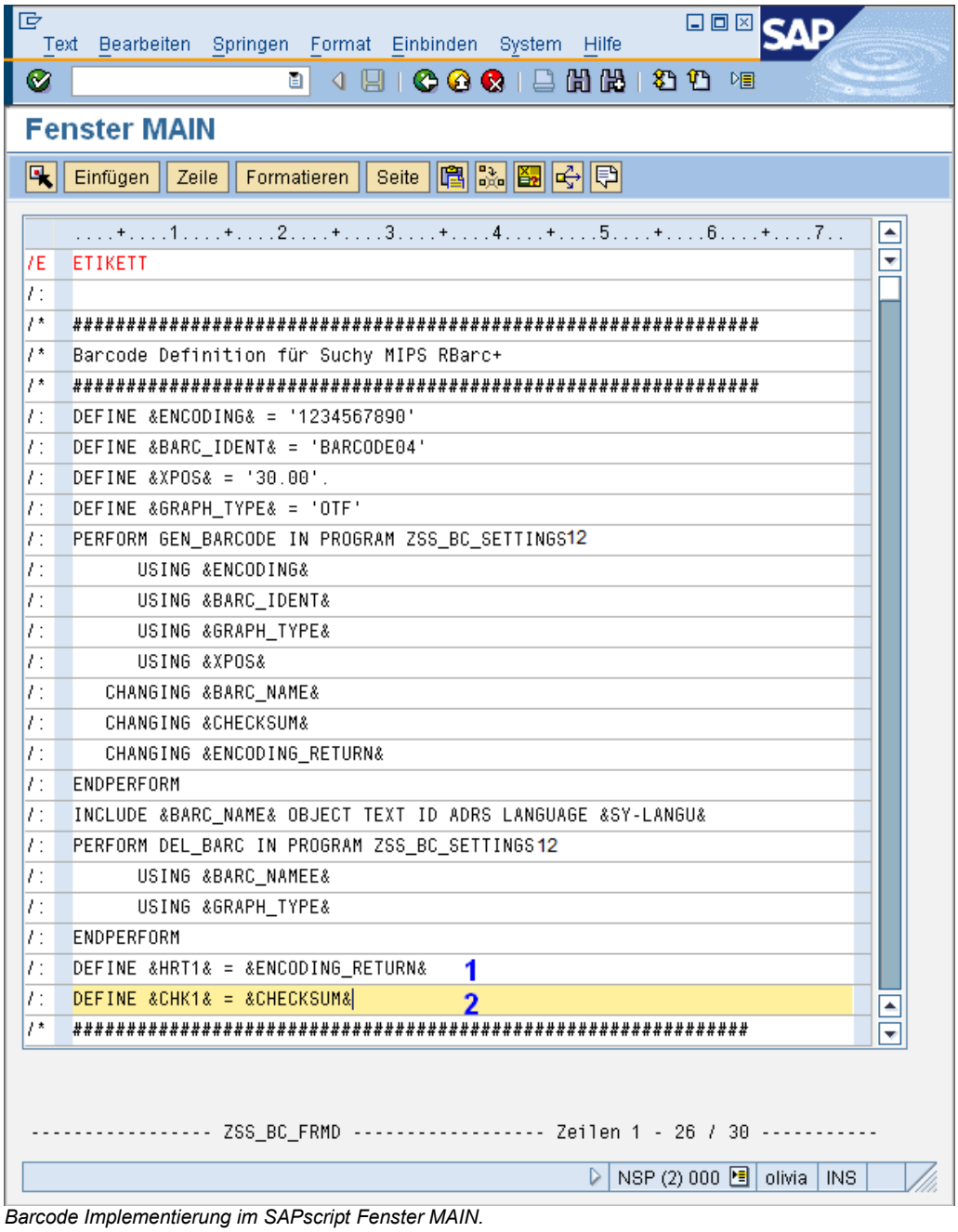
Merke: Sollte es notwendig sein, bei einer 90 bzw. 270 Grad Drehung, die Zeichen enger oder breiter aneinander zu setzen, müssen Sie einen entsprechenden Absatz im SAPscript Formular definieren und diesen den einzelnen Zeilen der Klarschriftzeilenausgabe in der Befehlsspalte zuweisen.

Da es oft schwierig ist, dieses Coding in einem bestehenden SAPscript Fenster unterzubringen, ohne dessen Layout zu beeinflussen, empfehlen wir, für die Klarschriftzeile ein eigenes Fenster zu verwenden. Das hat zusätzlich den Vorteil, dass die Positionierung der Klarschriftzeile bequem über die Fensterposition erreicht werden kann.

Das folgende Beispiel eines SAPscript Formulars mit um 90 Grad gedrehtem Barcode mit Klarschriftzeile soll Ihnen helfen, eigene Implementierungen durchzuführen. Es wird ein Barcode dargestellt, dessen Klarschriftzeile zusätzlich die von RBarc+ zur Laufzeit berechnete Prüfziffer enthält.

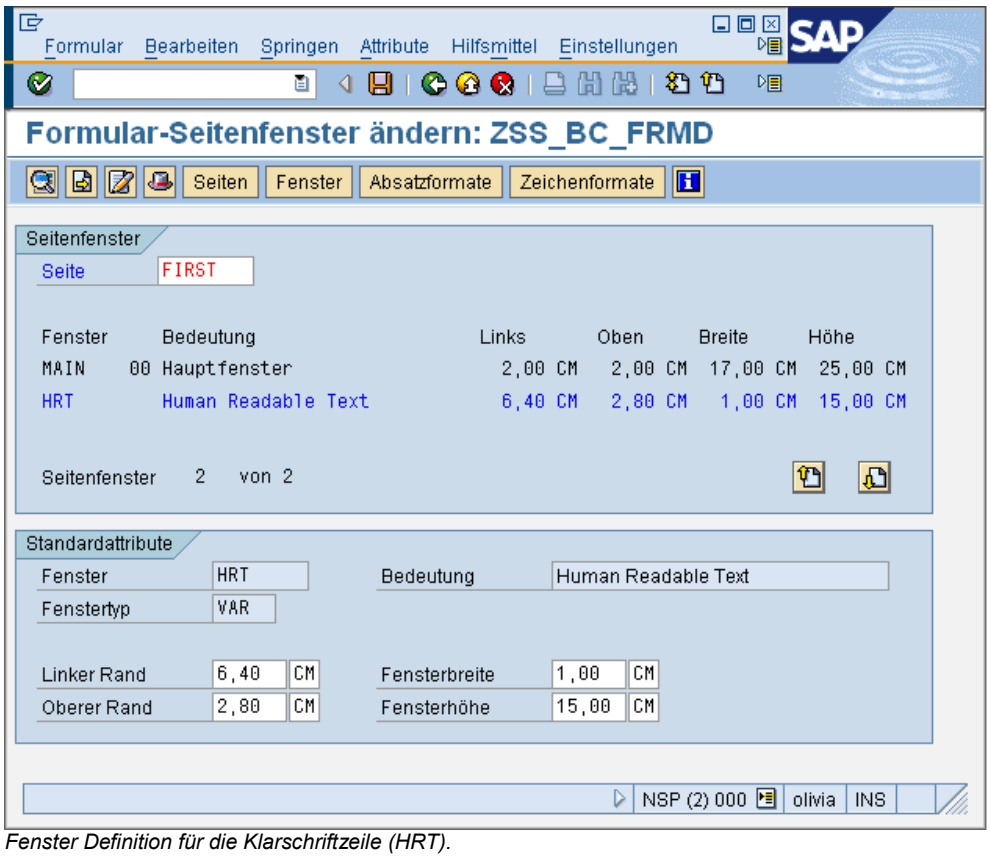
Das Beispielformular besteht aus zwei Fenstern: **MAIN** und **HRT**.

Im Fenster **MAIN** wurde der Barcode definiert, wie in Kapitel 5.1 beschrieben:



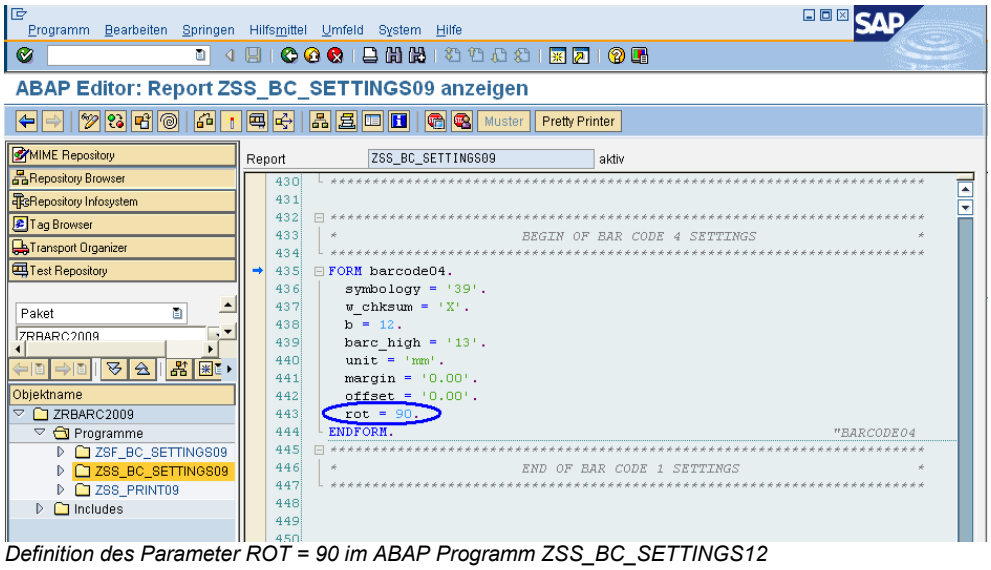
Barcode Implementierung im SAPscript Fenster MAIN.

Da wir die Klarschriftzeile nicht sofort ausgeben wollen, wurde die Standard Rückgabevariable **&ENCODING_RETURN&** in der Variablen **&HRT1&** zwischengespeichert (1). So wird sichergestellt, dass der Wert erhalten bleibt, auch wenn noch weitere Barcodes in dem Formular erzeugt werden sollten. Ähnliches gilt für die Variable **&CHECKSUM&,** welche die aktuelle Prüfziffer enthält (2). Sie wurde in der Variablen **&CHK1&** zwischengespeichert.



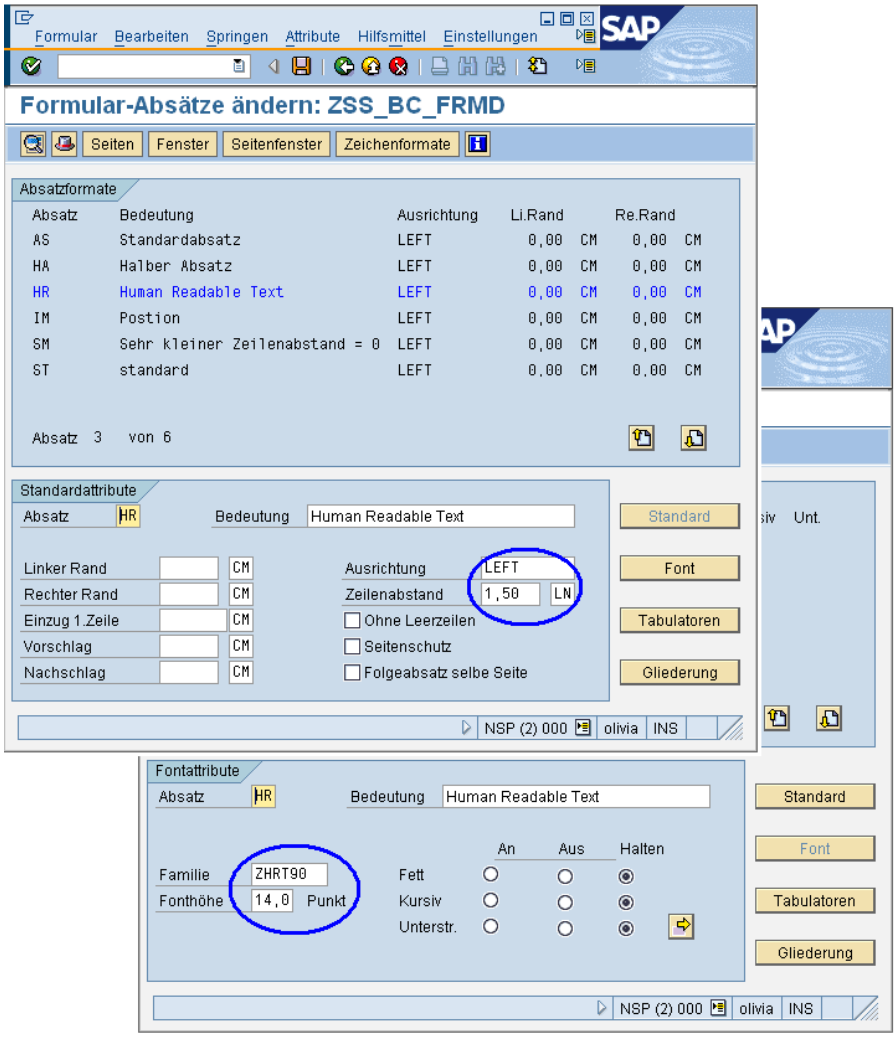
Fenster Definition für die Klarschriftzeile (HRT).

Damit der Barcode um 90 Grad gedreht wird, wurde der Parameter **ROT** im Programm **ZSS_BC_SETTINGS12** entsprechend definiert:



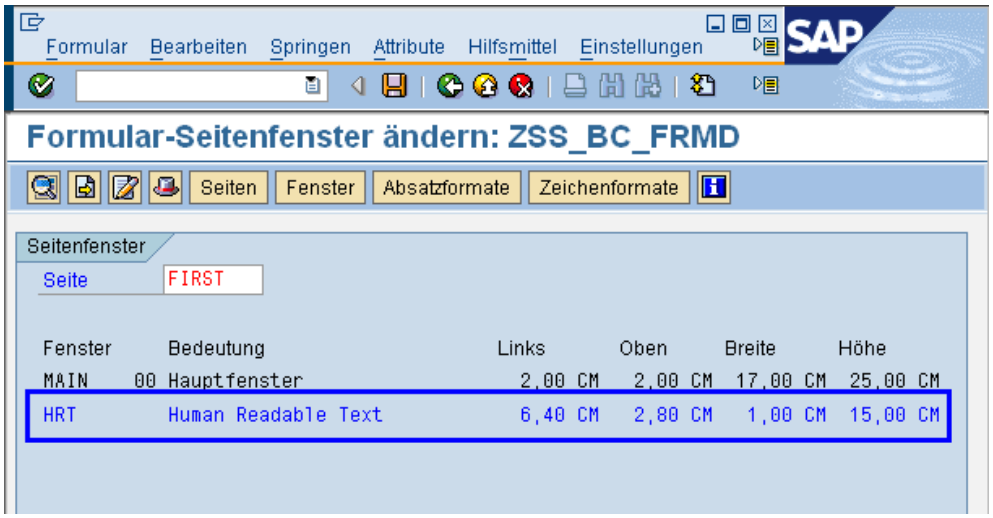
Definition des Parameter ROT = 90 im ABAP Programm ZSS_BC_SETTINGS12

Da die Klarschriftzeile um 90 Grad gedreht werden soll, wurde im SAPscript Formular der Absatz **HR** definiert. In der Absatzdefinition wurde festgelegt, dass der Zeilenabstand für diesen Absatz **1,5 LN** beträgt. Zusätzlich wurde festgelegt, dass die Schriftart **ZHRT90** mit **14 Pt.** Höhe zu verwenden ist.



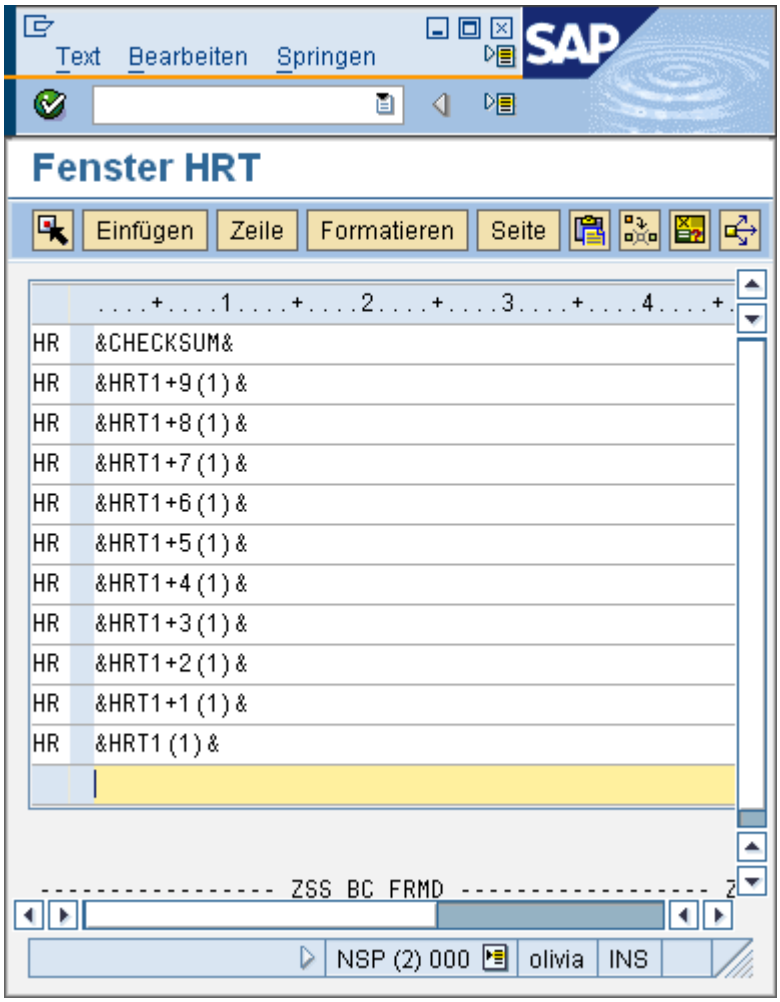
Definition des Absatzes HRT

Das Fenster HRT wurde so positioniert, dass die Klarschriftzeile direkt rechts neben dem Barcode erscheint:



Definition der Seitenposition des Fensters HRT

Im Fenster **HRT** wird die Klarschriftzeile (Variable **HRT1**) Zeichen für Zeichen (von hinten nach vorne) pro Zeile ausgegeben. Jede Zeile ist mit dem Absatz **HR** formatiert.



Merke: Das Ergebnis erscheint im Falle gedrehter Klarschriftzeilen in der SAP Druckvorschau nicht korrekt. Der Grund dafür ist die Tatsache, dass für die Vorschau nicht die angegebenen TrueType Schriften sondern deren Substitute verwendet werden. Sobald sie jedoch das Formular Drucken oder z.B. ein PDF Dokument erzeugen, wird die Klarschriftzeile in gedrehter Form korrekt dargestellt.

Ergebnis der SAP-Druckvorschau

Reales Druckergebnis

Merke: Mit der oben beschriebenen Vorgehensweise lassen sich auch gedrehte Barcodes mit eingebetteter, oder halb eingebetteter Klarschriftzeile gestalten. Definieren Sie dafür die Parameter **MARGIN** und **OFFSET** im Programm **ZSS_BC_SETTINGS12** für den entsprechenden Barcode (das erzeugt einen weißen Einschnitt an der langen Barcodekante) und positionieren Sie die Klarschriftzeile entsprechend

Gedrehter Barcode mit halb eingebetteter Klarschriftzeile.

7 Implementierung einer Barcode Ausgabe in SmartForms

7.1 Einbinden von Barcodes in ein SmartForms Formular

In einem SmartForms Formular können beliebig viele Barcodes eingebunden werden. Da es sich hier um **Knoten** vom Typ **Programmzeilen** und **Grafik** handelt, kann der Barcode an beliebiger Stelle eingefügt werden: in einem bestehenden Fenster, in einem separaten Fenster oder z.B. in einer Schablone oder Tabelle. Bestehende Druckprogramme für ein SmartForms Formular werden **NICHT** geändert. Die Anpassungen betreffen lediglich das Formular selbst (somit ist die RBarc+ Lösung für ein SAP Update geeignet). Um die Anpassungen im Formular so gering wie nur möglich zu halten, werden im SmartForms Formular – ähnlich wie in einem SAPscript Formular - nur die nötigsten Definitionen vorgenommen. Dazu gehören die Informationen: „**was soll kodiert werden**“? und „**wo soll der Barcode positioniert werden**“?. Alle anderen Barcode Eigenschaften, wie **Symbologie**, **Breite**, **Höhe** usw. werden in dem außen liegenden ABAP Programm **ZSF_BC_SETTINGS12** vorgenommen.

Die Logik der Vorgehensweise ist immer die gleiche und richtet sich nach folgendem Prinzip:

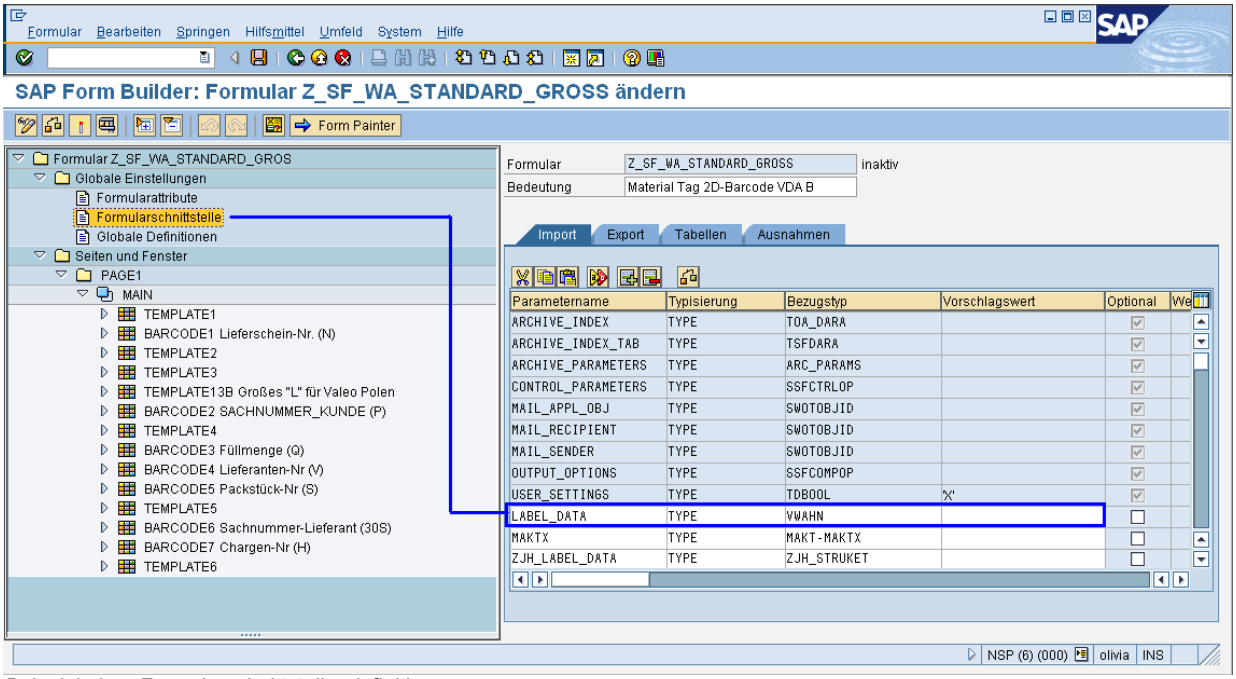
- Einmalige Definition der Barcodeeigenschaften (**Symbologie, Höhe, Breite** usw.) in einer separaten Form Routine im ABAP Programm **ZSF_BC_SETTINGS12**.
- Definition der Variable(n) im SmartForms Formular, die als Barcode kodiert werden soll(en).
- Festlegen einer **Barcode Identifikation** (jeder Barcode in einem Formular muss eindeutig identifizierbar sein).
- Aufruf der Form Routine **GEN_BARCODE** im Programm **ZSF_BC_SETTINGS12** und Übergabe der Parameter (zu der in erster Linie die zu verschlüsselnden Daten und die Barcode Identifikation gehören). Dies geschieht über einen anzulegenden Programmknoten.
- Übernahme der von RBarc+ zurück gelieferten **Rückgabeparameter** (in erster Linie der dynamisch erzeugte **Name** der einzubindenden **Barcodegrafik**).
- Dynamisches Einbinden der **Barcodegrafik** in das Formular über einen **Grafikknoten.**
- Löschen der zuvor erzeugten Barcodegrafik aus dem System über einen weiteren Programmierknoten.

Über die Eigenschaften der Barcodes, wie **Symbologie**, **Höhe**, **Breite** usw. schreiben wir später im **Kapitel 7 „Die Barcode Generierung“**, da dieser Teil identisch für **SAPscript** und **SmartForms** Anwendungen ist.

Auf den folgenden Seiten präsentieren wir ein Screenshot von einem SmartForms Formular, in das ein Barcode eingebunden wurde und erklären diese.

7.1.1 Definition der Variablen

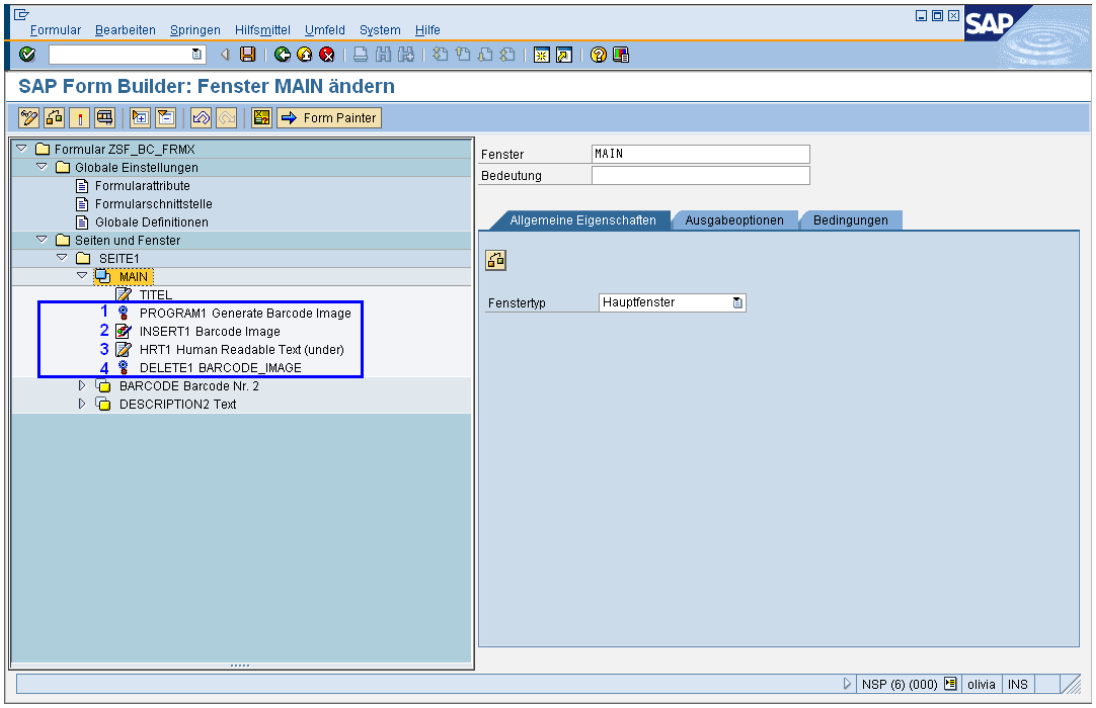
In der Regel werden in produktiver Umgebung wechselnde Daten aus der Prozessverarbeitung kodiert (z.B. Lieferscheinnummer). Diese Daten kommen aus Tabellen bzw. Strukturen, die in der **Formularschnittstelle** definiert wurden. Ein Beispiel für eine solche Definition in der **Formularschnittstelle** könnte die Struktur **LABEL_DATA** sein, die Bezug auf die Tabelle **VWAHN** nimmt. In unserem Beispiel werden wir das Feld **VBELN** aus der Struktur **LABEL_DATA** als Barcode verschlüsseln.



Beispiel einer Formularschnittstellendefinition.

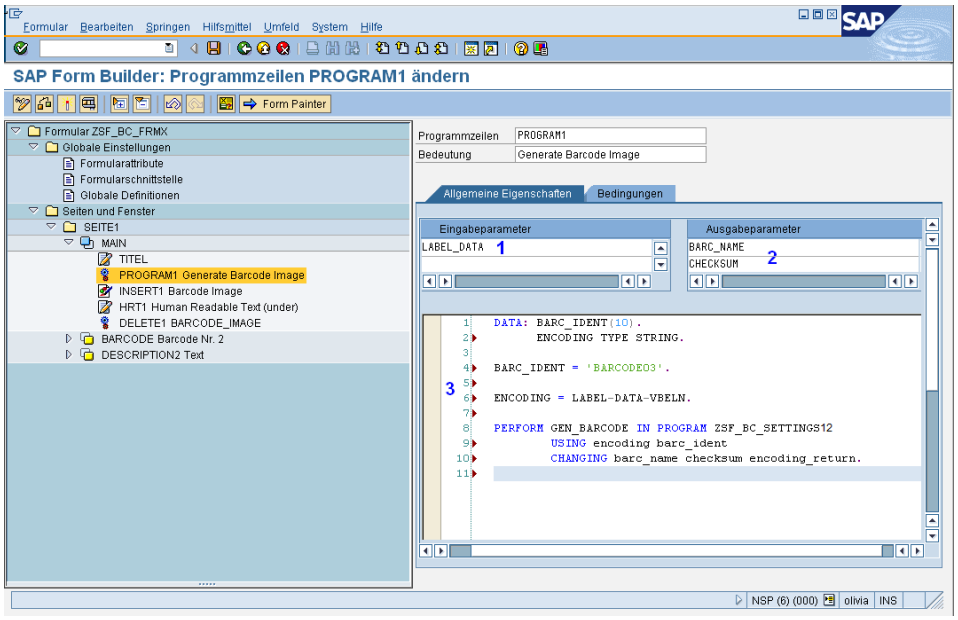
7.1.2 Formularaufbau für Barcode Erzeugung.

Auf dem folgenden Bild sehen Sie die Formularstruktur mit den für die Barcodeausgabe notwendigen Knoten. In diesem Fall wurden alle Knoten im Fenster **MAIN** eingefügt. Wie aber schon erwähnt, können Sie dazu ein beliebiges Fenster Ihrer Wahl benutzen.



Knoten 1

Programmknoten. In diesem Knoten wird die als Barcode zu kodierende Variable festgelegt und die Form Routine **GEN_BARCODE** im Programm **ZSF_BC_SETTINGS12** aufgerufen.



Programmlisting der Barcode Implementierung

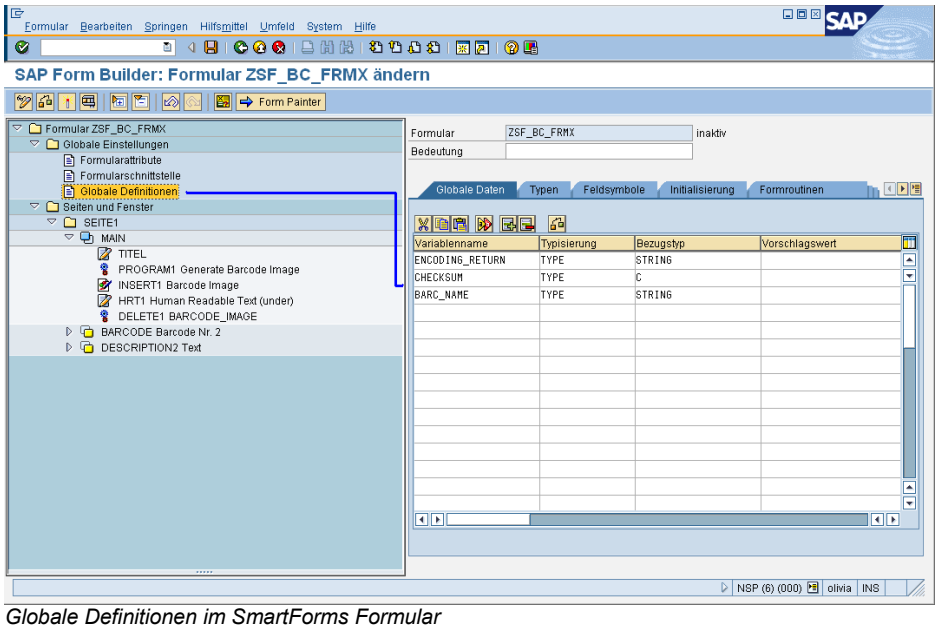
Als **Eingabeparameter** wurde die Struktur **LABEL_DATA** definiert (**1**). Das ist notwendig, damit die Variable von diesem Knoten heraus „gesehen“ wird.

Als Ausgabeparameter wurden die Variablen **BARC_NAME**, **CHECKSUM** und **ENCODING_RETURN** (die Letztere ist auf dem Screenshot nicht sichtbar) definiert (**2**). Das ist wichtig, denn diese Variablen, die vom Programm **ZSF_BC_SETTINGS12** zurückgegeben werden, werden in den weiteren Formularknoten benötigt. Damit diese Variablen überall sichtbar sind, müssen sie in den **globalen Definitionen** des SmartForms Formulars wie folgt definiert sein:

BARC_NAME TYPE STRING

CHECKSUM TYPE C

ENCODING_RETURN TYPE STRING



Globale Definitionen im SmartForms Formular

Programmlisting:

Zeilen 1 und 2:

In der DATA Anweisung werden die Variablen **BARC_IDENT(10)** und **ENCODING** definiert.

Zeile 4

BARC_IDENT = 'BARCODE03'

Hier wird eine Barcode Identifikation vergeben, die über die Variable **BARC_IDENT** an RBarc+ übermittelt wird. Die maximale Länge für die Variable beträgt 10 Zeichen. Jeder Barcode im Formular muss eine eineindeutige Identifikation besitzen, deshalb empfiehlt es sich, einen Zähler als diskriminierendes Mittel zu verwenden, wie im obigen Beispiel ‚03'.

Merke: Die Barcode Identifikation muss dem Namen der Form Routine im Programm **ZSF_BC_SETTINGS12** entsprechen, in der die Barcode Eigenschaften definiert werden. Ggf. müssen Sie die Form Routine für einen neuen Barcode im Programm **ZSF_BC_SETTINGS12** selbst erstellen. Bei der Auslieferung sind im Programm **ZSF_BC_SETTINGS12** 5 Form Routinen vordefiniert: **BARCODE01, BARCODE02, BARCODE03, BARCODE04** und **BARCODE05.** Die dort definierten Barcode Parameter können Sie jederzeit ändern.

Zeile 6

ENCODING = LABEL_DATA-VBELN

In dieser Zeile wird der Variablen **ENCODING** der Wert des Feldes **VBELN** aus der Struktur **LABEL_DATA** zugeordnet. Bitte beachten Sie, dass Barcodes in der Regel nur Daten vom Typ **TEXT** verschlüsseln können, so dass Zahlen, besonders solche, die einen Punkt oder ein Komma enthalten, vorher in **TEXT** umgewandelt werden müssen. Das erreichen Sie, in dem Sie die entsprechenden Variablen vom Typ Integer oder „F" in eine Textvariable schreiben.

Zeilen 8 – 9

```
PERFORM GEN_BARCODE IN PROGRAM ZSF_BC_SETTINGS12
        USING encoding barc_ident
```

Aufruf des der Form Routine **GEN_BARCODE** im Programm **ZSF_BC_SETTINGS12** und Übergabe der Parameter **ENCODING** und **BARC_IDENT**.

Merke: Es ist prinzipiell möglich, für jedes Formular ein eigenes ABAP Programm mit dort definierten Barcode Eigenschaften zu verwenden. In diesem Fall kopieren Sie das Programm **ZSF_BC_SETTINGS12** in ein anderes Programm und rufen dieses im SmartForms Formular auf. Achten Sie jedoch dabei darauf, dass Sie die Namen der Variablen nicht ändern, da sonst das Programm nicht einwandfrei funktionieren würde.

Zeile 10

CHANGING barc_name checksum encoding_return.

In **Zeile 10** werden die Variablen für Rückgabewerte aus Rbarc+ definiert. Diese Namen dürfen nicht verändert werden. Falls Sie die Werte dieser Variablen zu anderweitiger Verwendung benötigen, empfiehlt es sich, diese in andere, selbst definierte Variablen zu schreiben.

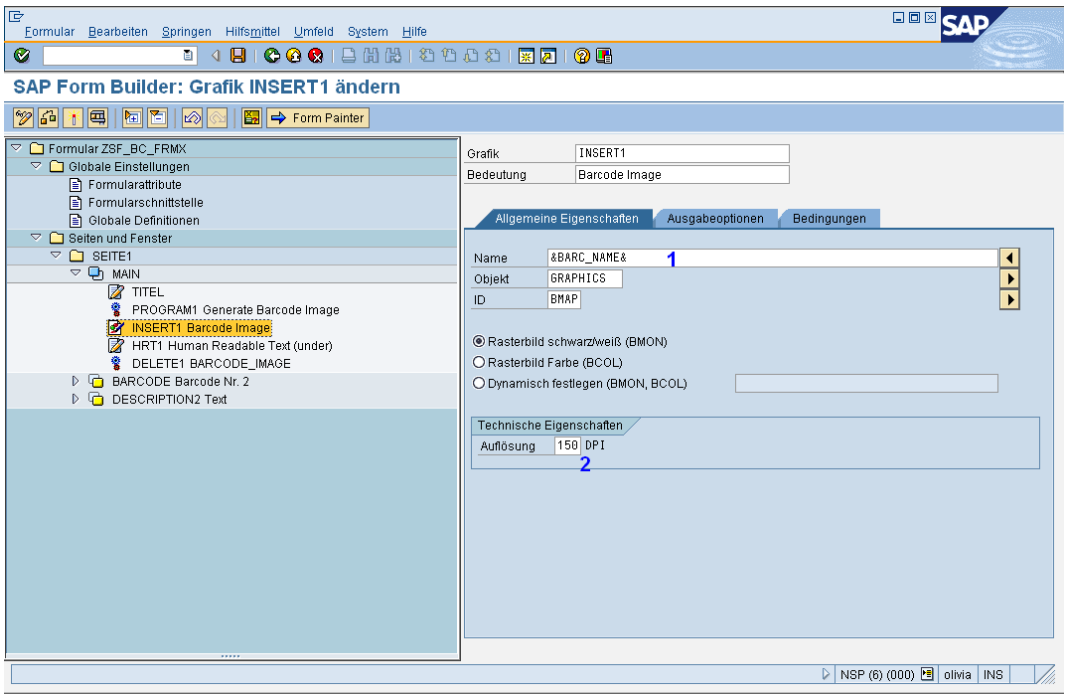
BARC_NAME – Name der von RBarc+ erzeugten Grafik mit dem Barcodebild.
CHECKSUM – die von RBarc errechnete Prüfziffer (falls erforderlich).
ENCODING_RETURN – gibt an, was wirklich verschlüsselt wurde.

Merke: Der zurückgegebene Wert für **ENCODING_RETURN** kann manchmal vom übergebenen Wert für **ENCODING** abweichen, z.B. dann, wenn bestimmte Optionen, wie das Löschen führender Nullen im Programm **ZSF_BC_SETTINGS12** eingestellt wurde.

.

Knoten 2

Grafikknoten. In diesem Knoten wird die von RBarc+ erzeugte Barcodegrafik dynamisch eingefügt. Deshalb steht im Feld *Name* kein fester Name, sondern die Variable **BARC_NAME (1)**. Diese Variable enthält den zur Laufzeit erzeugten Namen der im System abgelegten Barcodegrafik. Dieser Name wird über die **Ausgabeparameter** des vorherigen Programmknoten veröffentlicht. Damit das ganze auch funktioniert, muss die Variable **BARC_NAME** als **TYPE STRING** in den **globalen Definitionen** des SmartForms Formulars angegeben werden.



Das Feld *Object* muss mit dem Wert **GRAPHICS** und das Feld *ID* mit dem Wert **BMAP** gefüllt werden.

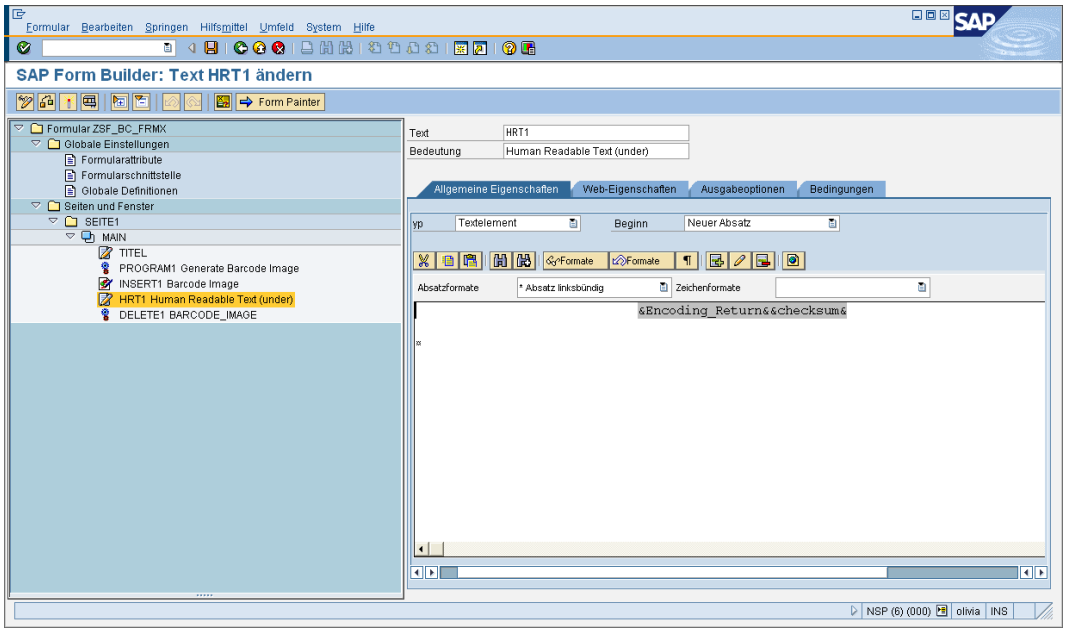
Im Feld *Auflösung* muss die tatsächliche Auflösung angegeben werden, mit dem die Grafik erzeugt wurde.

Merke: Die Auflösung, mit der die Barcodegrafik erzeugt wird, ist im Programm **ZSF_BC_SETTINGS12** in Parameter **RES** festgelegt. Falls die Angaben im Programm und im Formular nicht übereinstimmen, wird das Ergebnis nicht den gewünschten Erwartungen entsprechen: die Grafik wird entweder zu groß oder zu klein dargestellt.

Merke: Für jeden Barcode kann – je nach Bedarf – eine eigene Auflösung angegeben werden. Mit RBarc+ können beliebige Auflösungen definiert werden. Falls Sie Dokumente mit Barcodes drucken, sollten Sie jedoch darauf achten, dass das Ausgabegerät die von Ihnen gewählte Grafikauflösung auch wirklich unterstützt. Ist das nicht der Fall, wird die Barcodegrafik mit falscher Größe gedruckt. Gängige Laserdrucker unterstützen folgende Grafikauflösungen: 75dpi, 150dpi, 300dpi und 600dpi.

Knoten 3

Textknoten. In diesem Knoten wird die Klarschriftzeile (**HRT** = **H**uman **R**eadable **T**ext) eingefügt.

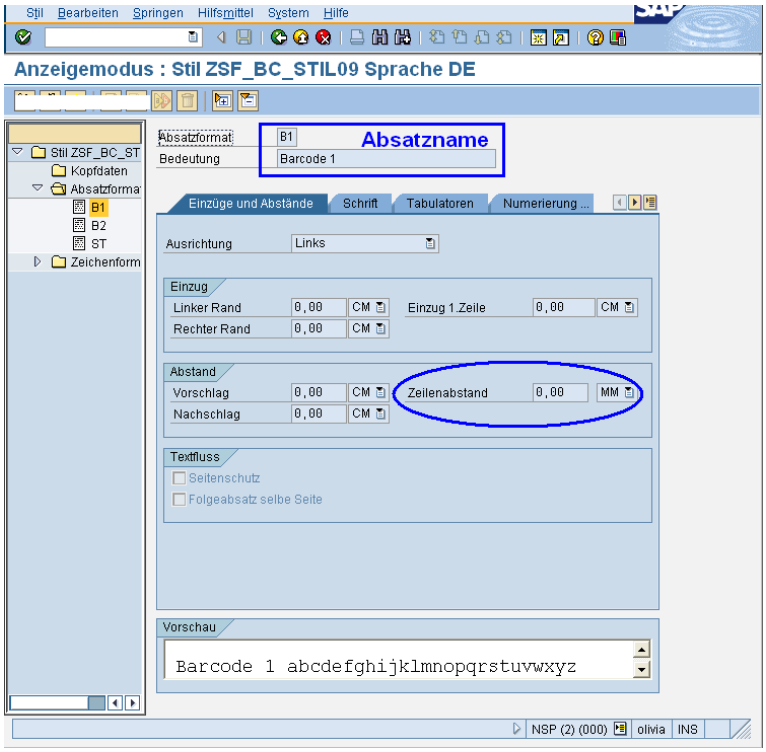


&ENCODING_RETURN&&CHECKSUM&

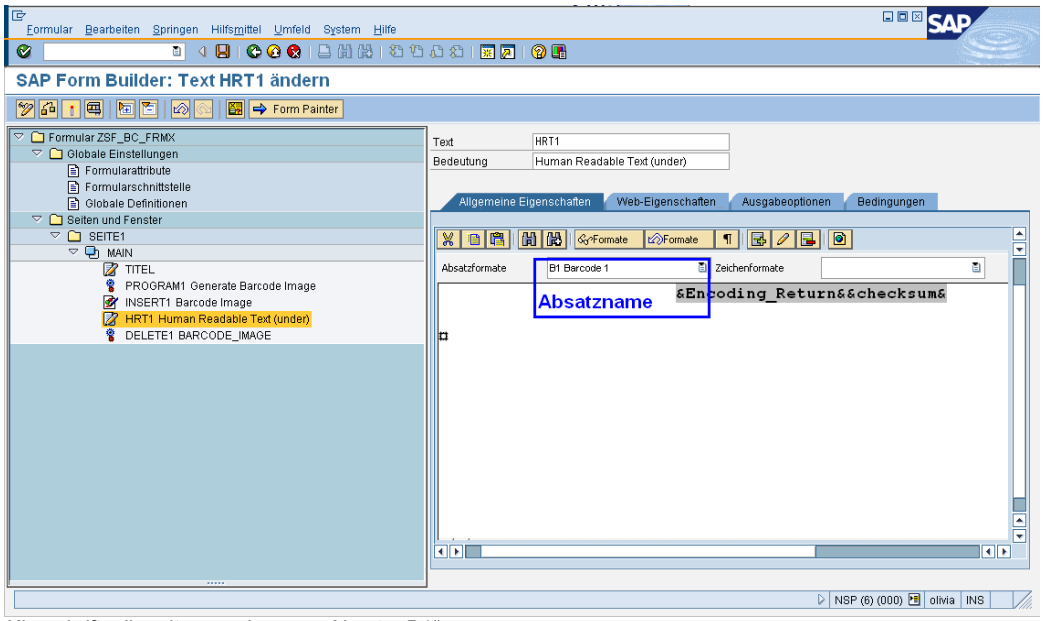
Als Klarschriftzeile wird die Variable **ENCODING_RETURN** unmittelbar gefolgt von **CHECKSUM** ausgegeben. Die horizontale Position der Klarschriftzeile wird mit Leerzeichen bzw. Tabulatoren erreicht.

In der Regel erscheint die Klarschriftzeile eine Zeile unter der Barcodegrafik:

Manchmal besteht jedoch die Notwendigkeit, die Klarschriftzeile in den Barcode „einzubetten“. Für diesen Fall müssen Sie für den Barcode einen Einschnitt (**Margin**) im Programm **ZSF_BC_SETTINGS12** definieren und einen SmartForms Stil anlegen (bzw. einen bestehenden Stil erweitern) und dort einen Absatz mit 0 oder z.B. 0,1 MM Zeilenabstand – je nach Bedarf - definieren. Der Stil muss dem Textknoten und der definierte Absatz der Klarschriftzeile zugewiesen werden:

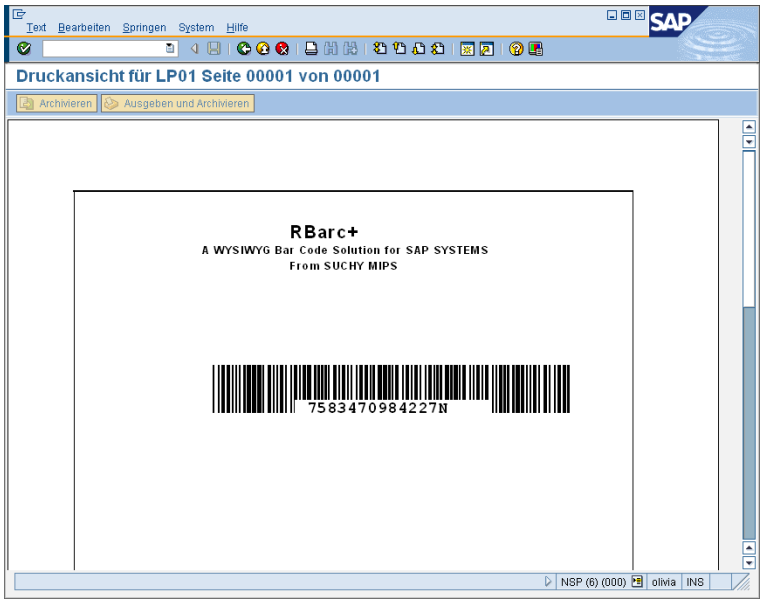


Beispiel eines SmartForms Stils mit Absatz B1 und Zeilenabstand 0 MM.



Klarschriftzeile mit zugewiesenen Absatz „B1“.

Das Ergebnis sieht dann in etwa so aus:

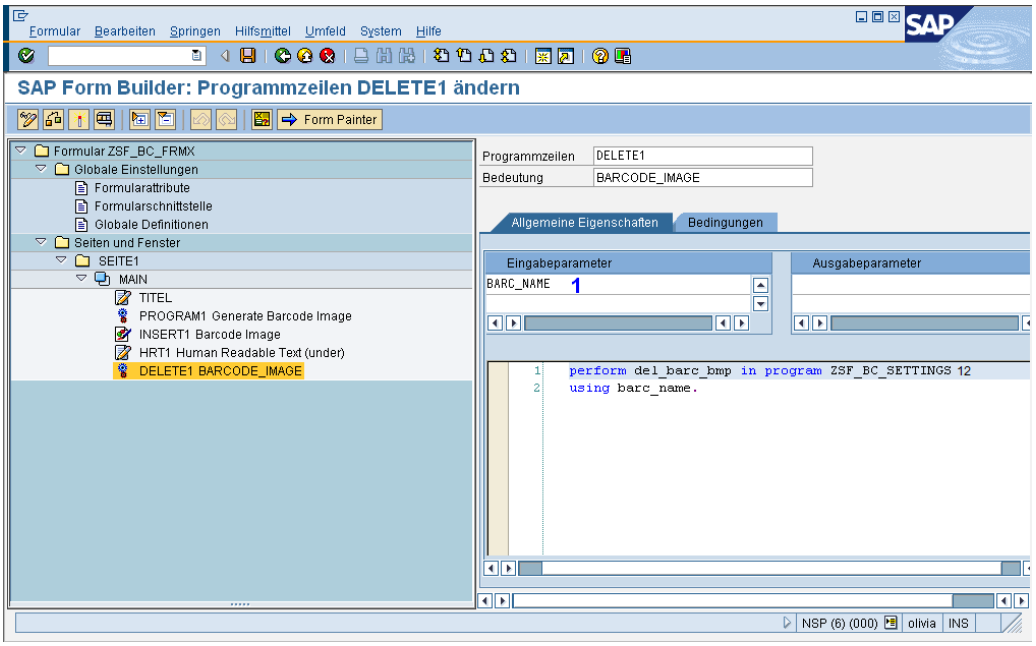


Merke: In den meisten Fällen entspricht der Wert der Variablen **ENCODING_RETURN** der Variablen **ENCODING**. Da RBarc+ jedoch zusätzliche Möglichkeiten zur Verfügung stellt, den Eingabewert (**ENCODING**) zu manipulieren (z.B. können führende Leerzeichen und Nullen entfernt werden), kann es sein, dass der Barcode einen anderen Wert verschlüsselt, als mit **ENCODING** angegeben. Aus diesem Grund gibt RBarc+ in der Variablen **ENCODING_RETURN** den tatsächlich verschlüsselten Wert zurück. Beispiel: **ENDODING = 00012345**. Da im Programm **ZSF_BC_SETTINGS12** die Unterdrückung führender Nullen eingeschaltet wurde, wird tatsächlich nur **12345** verschlüsselt. Der Rückgabeparameter **ENCODING_RETURN** enthält nun auch den Wert **12345**, der als **HRT** verwendet werden sollte.

Merke: Es ist grundsätzlich möglich, die Klarschriftzeile völlig getrennt vom Barcode in einem separaten Fenster auszugeben. Das hat den Vorteil, dass man keine speziellen Absatzformate definieren muss, sondern die gewünschte Position über die Fensterposition erreicht. Falls im gleichen Formular mehrere Barcodes implementiert werden, sollten Sie die originalen Rückgabevariablen in eigene Variablen zwischenspeichern, damit diese vom nächsten RBarc+ Aufruf nicht überschrieben werden, sondern über die gesamte Laufzeit erhalten bleiben. Z.B. definieren Sie für 3 verschiedene Barcodes **HRT1**, **HRT2** und **HRT3** und schreiben den Wert von **ENCODING_RETURN** gleich im ersten Programmknoten von jedem Barcode in die jeweilige **HRTx** Variable. Auf diese Weise bleibt jede Klarschriftzeile während des gesamten Formularablaufs erhalten und kann jederzeit eingesetzt werden

Knoten 5

Programmknoten. In diesem Knoten wird der zuvor erzeugte Barcode gelöscht. Der Barcode darf natürlich aus dem System erst gelöscht werden, nachdem er bereits ins Formular eingefügt wurde. Achten Sie deshalb darauf, dass in der Programmablauflogik dieser Knoten erst nach dem Grafikknoten kommt.



Definieren Sie **BARC_NAME** als **Eingabeparameter (1)**.

Programmlisting.

Zeilen 1 - 2

```
perform del_barc_bmp in program ZSF_BC_SETTINGS12
using barc_name.
```

Diese Anweisung ruft die Form Routine **del_barc_bmp** aus dem Programm **ZSF_BC_SETTINGS12** auf, die den zuvor erzeugten Barcode mit dem Namen **BARC_NAME** aus dem System löscht.

Merke: Falls Sie mehrere Barcodes in einem Formular ausgeben wollen und dabei die Klarschriftzeile in separaten Fenstern definiert haben, haben Sie sicherlich den Wert der Klarschriftzeile in einer eigenen Variablen (z.B. **HRT1**) zwischengespeichert. In diesem Fall müssen Sie statt **BARC_NAME** diese Variable als Eingabeparameter (1) und als USING Parameter in der Programmzeile 2 angeben.

7.2 Definition der Barcode Eigenschaften.

Wie schon in vorherigen Kapiteln erwähnt, werden die Barcode Eigenschaften in einem separaten ABAP Programm definiert. Diese Definitionen sind für **SAPscript** und **SmartForms** identisch. Da jedoch SAPscript eine andere Schnittstelle für Datenübergabe als SmartForms vorsieht, wird für SAPscript das Programm **ZSS_BC_SETTINGS12** und für SmartForms das Programm **ZSF_BC_SETTINGS12** verwendet. Achten Sie darauf, dass Sie jeweils das richtige Programm benutzen. Wir werden uns im Folgenden nur auf die Form Routinen beziehen, die für beide Programme gleich sind. Die genaue beschreibung der Definition von Barcode Eigenschaften finden Sie im **Kapitel 7 „Die Barcode Generierung“**,

7.3 Einfügen einer gedrehten Klarschriftzeile in ein SmartForms Formular

Ähnlich wie SAPscript, bietet auch SmartForms keine Möglichkeit, dynamische Texte gedreht auszugeben. Für diejenigen, die sich nur für SmartForms interessieren und Kapitel 5.3 übersprangen, wiederholen wir noch einmal die unter 5.3 enthaltenen Informationen. Falls Sie jedoch Kapitel 5.3. bereits gelesen haben, können Sie Kapitel 6.3 überspringen und mit dem darauf folgenden Kapitel fortfahren.

7.3.1 TrueType Schriften mit gedrehten Zeichen.

Um den Druck von gedrehten Klarschriftzeilen zu ermöglichen, haben wir für Sie 3 Spezialschriften entwickelt, in der die Zeichen jeweils um **90**, **180** bzw. **270** Grad gedreht sind. Wenn Sie alle Installationsschritte durchgeführt haben, sollten diese Schriften bereits auf Ihrem SAP System installiert sein. Falls nicht, kehren Sie bitte zu den Installationsanweisungen zurück und installieren Sie die mitgelieferten TrueType Schriften **ZHRT90**, **ZHRT180** und **ZHRT270**.

Beispiel HRT90:
1 2 3 4 5

Beispiel HRT180
12345

Beispiel HRT270
1 2 3 4 5

Da der Zeichenvorschub trotz Schriftdrehung nach wie vor vom System in horizontaler Richtung durchgeführt wird, muss für einen vertikal ausgerichteten Text jedes Zeichen aus der Klarschriftzeile in einer separaten Zeile ausgegeben werden:

1
2
3
4
5

Wie Sie sicherlich schon gemerkt haben, ist in diesem Fall die Reihenfolge der Ziffern umgekehrt, als in der ursprünglichen Zeichenfolge 12345. Deshalb müssen Sie bei der Ausgabe von Text, der **90** bzw. **180** Grad gedreht wurde, die Zeichen der Klarschriftzeile in umgekehrter Reihenfolge ausgeben. SAPscript bietet dafür entsprechende Möglichkeiten, die im Folgenden beschrieben werden.
Nehmen wir an, dass die Klarschriftzeile 10 Zeichen lang ist und in der Variablen **HRT1** gespeichert wurde. Dann muss im Falle eines um 90 oder 180 Grad gedrehten Textes – von oben bzw. von links gesehen – zuerst das zehnte, dann das neunte, dann das achte usw. Zeichen gedruckt werden. Das können sie mit folgender Kodierung erreichen:

(Beispiel HRT1 = ‘0123456789‘)

Befehlszeilen für 90 Grad	**Ergebnis bei 90 Grad Drehung**
&HRT1+9(1)&	9
&HRT1+8(1)&	8
&HRT1+7(1)&	7
&HRT1+6(1)&	6
&HRT1+5(1)&	6
&HRT1+4(1)&	5
&HRT1+3(1)&	4
&HRT1+2(1)&	3
&HRT1+1(1)&	2
&HRT1(1)&	1

Befehlszeile

&HRT1+9(1)&&HRT1+8(1)&&HRT1+7(1)&&HRT1+6(1)&&HRT1+5(1)&&HRT1+4(1)&&HRT1+3(1)&&HRT1+2(1)&HRT1+1(1)&&HRT1(1)&

Ergebnis bei 180 Grad Drehung

9876543210

Bei einer 270 Grad Drehung muss die Befehlsreihenfolge umgekehrt sein, als bei 90 Grad:

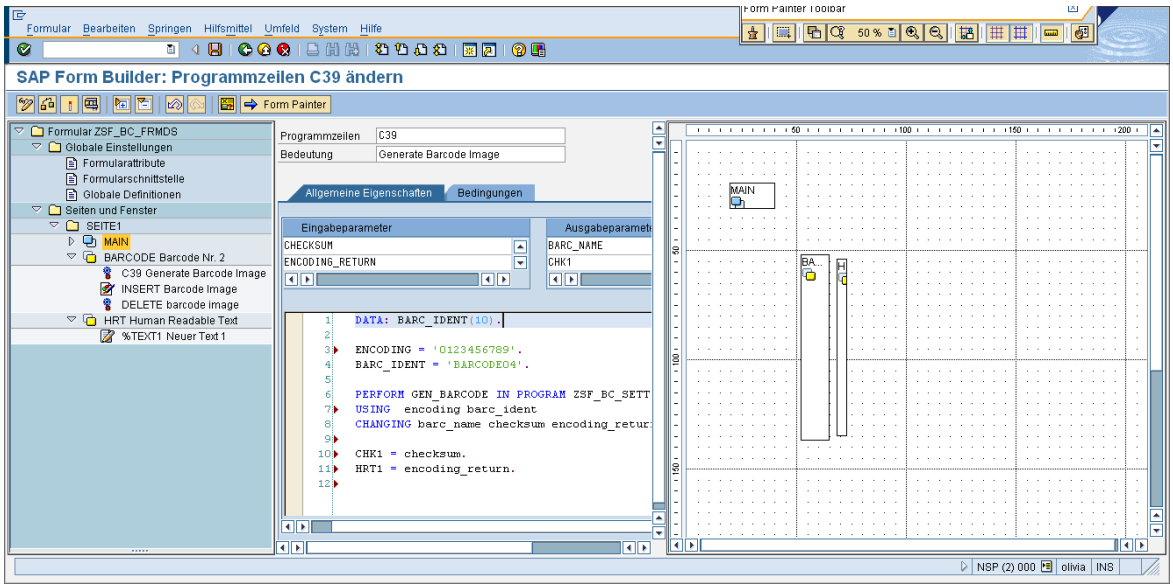
Befehlszeilen für 270 Grad	**Ergebnis bei 270 Grad Drehung**
&HRT1+(1)&	0
&HRT1+1(1)&	1
&HRT1+2(1)&	2
&HRT1+3(1)&	3
&HRT1+4(1)&	4
&HRT1+5(1)&	5
&HRT1+6(1)&	6
&HRT1+7(1)&	7
&HRT1+8(1)&	8
&HRT1+9(1)&	9

Merke: Sollte es notwendig sein, bei einer 90 bzw. 270 Grad Drehung die Zeichen enger oder breiter aneinander zu setzen, müssen Sie einen entsprechenden Absatz im dazugehörigen Stil definieren und damit alle Absätze der Zeichen aus der Klarschriftzeile formatieren.

Da es oft schwierig ist, diese Kodierung in einem bestehenden SmartForms Fenster unterzubringen, ohne dessen Layout zu beeinflussen, empfehlen wir, für die Klarschriftzeile ein eigenes Fenster zu verwenden. Das hat zusätzlich den Vorteil, dass die Positionierung der Klarschriftzeile bequem über die Fensterposition erreicht werden kann.

Das folgende Beispiel eines SmartForms Formulars mit um 90 Grad gedrehtem Barcode mit Klarschriftzeile soll Ihnen helfen, eigene Implementierungen durchzuführen. Es wird ein Barcode dargestellt, dessen Klarschriftzeile zusätzlich die von RBarc+ zur Laufzeit berechnete Prüfziffer enthält.

Das Beispiel Formular besteht aus drei Fenstern: **MAIN. BARCODE** und **HRT**, wobei **MAIN** für dieses Beispiel ohne Bedeutung ist.

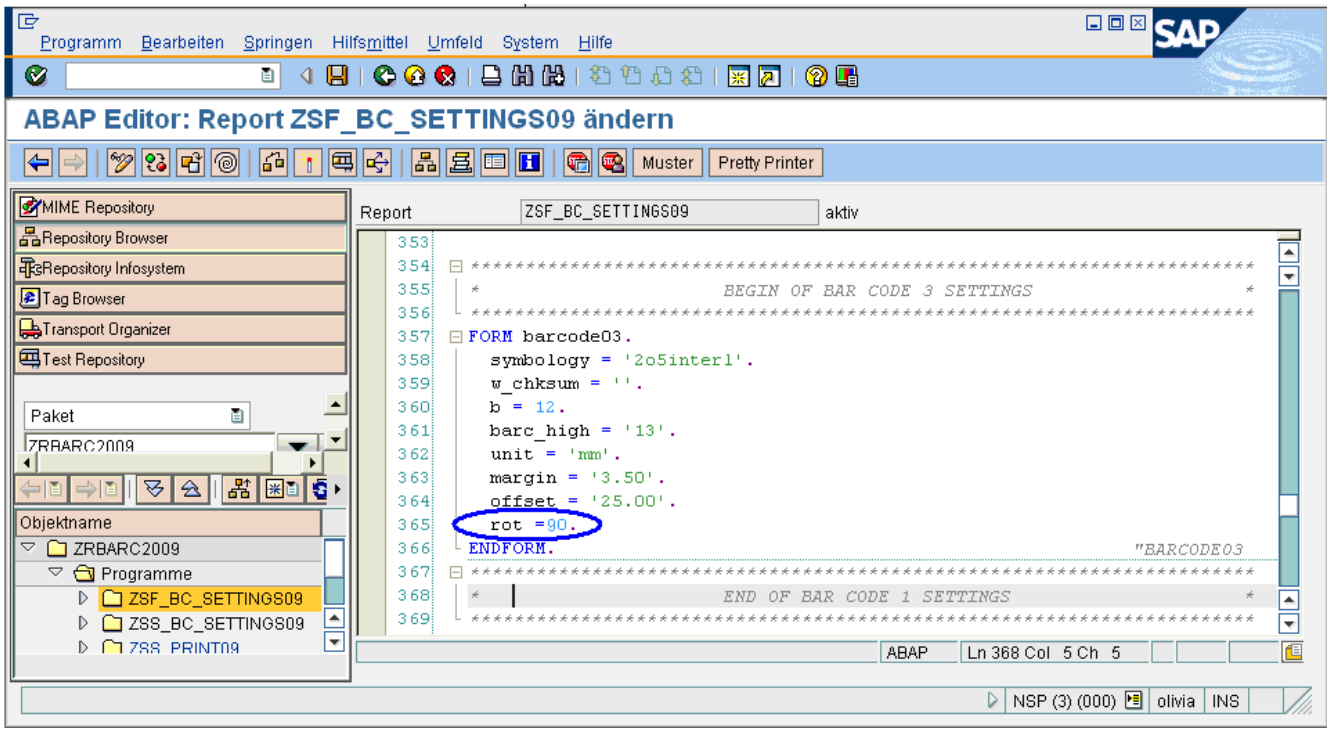


SmartForms Formular mit gedrehtem Barcode und Klarschriftzeile

Im Fenster **BARCODE** wurde der Barcode definiert, wie in Kapitel 6.1 beschrieben.
Das Fenster **HRT** wurde so positioniert, dass die Klarschriftzeile direkt rechts neben dem Barcode erscheint.

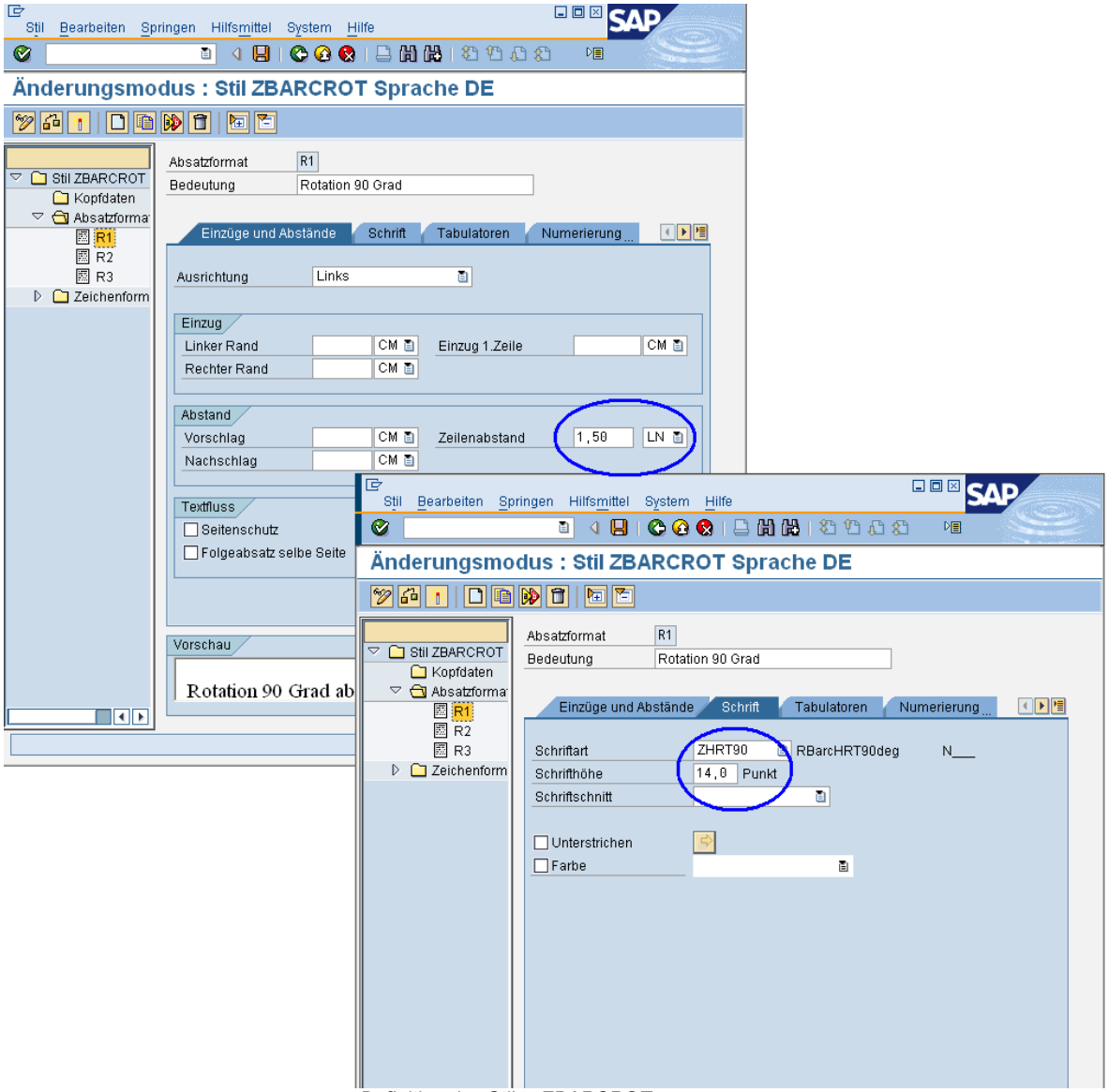
Da wir die Klarschriftzeile nicht sofort ausgeben wollen, wurde die Standard Rückgabevariable **ENCODING_RETURN** in der Variablen **HRT1** zwischengespeichert (1). So wird sichergestellt, dass der Wert erhalten bleibt, auch wenn noch weitere Barcodes in dem Formular erzeugt werden sollten. Ähnliches gilt für die Variable **CHECKSUM**, welche die aktuelle Prüfziffer enthält (2). Sie wird in der Variablen **CHK1** zwischengespeichert.

Damit der Barcode um 90 Grad gedreht wird, wurde der Parameter **ROT** im Programm **ZSF_BC_SETTINGS12** entsprechend definiert:



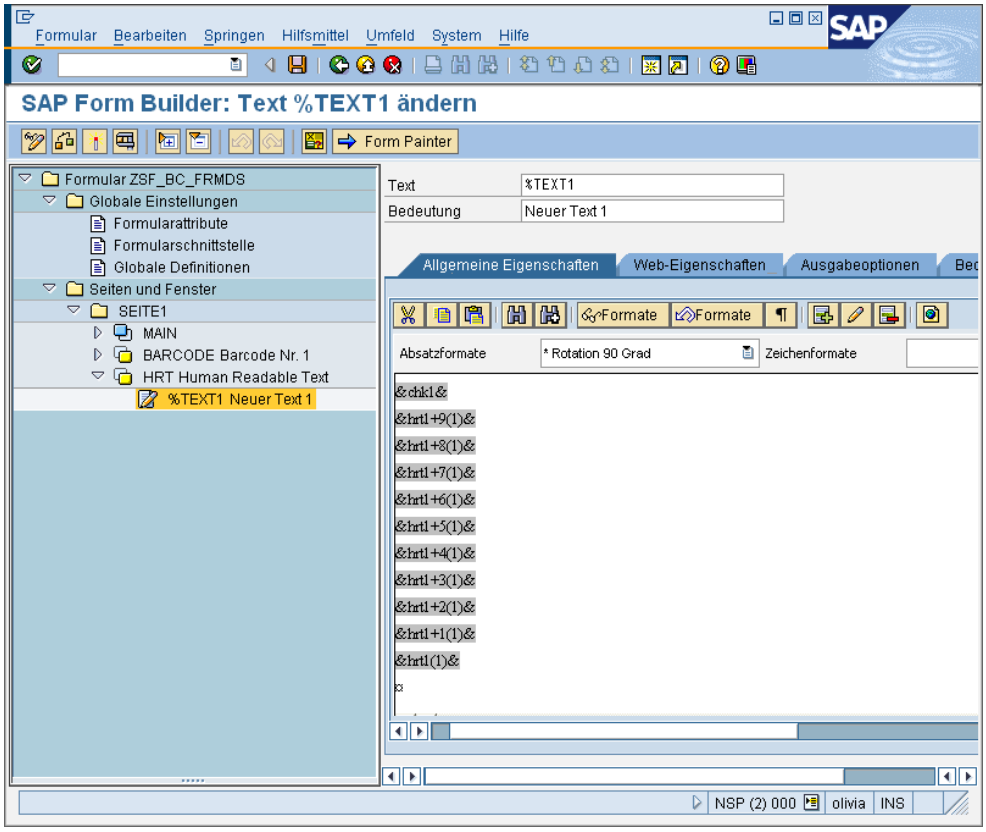
Definition des Parameter ROT = 90 im ABAP Programm ZSF_BC_SETTINGS12

Da die Klarschriftzeile um 90 Grad gedreht werden soll, wurde ein Stil mit dem Namen **ZBARCROT** angelegt. Dort wurden die Absätze **R1** – für 90 Grad, **R2** für 180 Grad und **R3** für 270 Grad Drehung angelegt. Den Absätzen wurde sowohl die jeweils passende TrueType Schrift **ZHRT90**, **ZHRT180** bzw. **ZHRT270** in der Größe **14 Pt**. Zugewiesen, als auch ein Zeilenabstand von **1,5 LN**.

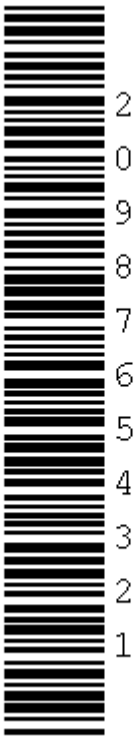


Definition des Stiles ZBARCROT

Im Fenster **HRT** wird dem Textknoten der Stil **ZBARCROT** zugewiesen. Im Textknoten wird die Klarschriftzeile (Variable **HRT1**) Zeichen für Zeichen (von hinten nach vorne) pro Zeile ausgegeben. Jede Zeile ist mit dem Absatz **R1** formatiert.



Merke: Das Ergebnis erscheint im Falle gedrehter Klarschriftzeilen in der SAP Druckvorschau nicht korrekt. Der Grund dafür ist die Tatsache, dass für die Vorschau nicht die angegebenen TrueType Schriften sondern deren Substitute verwendet werden. Sobald Sie jedoch das Formular Drucken oder z.B. ein PDF Dokument erzeugen, wird die Klarschriftzeile in gedrehter Form korrekt dargestellt.



Ergebnis der SAP-Druckvorschau

Reales Druckergebnis

Merke: Mit der oben beschriebenen Vorgehensweise lassen sich auch gedrehte Barcodes mit eingebetteter, oder halb eingebetteter Klarschriftzeile gestalten. Definieren Sie dafür die Parameter **MARGIN** und **OFFSET** in dem Programm **ZSF_BC_SETTINGS12** für den entsprechenden Barcode (das erzeugt einen weißen Einschnitt an der langen Barcodekante) und positionieren Sie die Klarschriftzeile entsprechend

Gedrehter Barcode mit halb eingebetteter Klarschriftzeile.

.

8 Implementierung einer Barcode Ausgabe mit AdobeForms

8.1 Nutzen von RBARC mit AdobeForms

SAP liefert im Standard Adobe Life Cycle Designer die gängigen Barcodes an. In welchen Situationen macht es dennoch Sinn, RBarc+ zu nutzen?

- **Der gewünschte Barcodetyp ist in AdobeForms nicht verfügbar.**

Adobe Forms bietet in seiner Objektbibliothek nicht alle Barcodes an. Barcodes, die in AdobeForms nicht verfügbar sind, können mit RBarc+ erzeugt werden.

- **AdobeForms druckt den Barcode nicht in der gewünschten Qualität.**

Insbesondere beim Drucken von Barcodes an dynamischen Positionen im Hauptfenster (z.B. bei der Ausgabe von Positionsdaten) zerfranst der von Adobe Forms erzeugte Barcode an der oberen und unteren Seite. RBARC kann in dem Fall Barcodes mit dynamischer Positionierung in wesentlich besserer Qualität drucken.

- **Mit AdobeForms kann man die Barcodes nicht exakt parametrisieren.**

Manchmal geben die Geschäftspartner exakte Parameter vor, mit denen ein Barcode gedruckt werden muss. Mit Adobe Forms kann zwar die Höhe genau vorgegeben werden und man kann den Barcode auch schmaler und breiter ziehen, aber eine exakte Vorgabe der Modulbreite ist nicht möglich. Mit RBARC können Sie die gesamte Palette an Parametern nutzen, um den Barcode exakt gemäß Definition zu erzeugen.

- **AdobeForms gibt die Prüfziffer nicht zurück.**

Adobe Forms erzeugt zwar den Barcode inklusive Prüfziffer, gibt die Prüfziffer jedoch nicht an das rufende Programm zurück. Mit RBarc+ können Sie den Barcode mit Prüfziffer drucken und die ermittelte Prüfziffer im rufenden Programm auswerten.

8.2 Funktionsweise des Erzeugen von Barcodes in AdobeForms mit RBarc+

Zunächst ruft das ABAP-Druckprogramm bzw. ein Stück ABAP-Coding in der Schnittstelle zum Zeitpunkt der Initialisierung die Schnittstelle zum RBarc+ auf und übergibt dabei den Wert, der im Barcode dargestellt werden soll und den Barcodetypen. Der Barcodetyp enthält dabei die Parameter, wie der Barcode ausgegeben werden soll, z.B. Barcodetyp (Code39, …), Barcodehöhe, Modulbreite, Berechnung der Prüfziffer usw.

RBarc+ erzeugt nun temporär den Barcode als Bitmap in der SAP-Grafikdatenbank (Transaktion SE78) und gibt ihn als binäres Objekt vom Typ XSTRING an das rufende ABAP-Programm zurück. Anschließend wird das temporäre Bitmap in der SAP-Grafikdatenbank wieder gelöscht.

8.3 Einbinden von Barcodes in ein AdobeForms-Formular

In ein Adobe Forms Formular können beliebig viele Barcodes eingebunden werden. Die Anpassungen betreffen die Schnittstelle und das Formular, so dass bestehende Druckprogramme nicht geändert werden müssen. Somit ist die RBarc+ Lösung für SAP-Updates geeignet.

Damit die Formularanpassungen so gering wie nur möglich ausfallen, werden in der Formularschnittstelle nur die nötigsten Definitionen vorgenommen. Dazu gehört insbesondere die Information: „**was soll kodiert werden**“? Alle anderen Barcode Eigenschaften, wie **Symbologie**, **Breite**, **Höhe** usw. werden in dem außen liegenden ABAP Programm **ZAF_BC_SETTINGS** vorgenommen.

Über die Eigenschaften der Barcodes, wie **Symbologie**, **Höhe**, **Breite** usw. schreiben wir später im **Kapitel 9 „Die Barcode Generierung“**, da dieser Teil identisch für **SAPscript**, **SmartForms** und **AdobeForms** Anwendungen ist. Im Folgenden präsentieren wir ein Listing einer

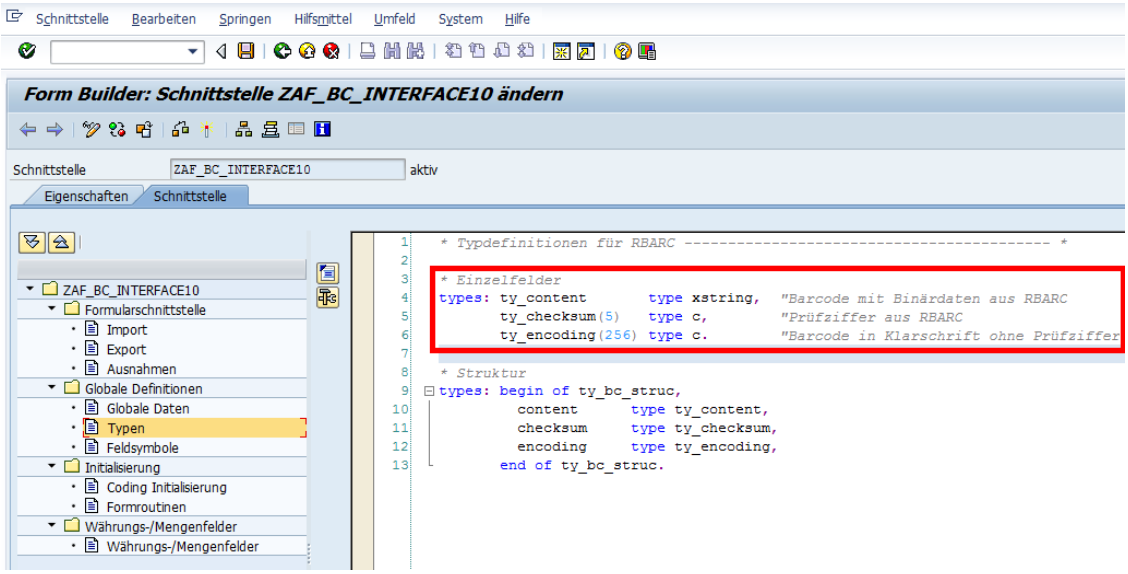
AdobeForms-Schnittstelle und eines **AdobeForms**-Formulars mit entsprechenden Erklärungen, die Sie in die Lage versetzen werden, selbst Barcodes mit RBarc+ in **AdobeForms** Formulare einzubinden.

Die Logik der Vorgehensweise ist immer die gleiche und richtet sich nach folgendem Prinzip:

- **Anlegen von globalen Datentypen in der Schnittstelle**

Es müssen zunächst folgende 3 Datentypen im Knoten „Globale Definition→ Typen“ angelegt werden:

```
types: ty_content        type xstring,   "Barcode mit Binärdaten aus RBARC
       ty_checksum(5)    type c,         "Prüfziffer aus RBARC
       ty_encoding(256)  type c.         "Barcode in Klarschrift ohne Prüfziffer
```

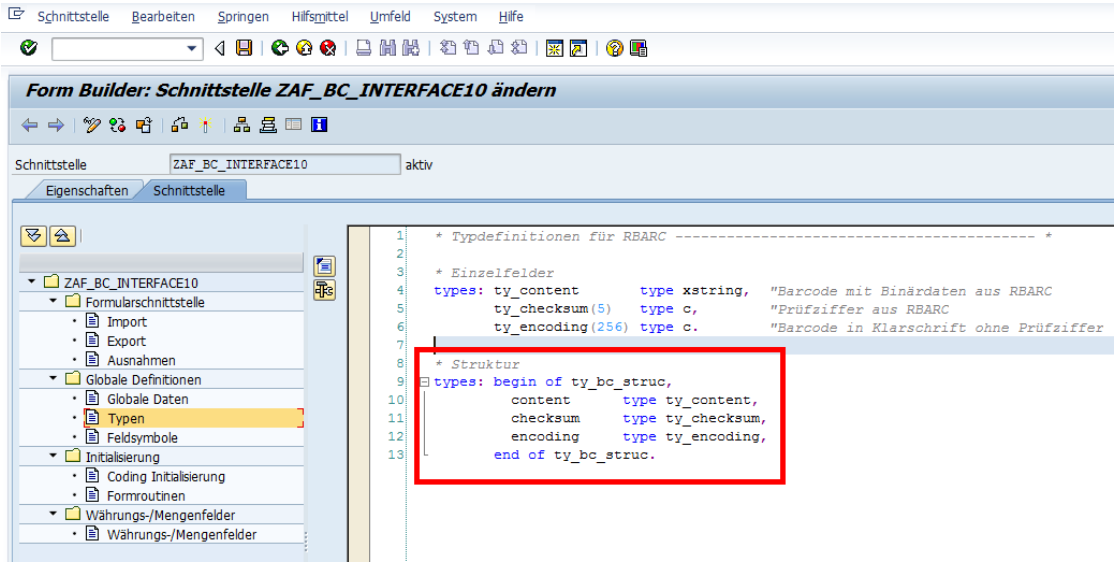


Anschließend definieren Sie noch eine globale Typdefinition für eine Struktur.

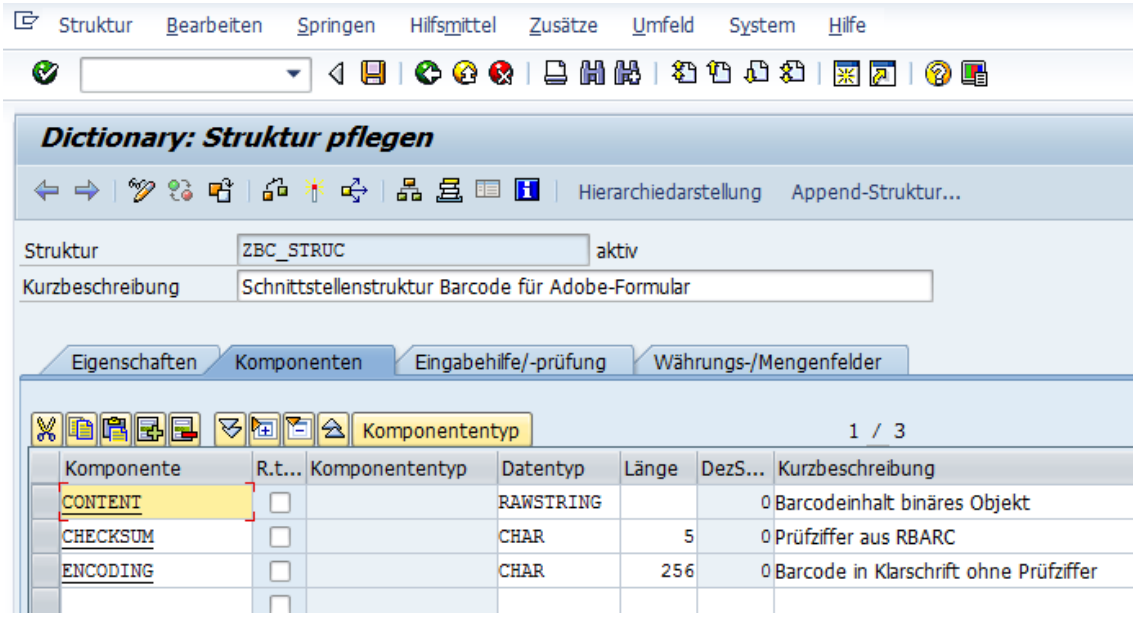
Die Struktur enthält folgende 3 Felder:

a) ein Feld, das den von RBarc+ erzeugten Barcode als Grafik mit Binärdaten enthält,
b) ein Feld, das den im Barcode zu codierenden Wert enthält,
c) ein Feld, das die in RBARC errechnete Prüfziffer enthält,

```
types:  begin of ty_bc_struc,
                content      type ty_content,
                checksum     type ty_checksum,
                encoding     type ty_encoding,
        end of ty_bc_struc.
```



Analog kann die Erweiterung der Struktur auch im Data Dictionary erfolgen, wenn in der Schnittstelle eine Struktur aus dem Data Dictionary verwendet wird, die im übergeordneten Druckprogramm gefüllt wird. In diesem Fall wird für das Feld, das den Barcode als Grafik enthält, der Typ RAWSTRING (entspricht dem elementaren ABAP-Datentypen XSTRING) verwendet.



Sie können die einzelnen Felder auch in bereits existierende Strukturen einer Schnittstelle einfügen.

- **Anlegen einer globalen Struktur in der Schnittstelle**

  Jetzt legen Sie eine globale Struktur in der Schnittstelle im Knoten „Globale Definition→ Globale Daten“ an:

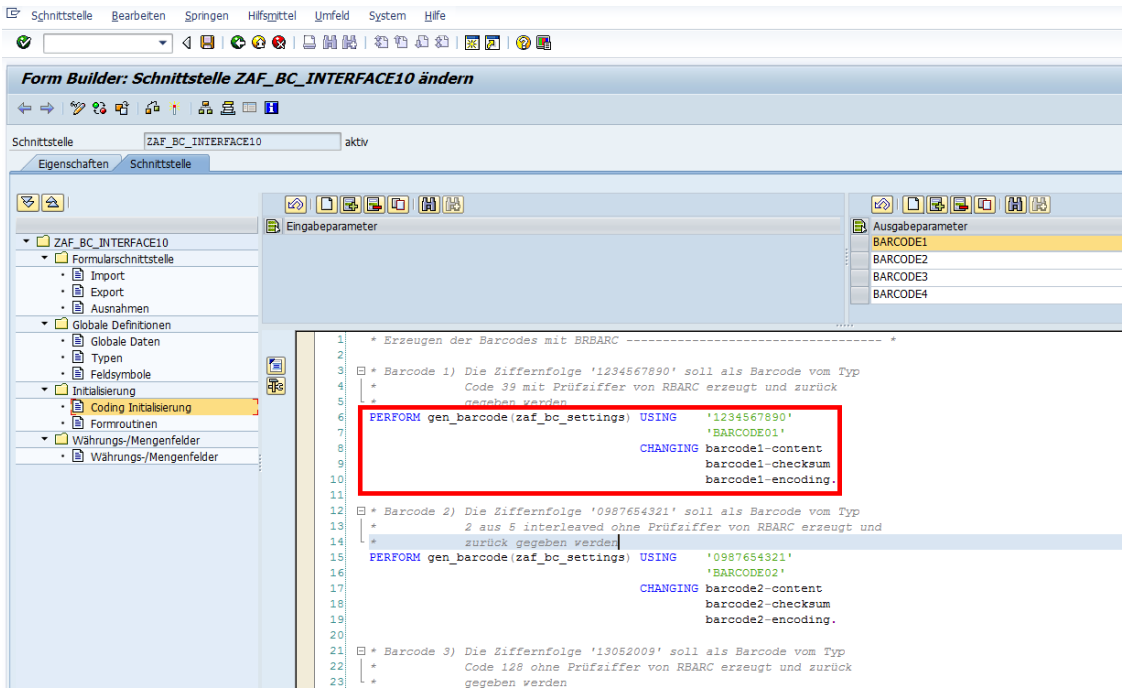
  Die globale Struktur ist vom Typ der soeben erstellten Strukturdefinition.

  Im Beispiel definieren wir eine globale Struktur BARCODE1 des Datentyps TY_BC_STRUC. Die globale Struktur enthält die 3 o.a. Variablen Barcodeinhalt, Prüfziffer und Barcode in Klarschrift.

- **Aufruf von RBarc+ in der Initialisierung der Schnittstelle**

  Im Knoten „Initialisierung → Coding Initialisierung“ der Schnittstelle wird jetzt die Schnittstelle zum RBarc+ aufgerufen, um den Barcode zu erzeugen.

```
PERFORM gen_barcode(zaf_bc_settings) USING     '1234567890'
                                               'BARCODE01'
                                     CHANGING barcode1-content
                                              barcode1-checksum
                                              barcode1-encoding.
```



Die Schnittstelle weist 2 USING-Parameter und 3 CHANGING-Parameter auf, die dabei folgende Bedeutungen haben:

USING-Parameter 1: der im Barcode zu verschlüsselnde Wert
Hier wird immer der im Barcode zu verschlüsselnde Wert eingetragen. Es kann sich dabei um eine Auftrags-, Material-, Chargennummer usw. handeln.
Im Beispiel wird ein fixer Wert übergeben.

USING-Parameter 2: Barcodetyp mit allen Eigenschaften und Parametern, der in RBarc+ erzeugt wird
USING-Parameter 2 enthält eine Routine, die im Programm ZAF_BC_SETTINGS (hier im Beispiel) aufgerufen wird und alle Parameter für den Barcode enthält, wie z.B. den Barcodetyp (Code39, …), die Barcodehöhe, die Modulbreite usw. Diese Routine (im Beispiel BARCODE01) muss im gerufenen Programm, das die Schnittstelle zum RBarc+ enthält (im Beispiel ist es das Programm ZAF_BC_SETTINGS), angelegt werden.

CHANGING-Param. 1: Barcodeinhalt = binäres Objekt, das den Barcode als Grafik enthält
Die Grafik enthält immer nur den Barcode, aber nicht den Wert des Barcodes in Klarschrift.

CHANGING-Param. 2: Prüfziffer, die in RBarc+ berechnet wird

CHANGING-Param. 3: der im Barcode verschlüsselte Wert in Klarschrift ohne Prüfziffer
Da der Barcodeinhalt immer nur die Grafik, aber niemals den Wert des Barcodes in Klarschrift enthält, ist es angebracht, den im Barcode verschlüsselten Wert in einem separaten Feld zu speichern.

- **Erweiterung des Programms ZAF_BC_SETTINGS in der SE38**

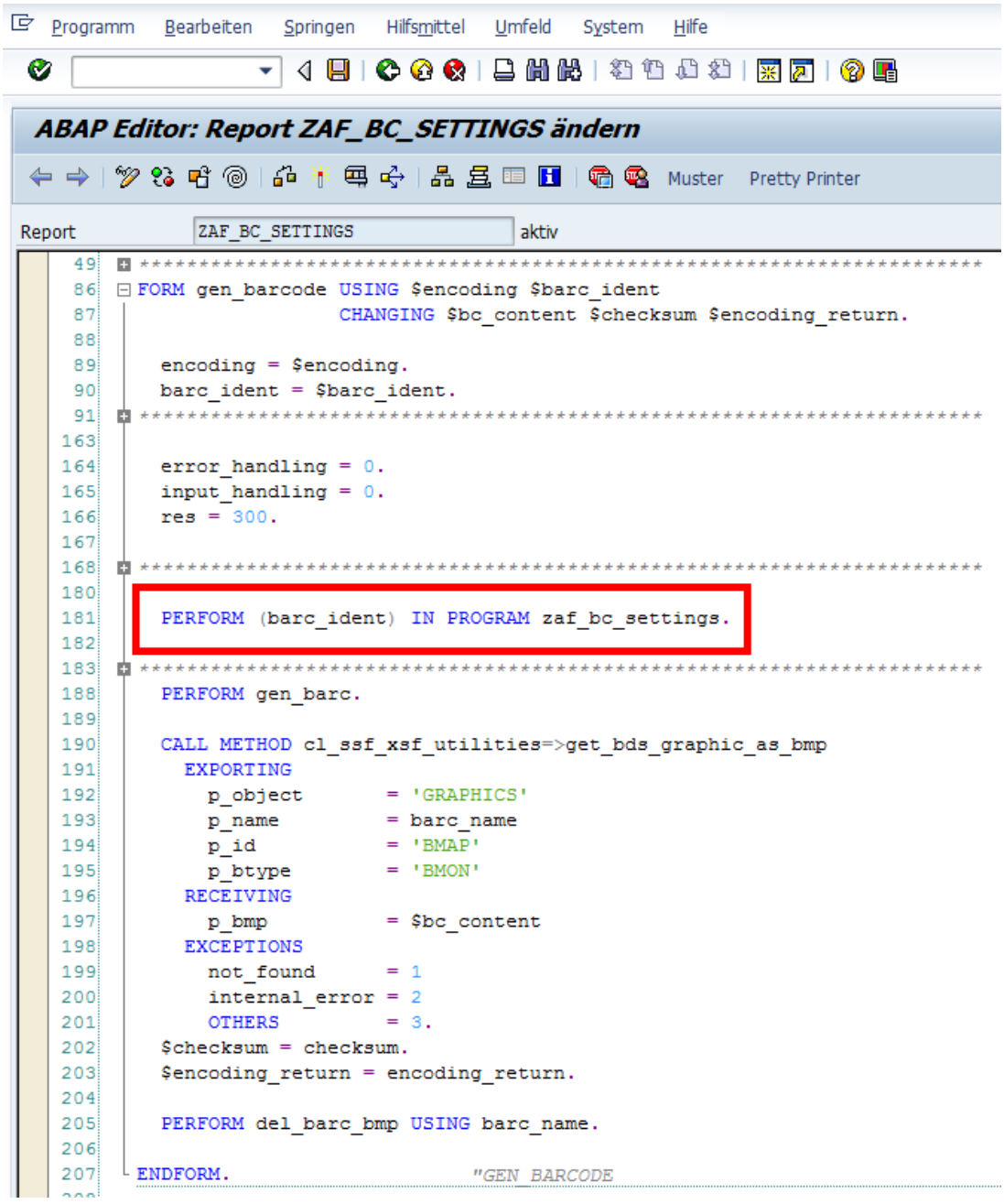
  Im Programm ZAF_BC_SETTINGS werden die Eigenschaften der verwendeten Barcodes (wie Barcodetyp, Barcodehöhe, Modulbreite usw.) definiert. Außerdem enthält es die zentrale Schnittstelle zum RBarc+, wo der Barcode letztendlich als Grafik erzeugt wird.

  Das Programm ZAF_BC_SETTINGS ist im Lieferumfang enthalten und sollte jetzt bereits installiert sein.

  Was nun im Programm ZAF_BC_SETTINGS getan werden muss, soll wiederum am Beispielformular ZAF_BC_FORM erläutert werden. Ziel ist es, eine Reihe von Ziffern im Formularkopf als Barcode mit folgenden Eigenschaften zu drucken:

Barcodetyp: Code 39 mit Prüfziffer
Barcodehöhe: 13mm

In der Adobe Forms Schnittstelle wird die Routine GEN_BARCODE des Programms ZAF_BC_SETTINGS aufgerufen.



Die Variable ENCODING enthält den im Barcode zu verschlüsselnden Wert, die Variable BARC_IDENT eine Formroutine, die aufgerufen werden soll, um mit den dort hinterlegten Parametern der Barcode zu erzeugen. Dem Feld BARC_IDENT ist im Programm ZAF_BC_SETTINGS ein Unterprogramm zugeordnet, das von Ihnen angelegt und mit Werten versorgt werden muss.

In unserem Beispiel enthält das Feld BARC_IDENT den Wert BARCODE01. Durch die ABAP-Anweisung

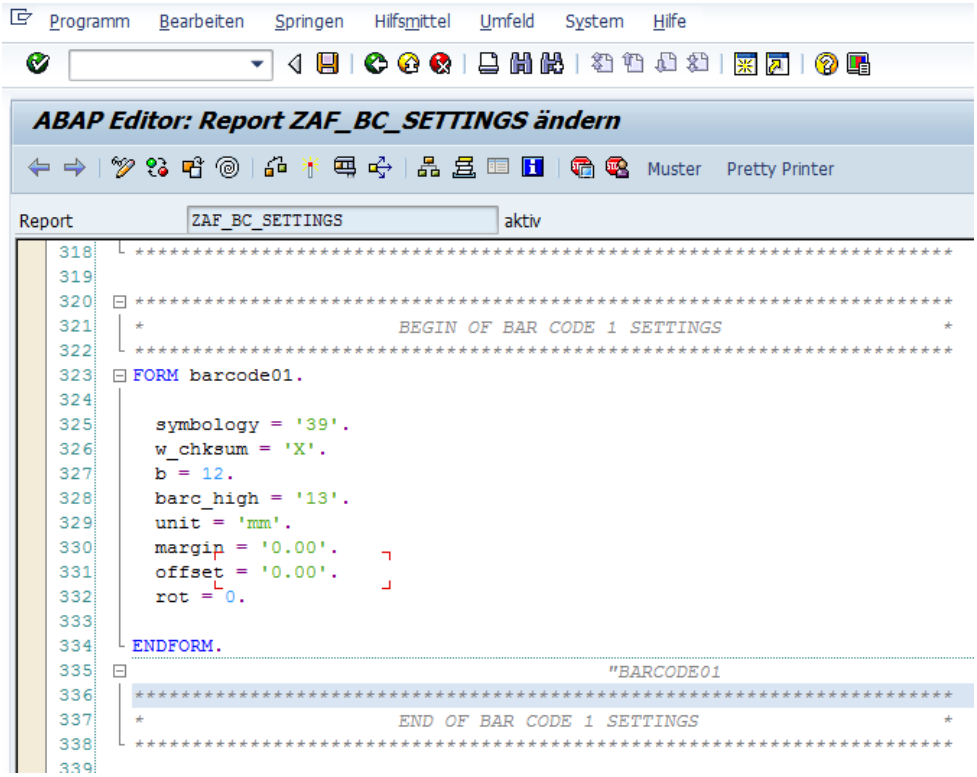
```
PERFORM (barc_ident) IN PROGRAM zaf_bc_settings.
```

wird die Routine BARCODE01 aufgerufen, die sich weiter unten im Programm ZAF_BC_SETTINGS befindet.

```
FORM barcode01.

  symbology = '39'.
  w_chksum = 'X'.
  b = 12.
  barc_high = '13'.
  unit = 'mm'.
  margin = '0.00'.
  offset = '0.00'.
  rot = 0.

ENDFORM.
```

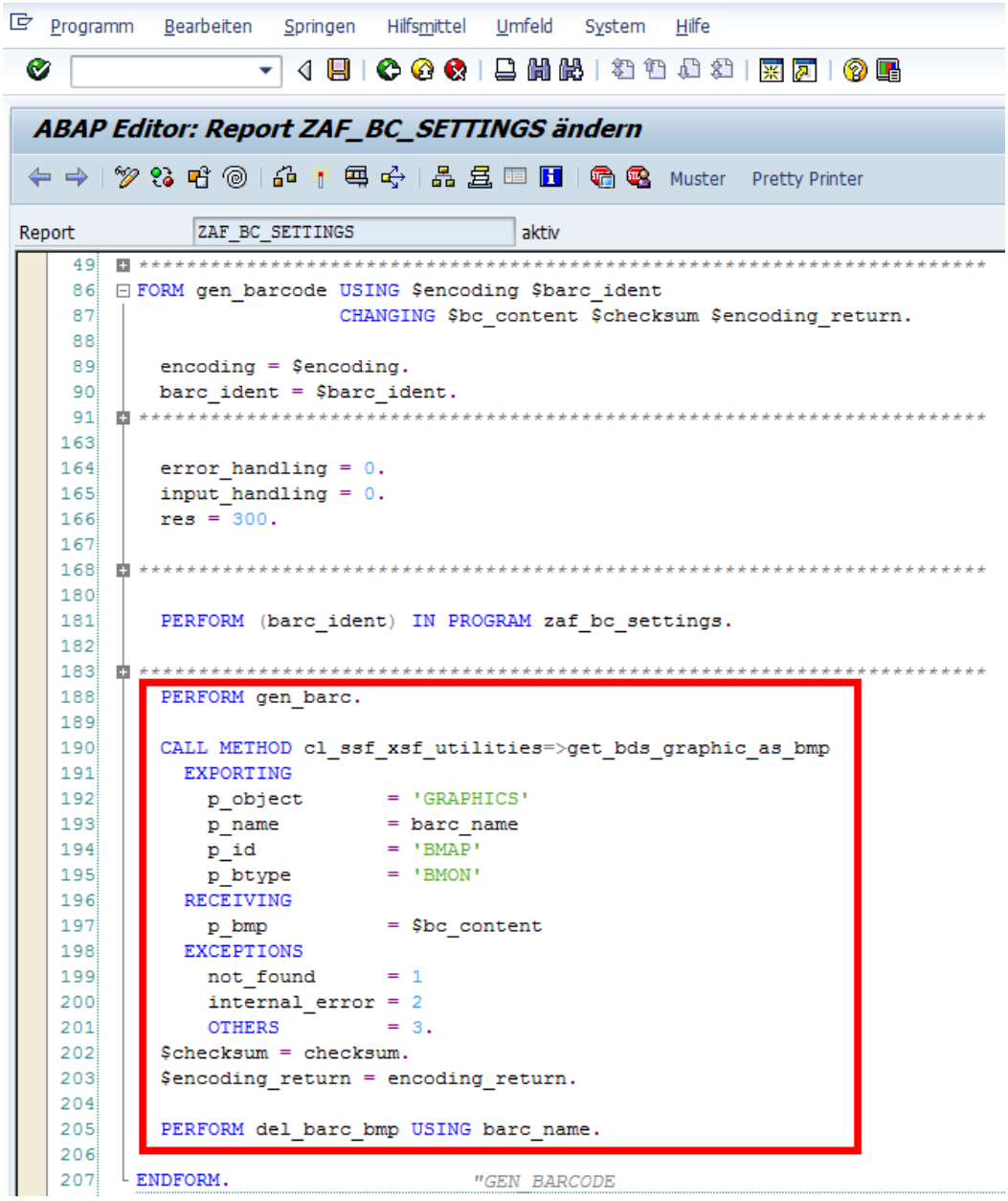


Folgende Parameter werden gesetzt:

symbology = ‚39‘ → Barcode vom Typ Code128-A wird erstellt
w_checksum = ‘X’ → Barcode wird mit Prüfziffer erstellt
barc_high = ‚13‘ → Barcodehöhe = 13mm

Alle weiteren Parameter, die gesetzt werden können, entnehmen Sie bitte der Dokumentation zu RBarc+.

Mit der Routine GEN_BARC wird der Barcode in RBARC erzeugt.



Die Methode

```
CALL METHOD cl_ssf_xsf_utilities=>get_bds_graphic_as_bmp
```

holt sich den erzeugten Barcode als binaries Objekt aus der Grafikdatenbank und füllt das Feld BC_CONTENT, das dann den Barcode an das rufende Programm übergibt.

In Routine

```
PERFORM del_barc_bmp USING barc_name.
```

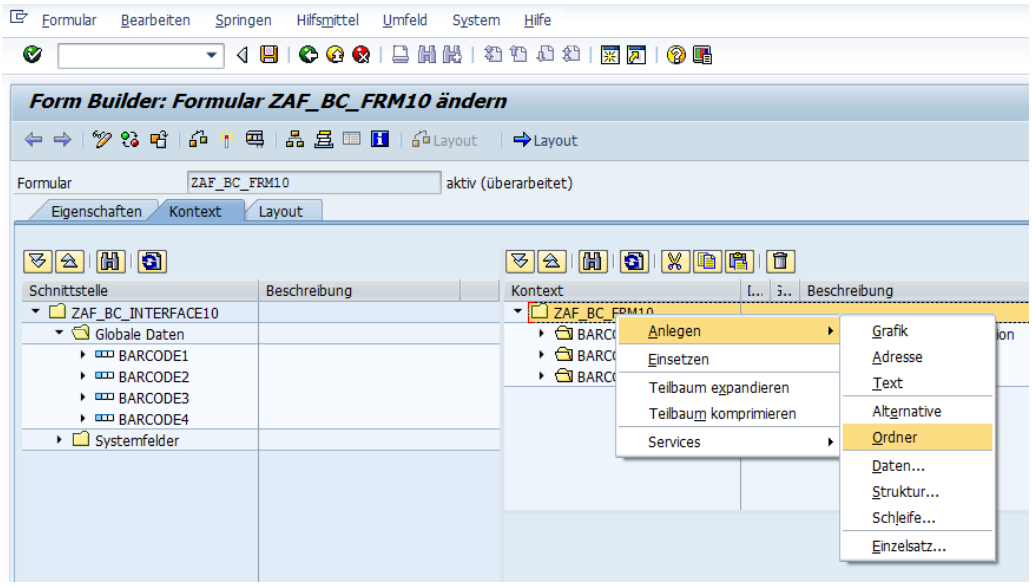
wird dann der erzeugte Barcode wieder gelöscht.

- **Erweiterung des Kontext im Formular**

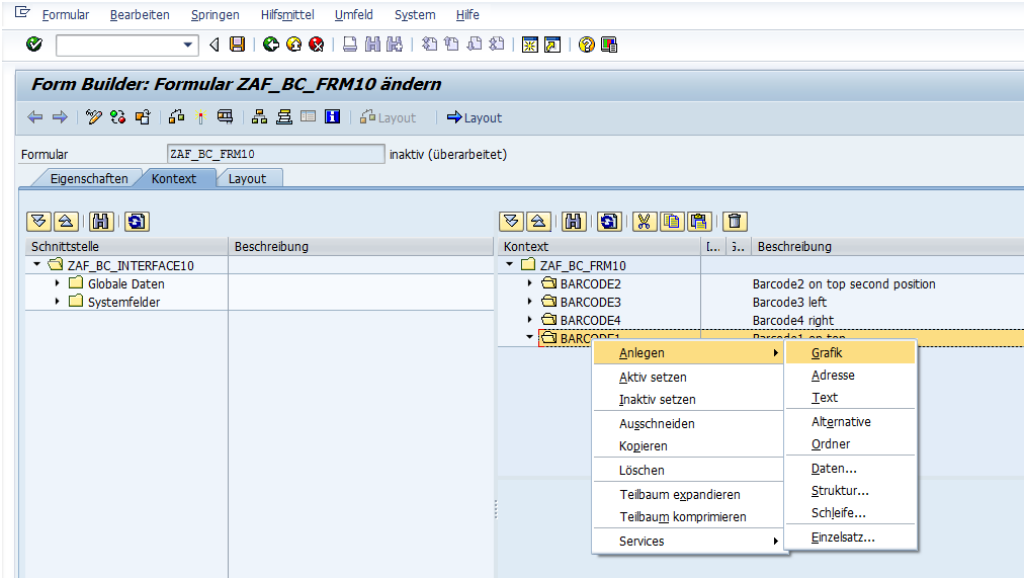
  Im Adobe Form Builder werden die mit RBarc+ erzeugten Barcodes als Bildobjekt dargestellt.

  Zunächst muss daher der Kontext um die notwendigen Elemente für einen Barcode erweitert werden.

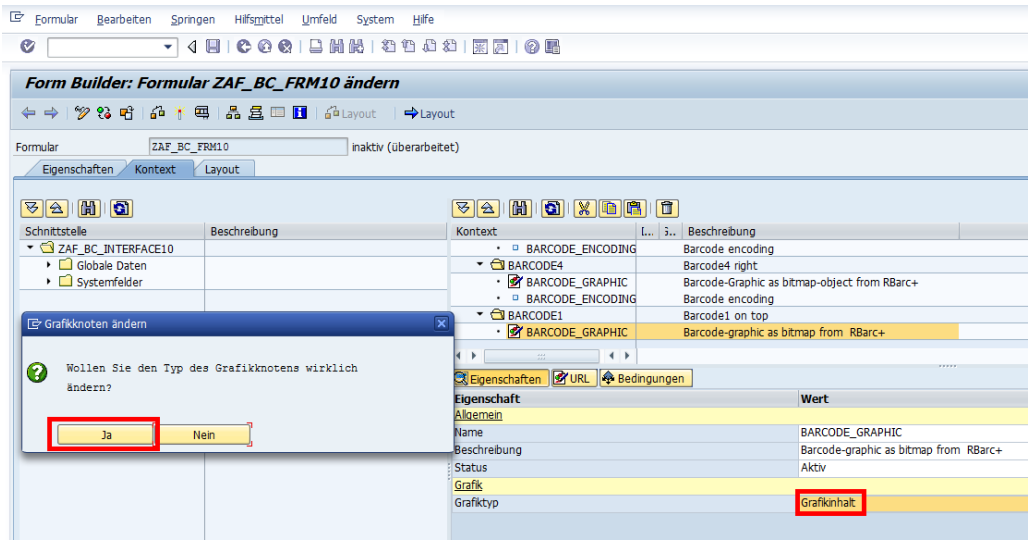
  Gehen Sie nun im Formular (Transaktion SFP) in den Kontext und legen Sie einen Ordner an. Geben Sie dem Order einen Namen und eine Beschreibung. Im Beispiel lautet der Name des Ordners BARCODE1.



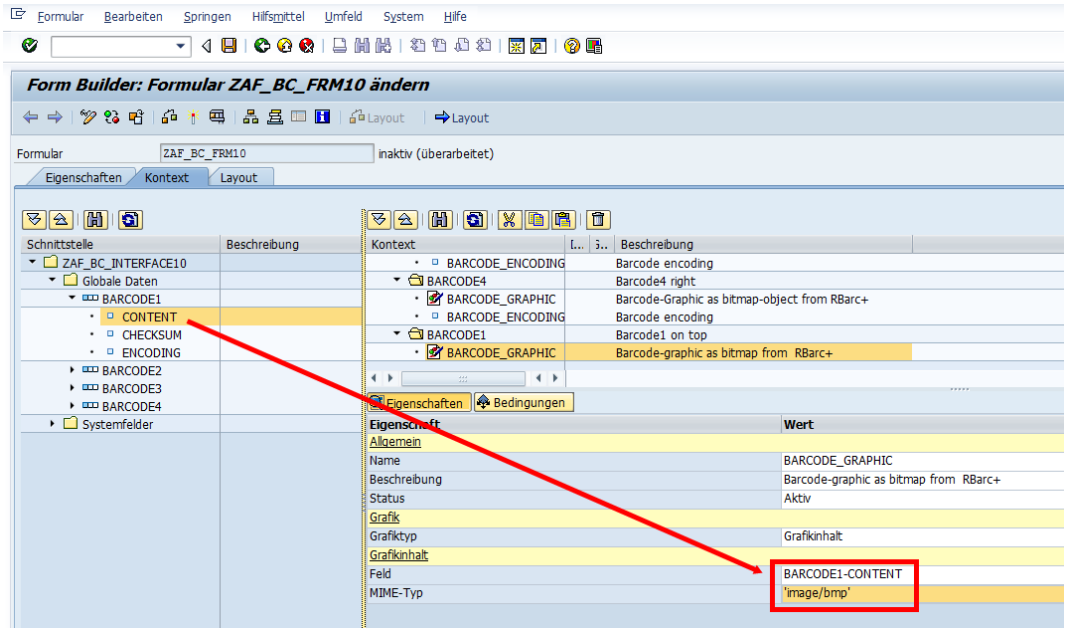
Legen Sie nun im Order BARCODE1 eine Grafik an.
Geben Sie der Grafik einen Namen und eine Beschreibung. Im Beispiel lautet der Name der Grafik BARCOD_GRAPHIC.



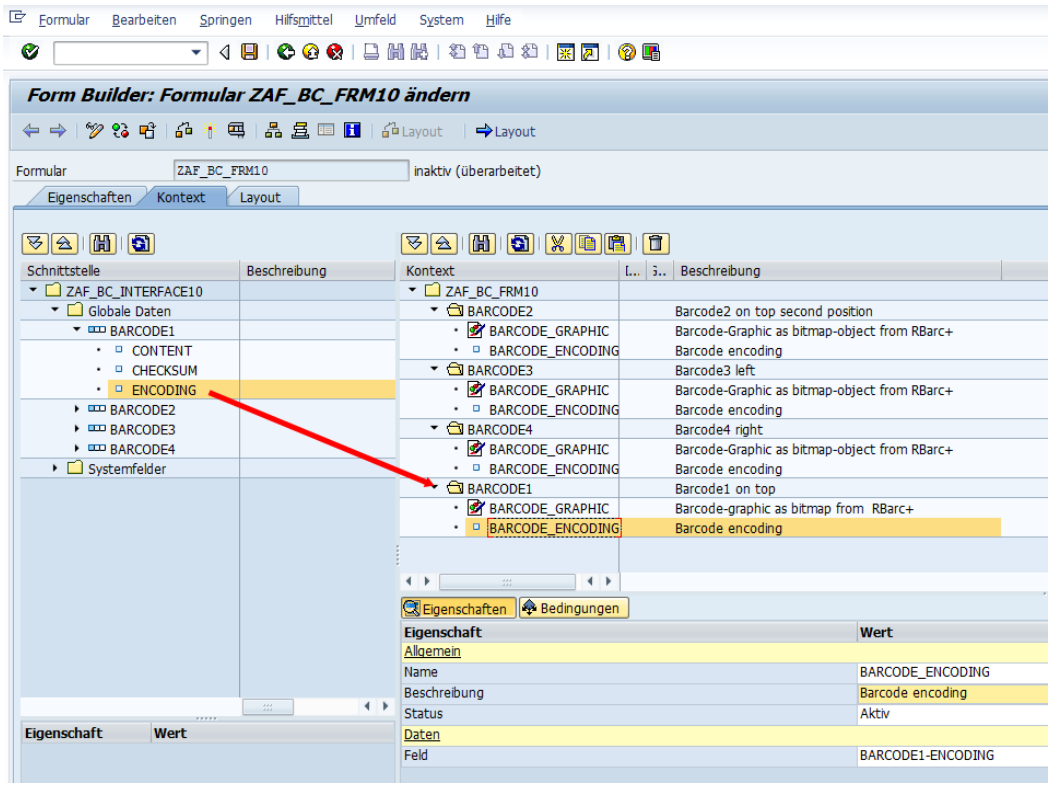
Wechseln Sie jetzt den Grafiktyp auf „Grafikinhalt“ und bestätigen Sie das Popup mit „Ja“.



Orden Sie nun der Grafik das Feld zu, in dem der Barcodeinhalt als Binärdatenstrom enthalten ist (im Beispiel ist es BARCODE1-CONTENT) und geben Sie als MIME-Type ‚image/bmp' mit.

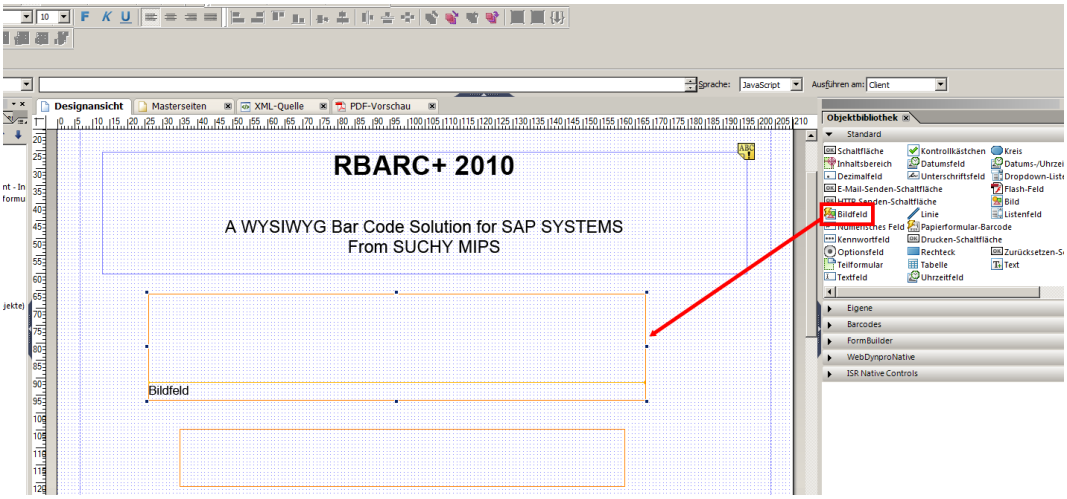


Legen Sie jetzt noch im Ordner BARCODE1 ein Datenfeld an, indem Sie mit gedrückter linker Maustaste aus der Schnittstelle das Feld BARCODE1-ENCODING auf den Order BARCODE1 ziehen. Dieses Feld enthält den Barcode in Klarschrift.

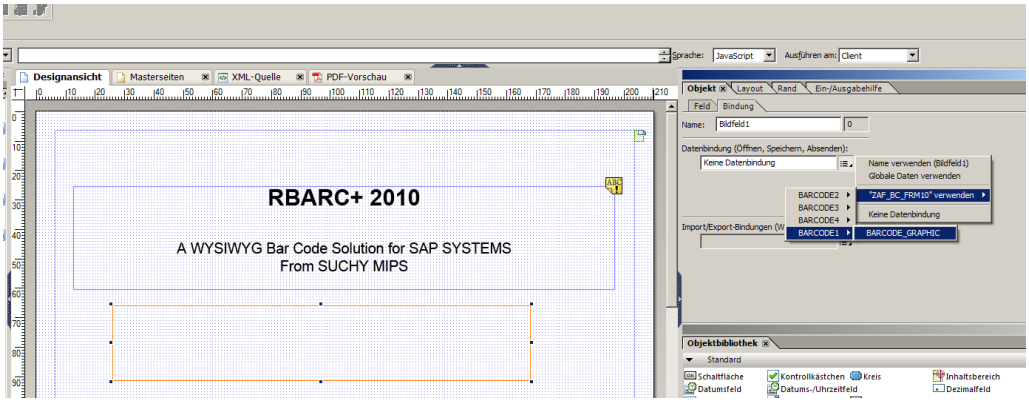


- **Erweiterung des Layouts im Formular**
  Wechseln Sie nun in die Layoutansicht des Form Builders.

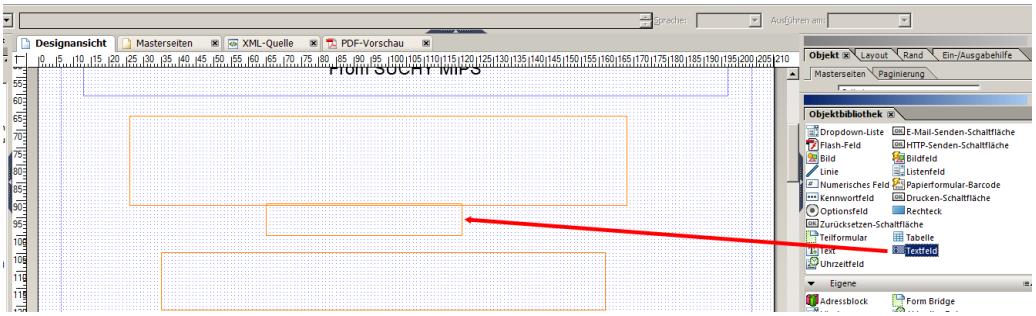
  Legen Sie ein Element vom Typ „Bildfeld" an.



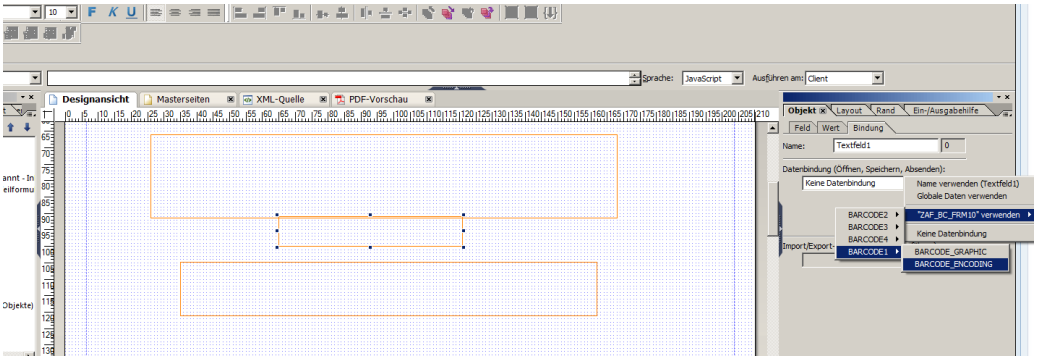
Geben Sie dem Bildfeld als Datenbindung den im Kontext definierten Grafikknoten mit (im Beispiel BARCODE1.BARCODE_GRAPHIC).



Legen Sie für den Wert des Barcodes in Klarschrift noch unterhalb des soeben angelegten Bildfeldes ein Feld vom Typ „Textfeld“ an.



Geben Sie dem Textfeld als Datenbindung das Feld der Struktur mit, das den Wert des Barcodes in Klarschrift enthält (im Beispiel BARCODE1.BARCODE_ENCODING).



Aktivieren Sie jetzt das Formular. Nun können Sie das Formular testen.

9 Die Barcode Generierung

Bei jeder Implementierung einer Barcode Ausgabe, sei es mit SAPscript, SmartForms oder mit AdobeForms, wird die Form Routine **GEN_BARCODE** aufgerufen. Diese Form Routine befindet sich im Programm **ZSS_BC_SETTINGS12** für SAPscript und im Programm **ZSF_BC_SETTINGS12** für SmartForms. Beide Programme sind identisch, bis auf die Schnittstellengestaltung zu den Formularen, die bei SAPscript anders als bei SmartForms vorgeschrieben ist. Aus diesem Grund werden wir und auf das **SETTINGS** Programm beziehen, wobei wir damit das jeweils entsprechende Programm meinen.

Das **SETTINGS** Programm ist dafür zuständig, den vom Formular angeforderten Barcode zu generieren, und dessen Namen an das Formular zurückzugeben.

Der allgemeine Programmablauf ist immer gleich und läuft nach folgendem Schema ab:

Schritt 1

Nach dem Start der Form Routine **GEN_BARCODE** aus dem Formular heraus, werden globale Einstellungen getätigt, die das generelle Verhalten von RBarc+ bestimmen, z.B. beim Auftreten von Fehlern.

Schritt 2

Im zweiten Schritt wird die aus dem Formular übergebene Variable **BARC_IDENT** ausgewertet und zu der Form Routine verzweigt, die den gleichen Namen trägt. Wenn Sie also neue Barcodes implementieren, sorgen Sie unbedingt dafür, dass der im Formular angegebene Wert für **BARC_IDENT** (z.B. **BARCODE06**) identisch ist mit dem Namen der Form Routine in der die grundlegenden Barcode Eigenschaften festgelegt werden.

Schritt 3

Im dritten Schritt wird der Barcode von den RBarc+ Routinen generiert und ins System gestellt. Zurückgegeben wird der Barcodename, die Prüfziffer (falls der Barcode mit Prüfziffer erzeugt wurde) und der tatsächlich verschlüsselte Wert.

Schritt 4

Zum Schluss werden die Rückgabe Parameter **BARC_NAME** (Name des Barcodes), **CHECKSUM** (Prüfziffer) und **ENCODING_RETURN** (der tatsächlich verschlüsselte Wert) an das aufrufende Formular zurückgegeben.

9.1 Definition globaler Parameter für die Ablaufsteuerung

Zu den globalen Parametern (standardmäßig gültig für alle zu erzeugenden Barcodes) gehören die 4 folgenden:

Graph_type Legt den Typ der Grafik fest. Standard für SAPscript ist **<0OTF>** und für SmartForms **<0BMP>**. Sollte in der Regel nicht geändert werden.

Error_handling Legt fest, wie Fehler von RBarc+ behandelt werden sollen.

Die zulässigen Werte sind:

0 bei einem Fehler hält das Programm an und zeigt ein Nachrichtenfenster mit der entsprechenden Fehlermeldung. Nach Quittierung der Meldung durch den Benutzer korrigiert das Programm den Fehler, falls er unkritisch ist (dabei werden für fehlerhafte Parameter, wie z.B. Höhe, Standardwerte gesetzt), sodass der Barcode erzeugt werden und das Programm fortfahren kann. Trat ein kritischer Fehler auf, wird eine Fehlermeldung generiert und das Programm fährt fort, aber es wird kein Barcode erzeugt, sondern ein schwarzes Viereck.

1 Falls ein nicht kritischer Fehler vorkommt, wird er automatisch korrigiert und es wird keine Fehlermeldung angezeigt. Bei kritischen Fehlern erscheint eine Fehlermeldung in einem Nachrichtenfenster. Nach Quittierung der Nachricht durch den Anwender fährt das Programm fort, aber es wird kein Barcode erzeugt, sondern ein schwarzes Viereck.

2 Es werden keine Fehlermeldungen angezeigt. Bei unkritischen Fehlern werden diese automatisch korrigiert. Bei kritischen Fehlern wird statt des Barcodes ein schwarzes Viereck erzeugt**.**

Alle anderen Werte werden wie '0' behandelt.

Die Liste aller kritischen und unkritischen Fehler finden Sie auf der nächsten Seite.

Kritische Fehler

Fehler, die nicht korrigierbar sind, z.B.

- Fehlender oder ungültiger Verschlüsselungswert
- Fehlende oder ungültige Symbologie Bezeichnung.

Nicht kritische Fehler

Fehler, die durch nicht oder falsch definierte Parameter verursacht wurden (z.B. Barcodehöhe = 0). Bei der automatischen Korrektur dieser Fehler werden den entsprechenden Parametern Standardwerte laut folgender Liste zugewiesen

- RES, Standardwert = 150.
- XPOS, Standardwert= 0.
- YPOS, Standardwert = 0.
- HIGH, Standardwert = 10 mm
- MARGIN, Standardwert = 0.
- OFFSET, Standardwert = 0.
- ROT, Standardwert = 0.
- UNIT, Standardwert = 'mm'
- SIGN, Standardwert = 0.
- B, Standardwert = 10,
- BB, Standardwert = 2xB,
- BBB, Standardwert = 3xB
- BBBB, Standardwert = 4xB
- W, Standardwert = B
- WW, Standardwert = 2xB
- WWW, Standardwert = 3xB
- WWWW, Standardwert = 4xB

Input_handling: Legt fest, wie der übergebene Verschlüsselungswert behandelt werden soll.

Die zulässigen Werte sind:

0 Der übergebene Verschlüsselungswert wird nicht verändert.

1 Führende Nullen werden gelöscht

2 Führende Leerzeichen werden gelöscht.

3 Führende Leerzeichen und führende Nullen werden gelöscht.

+10 Addieren sie die Zahl 10 zu Input_handling um ungültige Zeichen aus dem Verschlüsselungswert automatisch herauszufiltern. Beispiel: Code 39, Verschlüsselungswert '123ä456'. 'ä' ist ein ungültiges Zeichen und würde normalerweise einen Fehler verursachen. Bei Angabe von Input_handling 10, 11, 12 oder 13 wird 'ä' herausgefiltert und nur '123456' verschlüsselt.

Res Auflösung der Barcodegrafik. Der Standardwert ist 150dpi. Bitte beachten Sie, dass die im SETTINGS Programm definierte Auflösung mit der Auflösung, die im Formular beim Grafiktyp **BMP** angegeben wurde, übereinstimmen muss (gilt nicht für SAPscript mit **OTF** Format). Falls der Barcode gedruckt werden soll, achten Sie darauf, dass die gewählte Auflösung von Ihrem Druckgerät unterstützt wird. Die meisten Laserdrucker unterstützen Auflösungen von 75, 150, 300 und 600dpi. Manche Drucker jedoch, wie z.B. eine Reihe von ZEBRA Druckern, unterstützen nur 202dpi.

Merke: Je höher die Auflösung, desto mehr Daten werden produziert und desto mehr Rechenzeit pro Barcode wird benötigt. Obwohl die Barcode Generierung in der Regel sehr schnell ist und im Millisekunden Bereich liegt, kann sich das beim Drucken von Etiketten mit vielen Barcodes pro Seite bemerkbar machen.

Je geringer die Auflösung desto sprunghafter steigt die Barcode Breite des gesamten Barcodes bei Änderung der Modulbreite (siehe auch weiter unten in Kapitel 8).

Merke: Auf Laserdruckern darf die Grafikauflösung von der sonstigen Druckauflösung abweichen.

Die Standardwerte für alle 4 Parameter wurden so gewählt, dass sie von Ihnen nur in ganz seltenen Fällen geändert werden müssen. In der Regel können Sie diese Werte unverändert übernehmen.

Merke: Manchmal ist es notwendig, einen dieser „globalen“ Parameter einzeln, nur für einen Barcode zu ändern, wie z.B. die Auflösung. Definieren Sie in diesem Fall die Auflösung erneut in der Form Routine, in der die einzelnen Barcode Eigenschaften festgelegt wurden. Diese Einstellung wird dann für diesen Barcode gelten, da die Abarbeitung der Barcode Variablen nach der Abarbeitung der globalen Variablen geschieht. Löschen Sie jedoch nicht die globale Einstellung für einen solchen Parameter, sonst müssen sie später immer den Wert für jeden Barcode einzeln eingeben.

9.2 Definition der einzelnen Barcode Eigenschaften.

Die einzelnen Barcode Eigenschaften werden in einer Form Routine festgehalten, deren Name mit dem Wert des Parameters **BARC_IDENT** aus dem Formular übereinstimmen muss. Standardmäßig sind 5 solche Form Routinen in dem ABAP Programm vordefiniert: **BARCODE01, BARCODE02, BARCODE03, BARCODE04** und **BARCODE05**. Sie können aber jederzeit weitere Barcode Definitionen zu dem SETTINGS Programm hinzufügen. Wir erklären die einzelnen Eigenschaften anhand der Parameter, die in der Form Routine **BARCODE03** definiert wurden:

```
FORM barcode03.
  symbology = '39'.
  w_chksum = ''.
  b = 12.
  barc_high = '13'.
  unit = 'mm'.
  margin = '0.00'.
  offset = '0.00'.
  rot = 0.
ENDFORM.
```

SYMBOLOGY Legt fest, mit welcher Barcode Symbologie kodiert wird.

Folgende Werte sind zulässig:

'2o5interl'	2 of 5 interleaved
'2o5indust'	2 of 5 industrial
'2o5matrix'	2 of 5 matrix
'2o5gp'	2 of 5 German postal bar code (11 and 13)
'39'	Code 39
'39e'	Code 39 extended
'39f'	French postal 39 A/R
'39d'	Danish 39 PTT
'93'	Code 93
'93e'	Code 93 extended
'codabarXY'	Codabar mit dem entsprechenden START und STOP Symbol. Es gibt 4 START und STOP Symbole: „A“, „B“, „C“ und „D“. Sie können in beliebiger Reihenfolge eingesetzt werden. Wählen Sie z.B. **'codabarAC' um „A“** als **START-** und **„C“** als **STOP** Zeichen zu verwenden.
'128-A'	Code 128 A
'128-B'	Code 128 B
'128-C'	Code 128 C
‚128FREE‘	Frei editierbarer Code 128
'E.A.N.128'	Code EAN128
'128auto'	Code 128 Autoswitch

'ucc-128'	Code UCC-128
'128HIBC'	HIBC Barcode, 1-Zeilig, basierend auf Code 128 für **Supplier** und **Provider Label. Supplier** und Provider Daten müssen durch ein „/“ voneinander getrennt sein. Das Zeichen „+“ am Anfang kann, muss aber nicht gesetzt werden (in diesem Fall wird es von RBarc+ automatisch hinzugefügt). Die HIBC Prüfziffer wird automatisch erzeugt, darf also nicht mit gesendet werden.
'128HIBCP'	HIBC Barcode für Provider Label. Das Zeichen „+“ am Anfang wird nicht automatisch hinzugefügt. Die HIBC Prüfziffer wird automatisch berechnet.
'MSI1'	MSI (ohne Prüfziffer)
'MSI2'	MSI 10
'MSI3'	MSI10+CHK10
'MSI4'	MSI+CHK11+CHK10
'EAN-13'	EAN/JAN 13
'EAN-13+2'	EAN/JAN 13+2 (mit 2-zahligem add-on)
'EAN-13+5'	EAN/JAN 13+5 (mit 5-zahligem add-on)
'EAN-8'	EAN/JAN 8
'EAN-8+2'	EAN/JAN 8+2 (mit 2-zahligem add-on)
'EAN-8+5'	EAN/JAN 8+5 (mit 5-zahligem add-on)
'UPC-A'	UPC-A
'UPC-A+2'	UPC-A+2 (mit 2-zahligem add-on)
'UPC-A+5'	UPC-A+5 (mit 5-zahligem add-on)
'UPC-E'	UPC-E
'UPC-E+2'	UPC-E+2 (mit 2-zahligem add-on)
'UPC-E+5'	UPC-E+5 (mit 5-zahligem add-on)

.

W_CHKSUM Legt fest, ob der Barcode mit oder ohne Prüfziffer erzeugt werden soll. Dieser Parameter wird nur bei Barcode Symbologieen ausgewertet, die eine optionale Prüfziffer zulassen. Falls eine Barcode Symbologie zwingend eine Prüfziffer vorschreibt (z.B. Code 128), dann wird dieser Parameter ignoriert.

Folgende Barcodes lassen eine optionale Prüfziffer zu:

Code 39

Code 39 extended

Code 2 of 5 interleaved

Code 2 of 5 matrix

B

Die sog. Modulbreite. Das ist die Breite eines schmalen Striches. Die Breiten anderer Striche und aller Lücken werden automatisch berechnet. Falls es notwendig sein sollte, können jedoch die weiteren Striche und Lücken von Hand definiert werden. Dazu stehen die Parameter „**BB**“, „**BBB**“, „**BBBB**“, „**W**“, „**WW**“, „**WWW**“ und „**WWWW**“ zur Verfügung. Je nach verwendeter Barcode Symbologie gibt es 2 bzw. 4 verschiedene Breiten (siehe Tabelle in Kapitel 8.4). Sie stehen normalerweise im Verhältnis 1:2 bzw. 1:2:3:4.
Die Maßeinheit ist 1/720 Zoll. Lesen Sie mehr über die Barcode Breite in Kapitel 8.3

Unit

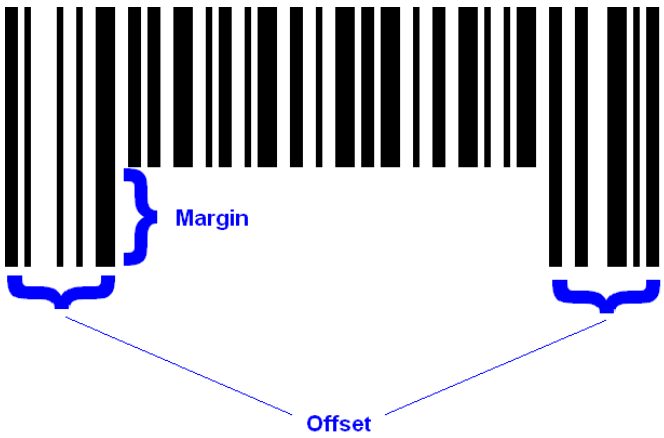
Maßeinheit für die Barcodehöhe (Parameter **BARC_HIGH**), die Tiefe des Einschnittes (Parameter MARGIN) und die Breite des Randes neben dem Einschnitt (Parameter OFFSET). Zulässige Werte sind „MM“ (Millimeter), „CM“ (Centimeter) und „INCH“ (Zoll). Die Werte können sowohl groß als auch klein geschrieben werden.

Margin

Tiefe des Einschnittes, falls eine Klarschriftzeile ganz oder teilweise in den Barcode eingebettet werden soll (siehe Bild weiter unten). Die Maßeinheit dafür wird mit dem Parameter UNIT festgelegt.

Offset

Mit diesem Parameter wird festgelegt, wie breit der Rand links und rechts neben dem Einschnitt sein soll (siehe Bild). Er bestimmt somit indirekt die Breite des Einschnittes, der mit dem Parameter MARGIN definiert wurde. Da der linke und rechte Rand gleich sind, wird der Einschnitt immer symmetrisch in den Barcode eingebettet sein.



Rot

Legt fest, ob und wie der Barcode gedreht werden soll. Zulässige Werte sind: 0, 90, 180 und 270 (Grad). Die Drehung wird nach rechts durchgeführt. Der Wert ist absolut, d.h. es wird immer von 0 Grad nach rechts gedreht.

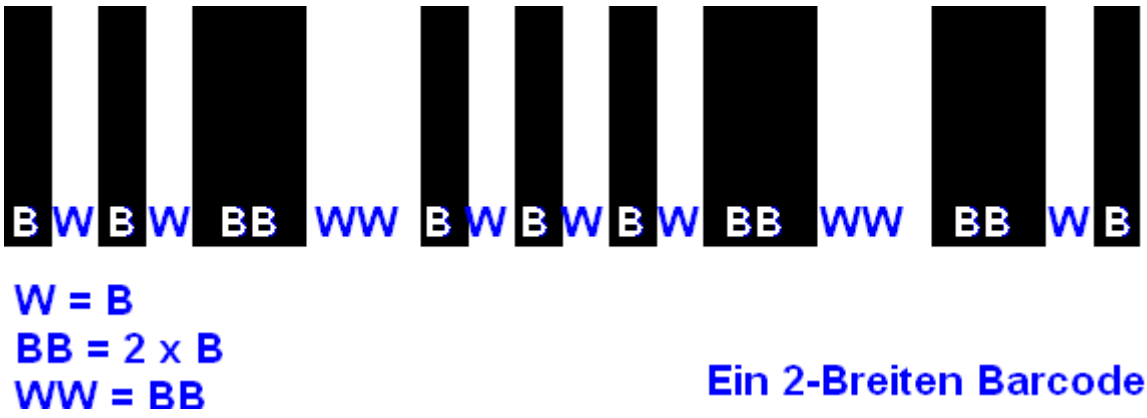
10 Mehr über Barcodes

10.1 Lineare Barcode Typen

Lineare Barcodes (auch 1D- oder eindimensionale Barcodes genannt) bestehen aus Strichen und Lücken, die in einem durch die verbindliche Spezifikation festgelegten Verhältnis zueinander stehen. Generell werden 1D Barcodes in sog. **2-Breiten** und **4-Breiten** Barcodes unterteilt.

2-Breiten Barcodes sind Barcodes, die aus Strichen und Lücken bestehen, die 2 verschiedene Breiten haben. Man spricht dabei von einem schmalen und einem breiten Modul – unabhängig davon ob es sich um einen Strich oder eine Lücke handelt (d.h. dass ein schmaler Strich genauso breit ist, wie eine schmale Lücke und ein breiter Strich genauso breit wie eine breite Lücke). In diesem Fall muss das Breitenverhältnis zwischen schmal und breit zwischen 1:2 und 1:3 liegen. Ist das Breitenverhältnis anders, führt das in der Regel zu einem nicht lesbaren Barcode. Folgende, von RBarc+ unterstützte Barcodes sind **2-Breiten** Barcodes:

2 of 5 interleaved
2 of 5 matrix
2 of 5 industrial
2 of 5 German Postal Barcode
Code 39
Code 39 extended
Codabar
MSI



Beispiel eines 2-Breiten Barcode Symbols

4-Breiten Barcodes weisen Striche und Lücken mit 4 verschiedenen Breiten auf. Auch hier müssen Striche und Lücken vom gleichen Typ gleich breit sein. Das Breitenverhältnis muss dabei wie 1:2:3:4 sein. Folgende von RBarc+ unterstütze Barcodes sind **4-Breiten** Barcodes:

Code 93
Code 93 extended
Code 128-A
Code 128-B
Code 128-C
Code 128-Auto
Code EAN-128
Code UCC-128
Code 128 HIBC
Code 128 HIBCP
Code EAN-13
Code EAN-13+2
Code EAN-13+5
Code EAN-8
Code EAN-8+2
Code EAN-8+5
Code UPC-A
Code UPC-A+2
Code UPC-A+5
Code UPC-E
Code UPC-E+2
Code UPC-E+5

Beispiel eines 4-Breiten Barcode Symbols

Merke: Sie müssen in der Regel nur die Breite des schmalsten Striches angeben (Parameter „**B**“). Alle anderen Striche und Lücken werden von RBarc+ automatisch berechnet. Dabei wird bei **2-Breiten** Barcodes das Breitenverhältnis **1:2** und bei **4-Breiten** Barcodes **1:2:3:4** angenommen und jede Lücke als gleich breit mit dem Strich gleichen Typs berechnet. Falls Sie für einen 2-Breiten Barcode ein anderes Breitenverhältnis wünschen, müssen Sie den Parameter „BB“ selbst definieren. Beachten Sie jedoch dabei, dass Sie das vorgegebene Breitenverhältnis zwischen 1:2 und 1:3 einhalten, sonst wird der Barcode unlesbar.

10.2 Definition der Modulbreite

Die Modulbreite (die Breite des schmalsten Striches) wird mit dem Parameter „**B**“ in dem Programm **ZSS_BC_SETTINGS12** (für SAPscript) bzw. im Programm **ZSF_BC_SETTINGS12** (für SmartForms) angegeben. Die Maßeinheit ist 1/720 Zoll. Diese Maßeinheit ist auflösungsunabhängig, somit müssen Sie diese Angaben nicht jedes Mal ändern, wenn Sie sich entscheiden, die Auflösung der Barcodegrafik zu ändern.

10.2.1 Modulbreite versus Auflösung

Da die Modulbreite, wie unter Kapitel 8.2 erwähnt, in der auflösungsunabhängigen Einheit von 1/720 Zoll angegeben wird, wird sie zur Laufzeit in auflösungsabhängige Einheiten (Pixel) umgerechnet. Das Ergebnis ist demzufolge immer eine Ganzzahl. Z.B. ergibt die Zahl 12 bei einer Auflösung von 150dpi 3 Pixel für ein Modul und bei einer Auflösung von 300dpi 5 Pixel für ein Modul. Die Berechnungsformel lautet: Br = (B x Auflösung) / 720. Das Ergebnis wird bis ausschließlich X.5 ab- und ab X.5 aufgerundet.

Merke: Aufgrund der oben beschriebenen Tatsachen wirkt sich womöglich die Änderung des Parameters „**B**“ bei einer bestimmten Auflösung nicht auf die Gesamtbreite des Barcodes aus. Ist z.B. B = 12, dann ergibt das bei 150dpi 12 * 150 / 720 = 2,5 und aufgerundet 3 Pixel. Wird der Wert von **B** auf 15 erhöht, ergibt das 15 * 150 / 720 = 2,9, also aufgerundet immer noch 3 Pixel. Bei einer Auflösung von 300dpi sieht es aber ganz anders aus, denn 12 * 300 / 720 = 5 Pixel und 15 * 300 / 720 = 6,25 also abgerundet 6 Pixel. Dadurch würde bei 150dpi die Änderung des Parameters „**B**“ von 12 auf 15 keine Auswirkung auf die Barcodebreite haben, jedoch bei 300dpi schon.

10.3 Bestimmung der Barcode Gesamtbreite

Oft stellt sich die Frage, wie der Parameter „**B**“ gewählt werden soll, wenn eine bestimmte Barcode Gesamtbreite gefordert wird. In diesem Fall ist folgendermaßen vorzugehen:

- Definieren Sie den Parameter „**B**“ mit einem Wert, z.B. **10**.
- Drucken Sie den Barcode aus und messen dessen Gesamtbreite.
- Bestimmen Sie das Verhältnis zwischen der gemessenen Gesamtbreite und der gewünschten Gesamtbreite.
- Vergrößern bzw. verkleinern Sie den Parameter „**B**“ um den errechneten Faktor.
- Drucken Sie den Barcode erneut.

Merke: Bedingt durch die Auflösung und die Tatsache, dass ein Barcode aus Qualitätsgründen nicht einfach auseinander gezogen werden kann, steigt bzw. verringert sich die Gesamtbreite des Barcodes sprunghaft. Wenn z.B. in einem Barcode 50 Striche vorhanden sind und Sie vergrößern den Parameter „**B**“ so, dass der Strich um 1 Pixel breiter wird, dann vergrößert sich die Gesamtbreite um mehr als 100 Pixel, da ja auch Lücken um 1 Pixel und die breiten Module sogar um 2 Pixel anwachsen.

Merke: Falls Sie bei einer bestimmten Auflösung nicht die gewünschte Barcode Gesamtbreite erreichen können, erhöhen Sie die Auflösung der Barcodegrafik (Parameter „**RES**“). Dadurch werden die Sprünge kleiner, denn je höher die Auflösung ist, desto kleiner ist ein Pixel.

10.4 Kodierbare Zeichen und Eingabeformat.

Jede Barcode Symbologie verfügt über einen sog. kodierbaren Zeichenvorrat. Der Zeichenvorrat enthält alle Zeichen, die mit einer bestimmten Barcode Symbologie kodiert werden können. Der Versuch, ein Zeichen mit einer Symbologie zu kodieren, das nicht zum Zeichenvorrat dieser Symbologie gehört, führt zu einem kritischen Fehler. Überprüfen Sie deshalb vor der Implementierung einer Barcodelösung für Ihr SAP System, welche Zeichen verschlüsselt werden sollen, und wählen dazu die entsprechende Symbologie. Ein Beispiel für ein solches Problem ist, wenn Zahlen mit Nachkommastellen mit Code 39 verschlüsselt werden sollen. Da das Komma nicht zum Zeichenvorrat von Code 39 gehört, kommt es hier zu einem kritischen Fehler. Eine Lösung in diesem Fall wäre, das Komma durch einen Punkt zu ersetzen (der Punkt gehört zum Zeichenvorrat von Code 39) oder die Symbologie Code 39 Extended oder 128-Autoswitch zu wählen, zu deren Zeichenvorrat das Komma ebenfalls gehört. Dabei ist jedoch äußerste Sorgfalt geboten, denn ein Symbologiewechsel sollte vorher mit allen an der Logistikkette beteiligten Partnern abgesprochen werden, es kann sonst zu unangenehmen Überraschungen kommen, wenn z.B. ein beim Warenempfänger eingesetztes Barcode Lesegerät den neuen Barcode nicht lesen kann. Manche Barcode Symbologieen erfordern auch die Eingabe in einem ganz bestimmten Format, z.B. müssen beim Code EAN-13 genau 12 Ziffern übergeben werden – nicht mehr und nicht weniger. Wird diese Regel nicht eingehalten, führt dies zu einem kritischen Fehler. Diese Regeln finden Sie in der Tabelle im nächsten Kapitel.

10.4.1 Zeichenvorräte der von RBarc+ unterstützten Barcode Symbologieen.

In der folgenden Tabelle wurden die Zeichenvorräte aller von RBarc+ unterstützten Barcode Symbologieen aufgelistet. Zusätzlich befinden sich in der Tabelle weitere Informationen, wie z.B. der Typ der Symbologie und ob die Prüfziffer optional oder obligatorisch berechnet wird. "**JA**" in der Spalte Prüfziffer bedeutet, dass die obligatorische Prüfziffer automatisch berechnet und der Parameter **w_chksum** ignoriert wird. Auch bei "**NEIN**" wird der Parameter **w_chksum** ignoriert aber diesmal keine Prüfziffer berechnet (weil nicht vorgesehen). Bei "**OPTIONAL**" wird der Parameter **w_chksum** ausgwertet und die Prüfziffer nur dann berechnet, wenn der Wert **w_chksum = 'X'** ist.

Bar Code Name	Zeichenvorrat	Prüfziffer	Typ	Kommentar
2 of 5 interleaved	00 – 99	OPTIONAL	2-Breiten	Nur Ziffernpaare erlaubt
German Postal Identcode	11 Ziffern fix	JA	2-Breiten	Aufbau wie 2 of 5 interleaved
German Postal Leitcode	13 Ziffern fix	JA	2-Breiten	Aufbau wie 2 of 5 interleaved
2 of 5 industrial	0-9	OPTIONAL	2-Breiten	
2 of 2 matrix	0-9	OPTIONAL	2-Breiten	
Code 39	0-9 A-Z Das Zeichen '-' (Minus) Das Zeichen '.' (Punkt) Das Leerzeichen Das Zeichen '$' Das Zeichen '/' Das Zeichen '+' Das Zeichen '%'	OPTIONAL	2-Breiten	
Danish PTT 39 Barcode	10 Ziffern fix	JA	2-Breiten	
French Postal 39 A/R	RA + 8 Ziffern RB + 8 Ziffern	JA	2-Breiten	
Code 39 extended	ASCII 0x00 – 0x7F	OPTIONAL	2-Breiten	Sehr breit, erzeugt 2 Symbole pro Zeichen, z.B. a=+A
Codabar	0-9	JA	2-Breiten	4 verschiedene Start- und Stoppzeichen können vom Anwender definiert werden.

Bar Code Name	Zeichenvorrat	Prüfziffer	Typ	Kommentar
MSI	0-1	NEIN	2-Breiten	
MSI (Chk 10)	0-1	JA	2-Breiten	
MSI (Chk 10 10)	0-1 10 10	JA	2-Breiten	2 Prüfziffern
MSI (Chk 11 10)	0-1 11 10	JA	2-Breiten	2 Prüfziffern
Code 93	0-9 A-Z Das Zeichen '-' (Minus) Das Zeichen '.' (Punkt) Das Leerzeichen Das Zeichen '$' Das Zeichen '/' Das Zeichen '+' Das Zeichen '%	JA	4-Breiten	2 Prüfziffern
Code 93 extended	ASCII 0x00-0x7F	JA	4-Breiten	2 Prüfziffern. Sehr breit, erzeugt 2 Symbole pro Zeichen, z.B. a=+A
Code 128 A	ASCII 0x00-0x5F FNC1 FNC2 FNC3 FNC4	JA	4-Breiten	
Code 128 B	ASCII 0x20-0x7E DEL FNC1 FNC2 FNC3 FNC4	JA	4-Breiten	
Code 128C	00-99	JA	4-Breiten	2 Prüfziffern. Sehr breit, erzeugt 2 Symbole pro Zeichen, z.B. a=+A
Code 128 Autoswitch	ASCII 0x00-0x7F	JA	4-Breiten	wechselt automatisch zwischen 128A, 128B und 128C
UCC-128	19 Ziffern fix	JA	4-Breiten	FNC1 wird automatisch nach dem Startzeichen eingefügt
EAN-128	0-9	JA	4-Breiten	FNC1 wird automatisch nach dem Startzeichen eingefügt
EAN/JAN 8	7 Ziffern fix	JA	4-Breiten	
EAN/JAN 8 +2	9 Ziffern fix	JA	4-Breiten	mit 2-Stelligem Add-on
EAN/JAN 8+5	12 Ziffern fix	JA	4-Breiten	mit 5-Stelligem Add-on
EAN/JAN 13	12 Ziffern fix	JA	4-Breiten	
EAN/JAN 13 +2	14 Ziffern fix	JA	4-Breiten	mit 2-Stelligem Add-on
EAN/JAN 13+5	17 Ziffern fix	JA	4-Breiten	mit 5-Stelligem Add-on
UPC-A	11 Ziffern fix	JA	4-Breiten	
UPC-A +2	13 Ziffern fix	JA	4-Breiten	mit 2-Stelligem Add-on
UPC-A+5	16 Ziffern fix	JA	4-Breiten	mit 5-Stelligem Add-on
UPC-E	7 Ziffern fix	JA	4-Breiten	
UPC-A-E +2	9 Ziffern fix	JA	4-Breiten	mit 2-Stelligem Add-on
UPC-A-E+5	12 Ziffern fix	JA	4-Breiten	mit 5-Stelligem Add-on

11 Barcode Einstellungen – kurze Zusammenfassung

In diesem Kapitel finden Sie eine Zusammenfassung aller Parameter, mit der die Barcode Eigenschaften beeinflusst werden können.

Alle Barcode Einstellungen finden in den ABAP Programmen **ZSS_BCSETTINGS12** (für SAPscript) und **ZSF_BC_SETTINGS** (für SmartForms) statt.

Die Barcodeeigenschaften werden je Barcode in einer Formroutine zusammengefasst. Der Name der Formroutine, z.B. **FORM BARCODE01** muss mit dem Wert des Parameters **BARC_IDENT** übereinstimmen. Der Parameter **BARC_IDENT** wird im Formular selbst festgelegt.

Folgende Parameter haben Einfluss auf den Barcode:

Symbologie	Barcode Symbologie (z.B. Code 39 oder Code 128A).
W_chksum	Berechnung der Prüfziffer bei Barcodes, die dieses nicht unbedingt vorschreiben (z.B. Code 39).
B	Breite des Moduls in 1/720 Zoll Einheiten. Bestimmt indirekt die gesamte Barcodebreite.
Unit	Maßeinheit **für Barcode Höhe**, **Margin** und **Offset**.
Margin	Tiefe eines Ausschnittes am unteren Rand des Barcodes für Klarschrift (HRT).
Offset	Breite der Ränder, falls ein Einschnitt mit dem Parameter Margin definiert wurde.
Rot	Barcodedrehung (in 90 Grad Schritten).
Res	Auflösung der Barcode Grafik.

Die genaue Beschreibung der Parameter befindet sich in **Kapitel 7.2**.

Mit weiteren Parametern können Sie das Programmverhalten in einigen Fällen steuern.

error_handling	Legt fest, wie sich das Programm im Fehlerfall verhält. Z.B. kann eine automatische Fehlerkorrektur bei sog. nicht-kritischen Fehlern (z.B. bei negativem Margin) durchgeführt werden.
Input_handling	Legt fest, wie der zu kodierende Wert behandelt werden soll. In der Regel wird er unverändert übernommen, es kann aber mit diesem Parameter festgelegt werden, dass z.B. führende Nullen oder ungültige Zeichen vor der Barcode Generierung entfernt werden sollen.

Die genaue Beschreibung der Parameter befindet sich in **Kapitel 7.1**

12 Besonderheiten des Barcodes 128FREE

Der RBarc Barcode 128FREE ist eine besondere Implementierung des Codes 128, die erlaubt Zeichen zu kodieren, die nicht über die Tastatur eingegeben werden können.
Der Code 128 besteht aus 3 sog. Zeichensätzen. Der Zeichensatz „A“ erlaubt z.B. das Kodieren der Zeichen mit dem ASCII-Wert 0 bis 127. Die Zeichen 0 bis 31 können aber nicht über die Tastatur eingegeben werden. Darüber hinaus stellt der Code 128 die Sonderzeichen FNC1, FNC2, FNC3 und FNC4 zur Verfügung, die auch keinem Tastaturzeichen entsprechen. Aus diesem Grunde wurde in RBarc eine spezielle Behandlung dieser Zeichen implementiert, die es dem Entwickler ermöglicht, die benötigten Zeichen für einen Code 128 selbst zusammen zu stellen. Die Prüfziffer wird natürlich auch einem solchen Fall automatisch berechnet.

Der Code 128 unterstützt folgende, nicht darstellbare Zeichen:

[NUL], [SOH], [STX], [ETX], [EOT], [ENQ], [ACK], [BEL], [BS], [HT],
[LF], [VT], [FF], [CR], [SO], [SI], [DLE], [DC1], [DC2], [DC3],
[DC4], [NAK], [SYN], [ETB], [CAN], [EM], [SUB], [ESC], [FS], [GS],
[RS], [US],
[FNC1], [FNC3], [FNC2], [FNC4],
[CODE A], [CODE B], [CODE C], [SHIFT],

Für die Verschlüsselung aller Zeichen eines Codes 128, der nicht darstellbare Zeichen enthält, ist die Formroutine „**collect_128**“ zuständig. Es gelten dabei folgende Regeln:

- Alle zu verschlüsselnden Zeichen (auch darstellbare) müssen mit dieser Formroutine verarbeitet werden.
- Darstellbare Zeichen können in zusammenhängenden Gruppen an die Formroutine übergeben werden.
- Nicht darstellbare Zeichen müssen einzeln an die Formroutine übergeben werden.
- Alle Zeichen müssen mit der Variablen „DataType“ typisiert werden: Darstellbare Zeichen mit ‚T‘ und nicht darstellbare Zeichen mit ‚F‘.
- Startzeichen und alle eventuellen Wechsel des Zeichensatzes zwischen „A“, „B“ und „C“ müssen manuell vom Programmierer eingegeben werden.

Beipiel:

Es soll folgende Zeichenfolge verschlüsselt werden (Leerzeichen und eckige Klammern im Beispiel dienen nur der Übersichtlichkeit):

[START B] [FNC1] 241GA1 [FNC1] 101274 [FNC1] [CODE C] 370587

Implementierung:

```
encoding = 'Start Code B'.
DataType = 'F'.
perform collect_128 using encoding DataType.

encoding = 'FNC1'.
DataType = 'F'.
perform collect_128 using encoding DataType.

encoding = '241G1A'.
DataType = 'T'.
perform collect_128 using encoding DataType.

encoding = 'FNC1'.
DataType = 'F'.
perform collect_128 using encoding DataType.

encoding = '101274'.
DataType = 'T'.
perform collect_128 using encoding DataType.
```

```
encoding = 'FNC1'.
DataType = 'F'.
perform collect_128 using encoding DataType.

encoding = 'Code C'.
DataType = 'F'.
perform collect_128 using encoding DataType.

encoding = '370587'.
DataType = 'T'.
perform collect_128 using encoding DataType.
```

Wenn der Barcodetyp “128FREE” generiert werden soll, kann die Variable “encoding” leer bleiben. RBarc erkennt automatisch, um welchen Barcode es sich handelt und interpretiert die Variable „encoding“ nicht, sondern verarbeitet stattdessen die Tabelle, die mit der Formroutine „collect_128“ erstellt wurde.

13 Der Barcode GS1-128

Der RBarc Barcode GS1-128 ist der offizielle Nachfolger des Barcodes EAN-128.
Damit dieser Barcode von anderen Barcodes vom Typ Code 128 eindeutig unterschieden werden kann, kodiert er das Sonderzeichen "FNC1" an erster stelle direkt nach dem Start. "FNC1" ist kein darstellbares Zeichen, wird jedoch vom Laserscanner übertragen, so dass die verarbeitende Software prüfen kann, ob es sich tatsächlich um einen GS1-128 Barcode handelt.

Für den Barcode GS1-128 wurde in RBarc+ die Symbologie '128GS1' implementiert. Bei dieser Symbologie wird das Sonderzeichen 'FNC1' an erster Stelle immer automatisch generiert, es darf also nicht explizit angegeben werden.

Nach der Spezifikation für den Barcode GS1-128 können Felder durch Sonderzeichen, wie z.B. "FNC1" oder "GS1" voneinander getrennt werden. Da diese Sonderzeichen nicht über die Tastatur eingeben werden können, wurde in RBarc+ ein spezielles Verfahren entwickelt, dass Ihnen erlaubt, diese Sonderzeichen zu kodieren.
Dazu müssen alle zu kodierenden Daten klassifiziert und an die Formroutine " **collect_128GS1**" übergeben werden. Es wird dabei zwischen 2 Datentypen unterschieden: Text und Funktionscode. Solange es sich bei den zu kodierenden Daten um Daten handelt, die auch über die Tastatur eingegeben werden können (das sind ASCII-Zeichen von 32 dez. bis 126 dez.), werden sie als Textvariablen deklariert. Das geschieht mit dem Statement **Datatype = 'T'**. Textvariablen dürfen beliebige längen haben. Handelt es sich dagegen um Zeichen, die nicht über die Tastatur eingegeben werden können, dann müssen solche Daten als Funktionszeichen deklariert werden. Das geschieht mit dem Statement **Datatype = 'F'**. Funktionsziechen (z.B. **'GS'** dürfen nur einzeln kodiert werden, d.h. wenn zwei Funktionscodes hintereinander stehen sollten, dann muss die Formroutine collect_128GS1 zweimal aufgerufen werden.

Das oben beschrieben Verfahren gilt sowohl für SAPscript als auch für Smart Forms und Interactive Forms

Beipiel:

Es soll die Materialnummer (Variable matnr) und deren Anzahl (Variable meng) getrennt durch das Zeichen "GS" (Group Separator) kodiert werden.

Implementierung Smart Forms und Interactive Forms:

```
encoding = matnr.
datatype = 'T'.
perform perform collect_128gs1  in program zsf_bc_settings12
using encoding DataType.

encoding = 'GS'.
datatype = 'F.
perform perform collect_128gs1  in program zsf_bc_settings12
using encoding DataType.

encoding = meng.
datatype = 'T'.
perform perform collect_128gs1  in program zsf_bc_settings12
using encoding DataType.

BARC_IDENT = 'BARCODE07'.
encoding = 'dummy'.
PERFORM GEN_BARCODE IN PROGRAM ZSF_BC_SETTINGS12
USING encoding barc_ident
CHANGING barc_name checksum encoding_return.
```

Bemerkungen:

- Der passende Code, wie im obigen Bespiel ist als Code im Programmknoten eines Formularfensters einzugeben.
- Es wird davon ausgegangen, dass **matnr** und **meng** aus der Datenbank ermittelt werden
- Da die Implementierung von GS1-128 abwärtskompatibel zu früheren Versionen von RBarc+ sein soll, muss die Variable **encoding** - obwohl sie nicht benötigt wird - mit dem Wert **'dummy'** gefüllt werden, bevor sie an GEN_BARCODE übergeben wird.
- Bei Interactive Forms verwenden Sie ZAF_BC_SETTINGS12 anstatt ZSF_BC_SETTINGS

Implementierung SAPscript:

```
DEFINE &ENCODING& = &MATNR&
 DEFINE &DATATYPE& = 'T'
 PERFORM COLLECT_128GS1 IN PROGRAM ZSS_BC_SETTINGS12
  USING &ENCODING&
  USING &DATATYPE&
 ENDPERFORM

 DEFINE &ENCODING& = &GS&
 DEFINE &DATATYPE& = 'F'
 PERFORM COLLECT_128GS1 IN PROGRAM ZSS_BC_SETTINGS12
  USING &ENCODING&
  USING &DATATYPE&
 ENDPERFORM

 DEFINE &ENCODING& = &MENG&
 DEFINE &DATATYPE& = 'T'
 PERFORM COLLECT_128GS1 IN PROGRAM ZSS_BC_SETTINGS12
  USING &ENCODING&
  USING &DATATYPE&
 ENDPERFORM

 DEFINE &BARC_IDENT& = 'BARCODE07'
 DEFINE &ENCODING& = 'DUMMY'
 DEFINE &XPOS& = '10.00'
 DEFINE &GRAPH_TYPE& = 'OTF'

PERFORM GEN_BARCODE IN PROGRAM ZSS_BC_SETTINGS12
  USING &BARC_IDENT&
  USING &ENCODING&
  USING &GRAPH_TYPE&
  USING &XPOS&
  CHANGING &BARC_NAME&
  CHANGING &USED_LINES&
  CHANGING &ENCODING_RETURN&
  CHANGING &CHECKSUM&
ENDPERFORM
```

Bemerkungen:

- Der passende Code, wie im obigen Bespiel ist als Code im Formular einzugeben.
- Es wird davon ausgegangen, dass **matnr** und **meng** aus der Datenbank ermittelt werden
- Da die Implementierung von GS1-128 abwärtskompatibel zu früheren Versionen von RBarc+ sein soll, muss die Variablen **encoding** - obwohl sie nicht benötigt wird - mit dem Wert **'dummy'** gefüllt werden, bevor sie an GEN_BARCODE übergeben wird.

14 Index

C

D

E

F

G

H

ZSF_BC_FORM 8, 9, 11
ZSF_BC_SETTINGS12 6, 27, 30, 31, 32, 33, 34, 36, 37, 38, 40, 43, 54, 63
ZSS_12 5
ZSS_BC_FORM 8, 10, 11, 12
ZSS_BC_PRINT 8, 10
ZSS_BC_SETTINGS12 6, 13, 15, 16, 17, 23, 26, 38, 44, 54, 63