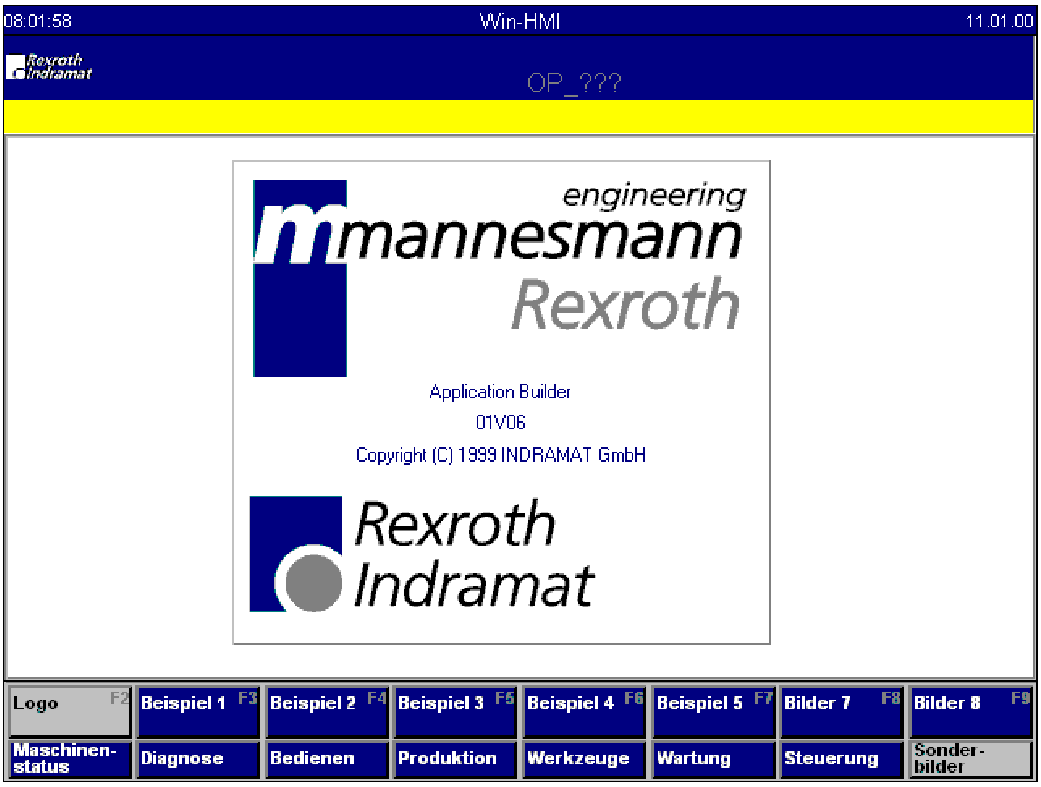


Application Builder
01V06
Copyright (C) 1999 INDRAMAT GmbH

Win-HMI
Application Builder
Getting started

Application Manual

SYSTEM200

DOK-CONTRL-APB********-AW01-EN-P

Title

WIN-HMI
Application Builder
Getting started

Type of Documentation Application Manual

Document Typecode DOK-CONTRL-APB********-AW01-EN-P

Internal File Reference

- Document Number 120-0400-B332-01/EN

Purpose of Documentation This documentation contains an introductory description of how to use the Application Builder for the creation of customer-specific WinHMI user interfaces.

Record of Revisions

Description	Release Date	Notes
120-0400-B332-01/EN	06.00	New issue

Copyright

Validity All rights are reserved with respect to the content of this documentation and the availability of the product.

Published by

Rexroth Indramat GmbH
Bgm.-Dr.-Nebel-Str. 2 • D-97816 Lohr a. Main

Telephone 09352/40-0 • Tx 689421 • Fax 09352/40-4885

http://www.rexroth.com/indramat

Dept. ESM (JA)

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

Table of Contents

1 Overview

The Application Builder is a development system which can be used to easily create operator pictures as an expansion or supplement to the WIN-HMI user interface, with these operator pictures being connected to the process.

The Application Builder permits modification of custom-designed applications which have been created by INDRAMAT with the help of the Application Builder. In addition to this, you can use the Application Builder to create your own applications within the user interface.

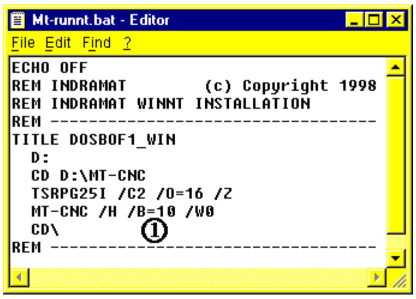
The present documentation is intended to assist you in becoming familiar with the capacity of the Application Builder step by step. The files of the examples described here are supplied on a diskette.

A comprehensive description of the Application Builder is installed as online help on the MTC200. The online help contains a description of all of the classes and methods of the Application Builder.

1.1 Standard HMI

In the normal configuration of the MTC200, the standard HMI is displayed after the control has run up. To achieve this, the “Mt-runnt.bat” file must be started.

Call up HMI

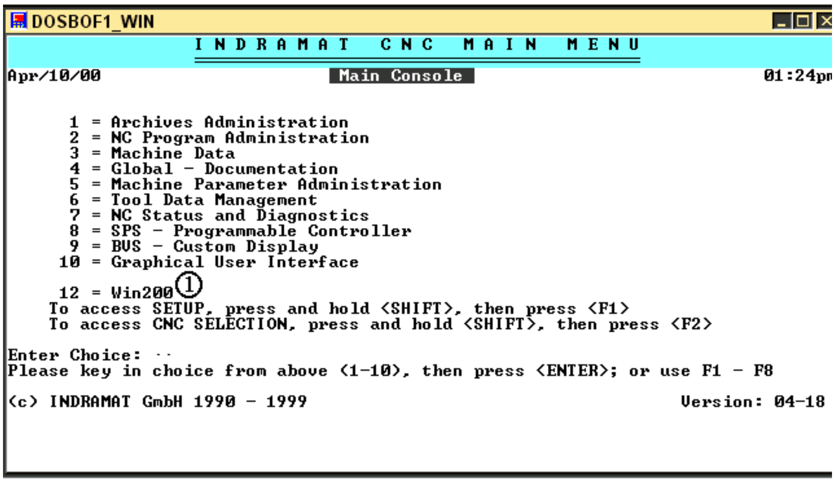


```
ECHO OFF
REM INDRAMAT         (c) Copyright 1998
REM INDRAMAT WINNT INSTALLATION
REM ----------------------------------
TITLE DOSBOF1_WIN
  D:
  CD D:\MT-CNC
  TSRPG25I /C2 /O=16 /Z
  MT-CNC /H /B=10 /W0
  CD\
REM ----------------------------------
```

Bild4.bmp

(1): B=10: Starting the GUI
B=12: Starting the HMI
Without switch B: Start of main menu

Fig. 1-1: Mt-runnt.bat



```
                    INDRAMAT  CNC  MAIN  MENU
Apr/10/00                    Main Console                        01:24pm

      1 = Archives Administration
      2 = NC Program Administration
      3 = Machine Data
      4 = Global - Documentation
      5 = Machine Parameter Administration
      6 = Tool Data Management
      7 = NC Status and Diagnostics
      8 = SPS - Programmable Controller
      9 = BVS - Custom Display
     10 = Graphical User Interface

     12 = Win200
    To access SETUP, press and hold <SHIFT>, then press <F1>
    To access CNC SELECTION, press and hold <SHIFT>, then press <F2>

Enter Choice: ..
Please key in choice from above (1-10), then press <ENTER>; or use F1 - F8

(c) INDRAMAT GmbH 1990 - 1999                              Version: 04-18
```

Bild3.bmp

(1): Calling up HMI

Fig. 1-2: Start from the main menu

From the main menu, start HMI by entering <12> and pressing <Enter>.

Exit HMI

Exit the WIN-HMI user interface by pressing <Shift> + <F4>.

HMI standard

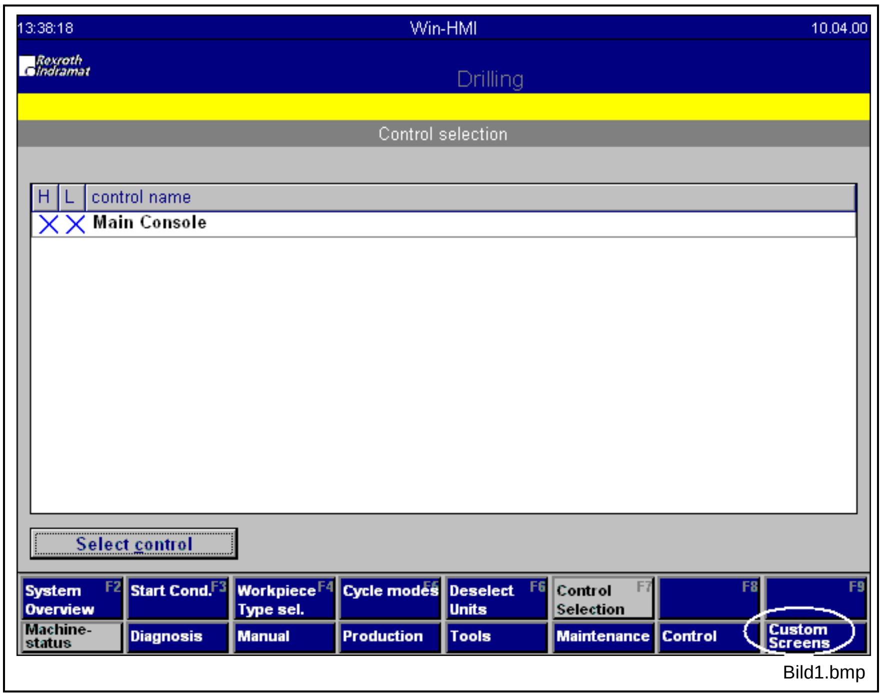
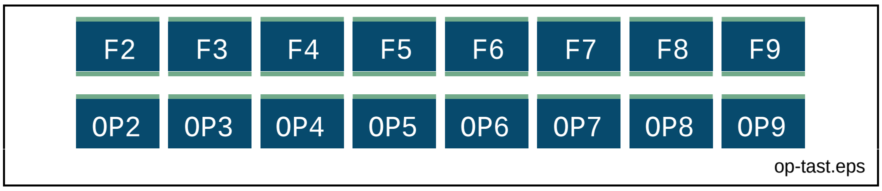


Bild1.bmp

Fig. 1-3: Standard HMI with machine operator menu



op-tast.eps

Fig. 1-4: Range selection by means of soft keys

Below, standard HMI will stand for the HMI user interface, which does not contain any custom-designed applications. A typical feature of the standard HMI is the double-line arrangement of the operating elements (see Fig. 1-4: Range selection by means of soft keys).

In the lower line, the OP keys (OP = operator) allow selection of the range, each of which comprises typical functions for a specific operating situation.

In the upper line, the F keys (F = function) allow to call up functions within the range selected.

<OP9> key ➔ Custom Screens

Select the machine operator menu by using the OP keys. The menus are predetermined and cannot be changed. The <OP9> key is prepared for custom-designed menus. In the standard HMI, pressing this key will not cause any reaction since, here, custom screens are not incorporated.

Start call

Pictures created with the Application Builder can only be started by pressing the <OP9> key, if a call command is entered in the **HMI_OPKey.fkl** FKL file (FKL = Function Key Level). The FKL files are used to organize the soft keys of the HMI and are edited in the Application Builder (see below).

Button	fkl file
<OP2>	HMI_Einschalt.fkl
<OP3>	HMI_Diagnosis.fkl
<OP4>	HMI_Manual.fkl
<OP5>	Prodat_Produktion.fkl
<OP6>	Toolman.fkl
<OP7>	HMI_Wartung.fkl
<OP8>	HMI_NCData.fkl
<OP9>	HMI_SpecialPicture.fkl

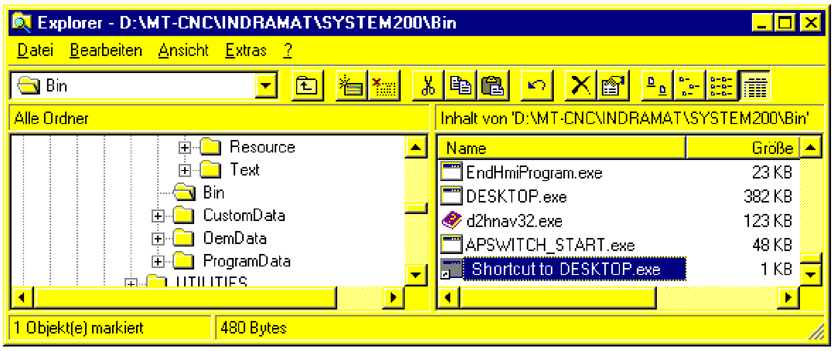
Fig. 1-5: fkl files of the HMI

Separate start of the HMI desktop

To save time if tests are to be performed, the HMI user interface can be started separately (without the MUI and GUI control components). The central program of the HMI user interface is filed to open the working directory (e.g. drive D):

- **D:\MT-CNC\INDRAMAT\SYSTEM200\Bin\Desktop.exe**

It is a practical solution to generate a link of the Windows NT Explorer with "Desktop.exe" (press the right mouse button in the context-sensitive menu).


Explorer - D:\MT-CNC\INDRAMAT\SYSTEM200\Bin
Datei Bearbeiten Ansicht Extras ?
Bin
Alle Ordner
Resource
Text
Bin
CustomData
OemData
ProgramData
UTILITIES

Inhalt von 'D:\MT-CNC\INDRAMAT\SYSTEM200\Bin'

Name	Größe
EndHmiProgram.exe	23 KB
DESKTOP.exe	382 KB
d2hnav32.exe	123 KB
APSWITCH_START.exe	48 KB
Shortcut to DESKTOP.exe	1 KB

1 Objekt(e) markiert 480 Bytes

Bild1-1.bmp

Fig. 1-6: Generating a link with "Desktop.exe"

Then drag the link to the Windows desktop while holding down the left mouse button.

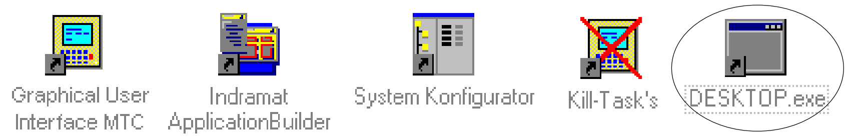


Bild1-2.bmp

Fig. 1-7: Icons for starting the MTCNC components

Closing the desktop

If the desktop has been started separately, it must be exited with Kill Tasks. Only then can it be restarted again!!!

If you press the <Ctrl>+<s> keys, the borders of the desktop will be visible. This makes the Windows task bar accessible. You can now close "Desktop.exe" (<right mouse button>).

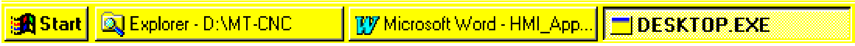


Bild4_4.bmp

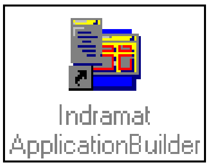
Fig. 1-8: Windows task bar

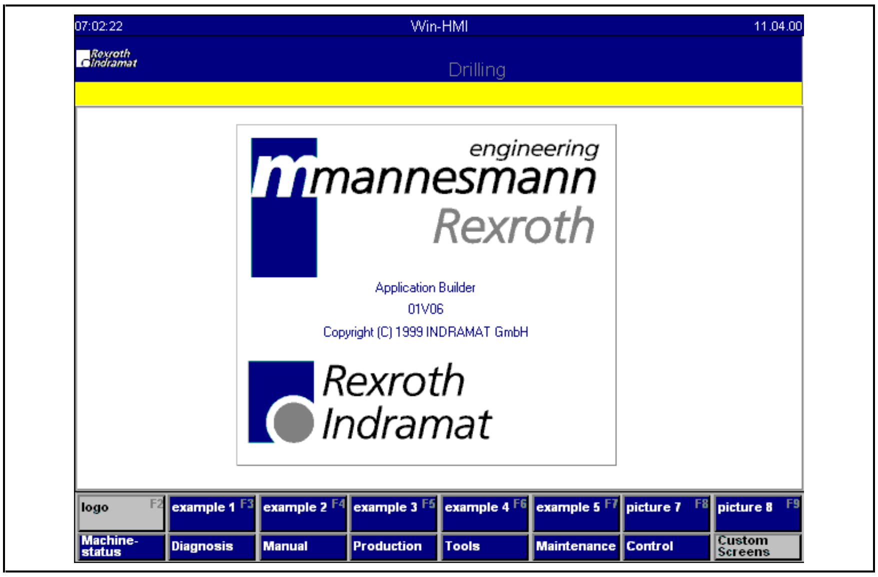
1.2 Application Builder with FKL Editor

The Application Builder is an independent development tool for creating pictures for the HMI user interface.

Starting the Application Builder

Start the Application Builder from the Windows user interface. Fig. 1–9 shows the Application Builder after having been opened. In the figure, the Help menu with “About the Product…”((?)) has also been clicked. The version of the Application Builder is 01V06.





Application Builder
01V06
Copyright (C) 1999 INDRAMAT GmbH

Fig. 1-9: Application Builder version 01V06

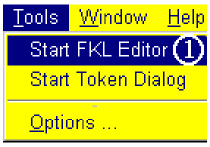
If you wish to open the HMI_OPkey.fkl file, which contains the definitions of the OP keys, you must open the FKL Editor of the Application Builder. Use this Editor to edit the FKL files.

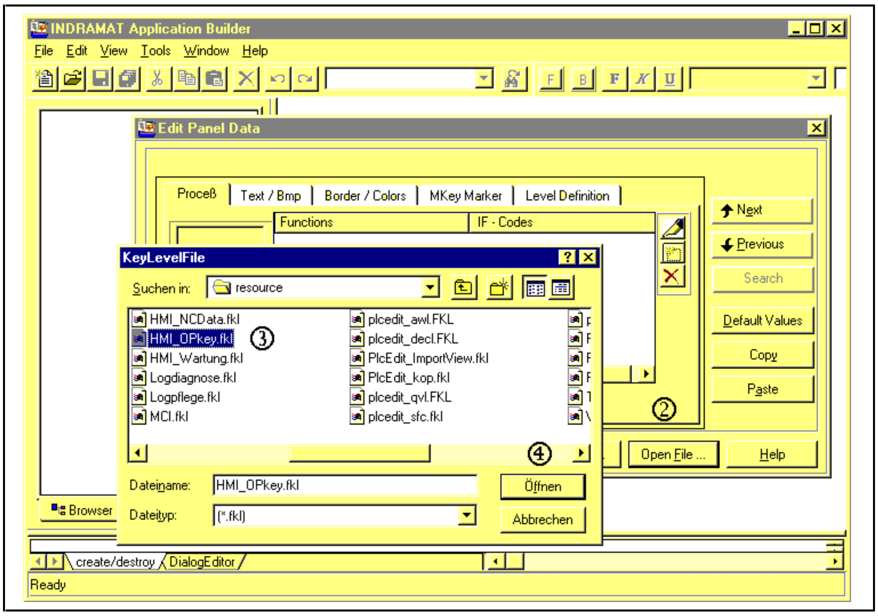
FKL Editor and HMI_OPkey.fkl

It is not necessary to modify the “HMI_Opkey.fkl” file for calling up custom screens. This file is interesting in that it contains the item for starting the special pictures. This file can also be found on the example diskette under “Example 1”.

Calling up the FKL Editor

After you have opened the Application Builder, the FKL Editor is started:





Suchen in: resource

Dateiname: HMI_OPkey.fkl

Dateityp: (*.fkl)

Fig. 1-10: Opening the “HMI_OPkey.fkl” file in the FKL Editor

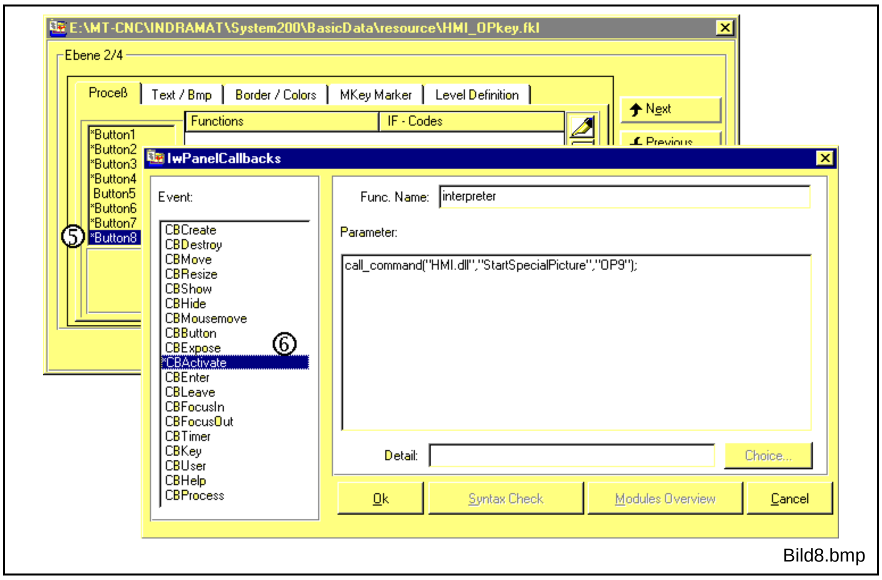
Follow the steps below to define the command which is to be triggered in the WIN-HMI by pressing the OP9 key.

(1) Start the FKL Editor (Tools menu ➔ Start FKL Editor)

(2) Click the <Open File> button in the FKL Editor.

(3) The “HMI_OPkey.fkl” file resides in the following path:
[Drive]:\mt-cnc\indramat\system200\basicdata\resource\
Highlight the file.

(4) Press the <Öffnen> button to load the file selected in the FKL Editor. Its content is divided in tabs:



Func. Name: interpreter

Parameter:

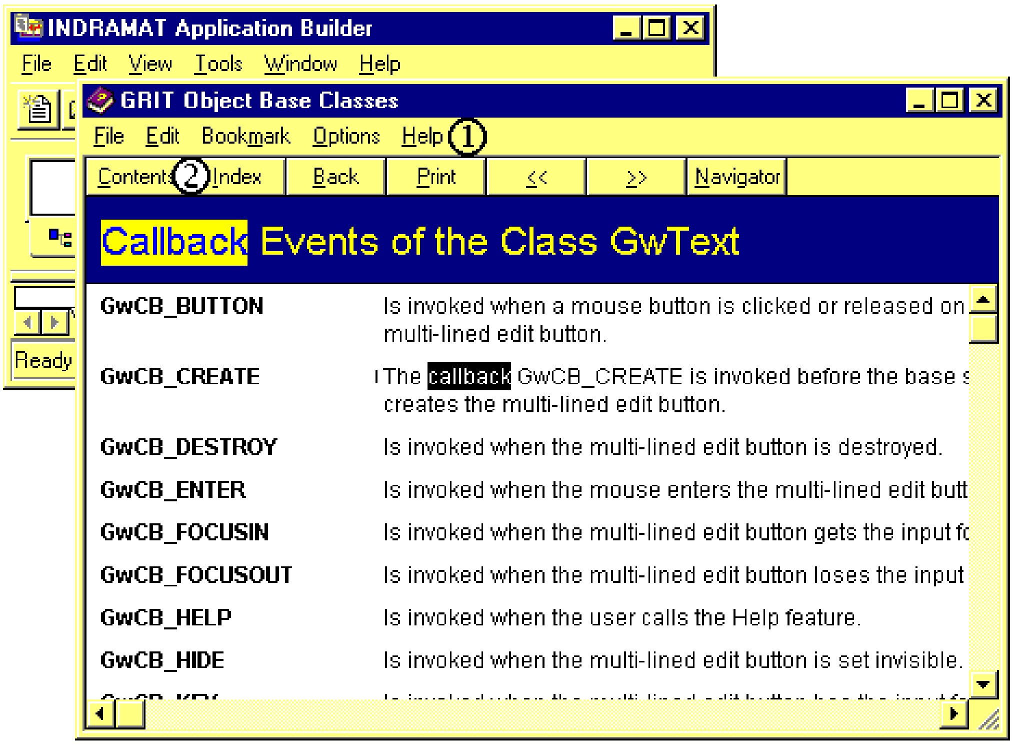
```
call_command("HMI.dll","StartSpecialPicture","OP9");
```

Fig. 1-11: Opening the “HMI_OPkey.fkl” file (continued)

(5) Select the “*Button8” item on the **Process** tab. If a list item is marked with “*”, then at least one “Callback” (command call) is written behind this button. To view or and/or edit the program code of this command call, open the IwPanelCallbacks dialog box by *double-clicking* ***Button8.**

(6) The selection list of the **IwPanelCallbacks** dialog lists all events which can be triggered by the previously selected button. The events marked with “*” possess a callback. A description of the events can be found in the Online Help.

Online Help Example of using the Online Help:


INDRAMAT Application Builder
File Edit View Tools Window Help
Ready
GRIT Object Base Classes
File Edit Bookmark Options Help
1
Contents
2
Index Back Print << >> Navigator

Callback Events of the Class GwText

GwCB_BUTTON	Is invoked when a mouse button is clicked or released on multi-lined edit button.
GwCB_CREATE	The callback GwCB_CREATE is invoked before the base s creates the multi-lined edit button.
GwCB_DESTROY	Is invoked when the multi-lined edit button is destroyed.
GwCB_ENTER	Is invoked when the mouse enters the multi-lined edit butt
GwCB_FOCUSIN	Is invoked when the multi-lined edit button gets the input fc
GwCB_FOCUSOUT	Is invoked when the multi-lined edit button loses the input
GwCB_HELP	Is invoked when the user calls the Help feature.
GwCB_HIDE	Is invoked when the multi-lined edit button is set invisible.

Bild80.bmp

Fig. 1-12: Calling up the “Workbench” Online Help

(1) Call up the “Workbench” help in the Application Builder.

(2) Make your selection under “Contents” or “Search”.

Callback for an internal start call

A callback is a program block (comparable with a subprogram) which is started by an event. In the example below, the event “CBActivate” (pressing of <OP>) starts the callback in the edit field.

The example

```
call_command("HMI.dll", "StartSpecialPicture", "OP9");
```

starts an internal routine (StartSpecialPicture) in “HMI.dll”. This routine causes the

- HMI_SpecialPicture.cgw and
- HMI_SpecialPicture.fkl

files to be called up in the

- [Drive]:\MT-CNC\INDRAMAT\SYSTEM200\CustomData\Resource\

folder.

The scope of delivery of the standard HMI does not include these files. That is the reason why there is no reaction when <OP> is pressed.

The following chapter describes how the user can develop his own operator pictures by creating these files.

2 Custom-Designed Expansion (HMI_SpecialPicture)

2.1 Directory Structure and Files

Applications can be incorporated in the HMI user interface in different ways. The strategy described here is supported by INDRAMAT. This strategy should be used as a standard by first users. The requisite files and the start call (see 1-6) have been prepared. The files can be found under the joint path:

[Drive]:\Mt-cnc\Indramat\System200\...

Observe the branching in ...\BasicData\... and ...\CustomData\... :

Folder with template files	Standard HMI	HMI with application
...\BasicData\Resource\ HMI_OPkey.fkl	**“HMI_OPkey.fkl”** <OP9> calls HMI_SpecialPicture.cgw	
...\CustomData\Resource\ Example HMI_SpecialPicture.fkl	“**Example** HMI_SpecialPicture.fkl”, is not called up (unknown)	rename with file **“HMI_SpecialPicture.fkl”**
...\CustomData\Resource\ Example HMI_SpecialPicture.cgw	“**Example** HMI_SpecialPicture.cgw”, is not called up (unknown)	rename with file **“HMI_SpecialPicture.cgw”**

Fig. 2-1: Already prepared files for incorporating applications

The **“HMI_OPkey.fkl”** file resides in the ...\basicData\... folder. It will be taken into consideration in an update by INDRAMAT. It does not contain any special data for the applications except the callback which starts the application when the <OP9> key is pressed.

The starting files for an application are the following files:

- “HMI_SpecialPicture.fkl”
- “HMI_SpecialPicture.cgw”

cgw file

A cgw file is a so-called resource file (binary file), which contains the data on the user interface.

In our case, **“HMI_SpecialPicture.cgw”** is a resource file predefined by INDRAMAT and “empty” upon delivery. This file determines the outer scope and the internal incorporation of the custom-designed application in the HMI interface.

The **“HMI_SpecialPicture.cgw”** file permits accommodation of further windows (application windows, dialog windows, MDI windows), which are created by the user and are, in turn, cgw files themselves.

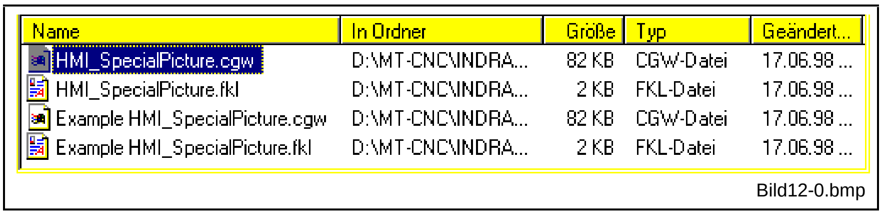
“HMI_SpecialPicture.cgw” (Original) in the HMI

Renaming the files

The simplest way to generate the “HMI_SpecialPicture.fkl” and “HMI_SpecialPicture.cgw” files is by **Copying** and **Renaming** the files

- “Example HMI_SpecialPicture.fkl”
- “Example HMI_SpecialPicture.cgw”

We recommend to retain either file in its original state.



Name	In Ordner	Größe	Typ	Geändert...
HMI_SpecialPicture.cgw	D:\MT-CNC\INDRA...	82 KB	CGW-Datei	17.06.98 ...
HMI_SpecialPicture.fkl	D:\MT-CNC\INDRA...	2 KB	FKL-Datei	17.06.98 ...
Example HMI_SpecialPicture.cgw	D:\MT-CNC\INDRA...	82 KB	CGW-Datei	17.06.98 ...
Example HMI_SpecialPicture.fkl	D:\MT-CNC\INDRA...	2 KB	FKL-Datei	17.06.98 ...

Bild12-0.bmp

Fig. 2-2: Original and renamed files

The application can now be run. It is displayed with an “empty” picture and with the FKey bar activated:

HMI_SpecialPicture.cgw (original)

DESKTOP.exe

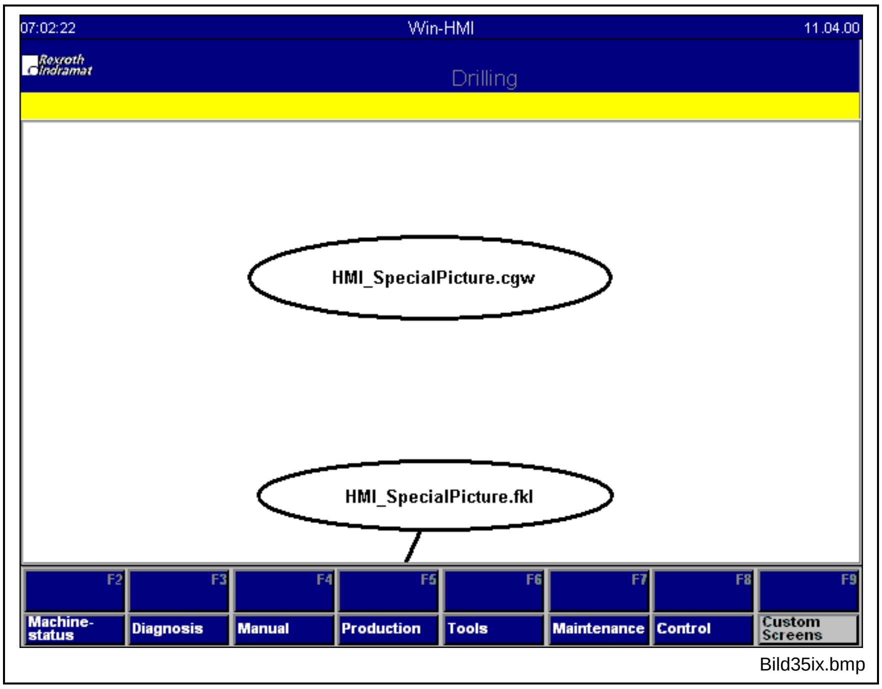


Bild35ix.bmp

Fig. 2-3: “Empty” application by renaming “Example…” with “HMI_SpecialPicture.*”

Note: The starting files for the following examples also bear the name **HMI_SpecialPicture.***. They must be stored separately from the “empty” files (e.g. by being renamed).

“HMI_SpecialPicture.cgw” (Original) in the Application Builder

The cgw files for custom-designed operator pictures are filed in the

- **\MT-CNC\INDRAMAT\SYSTEM200\CustomData\Resource\** folder. The files contained in this folder are not re-created when the System 200 software is updated (contrary to the ...\BasicData\... folder).

Opening a cgw file

First of all, open the cgw file using the Application Builder:

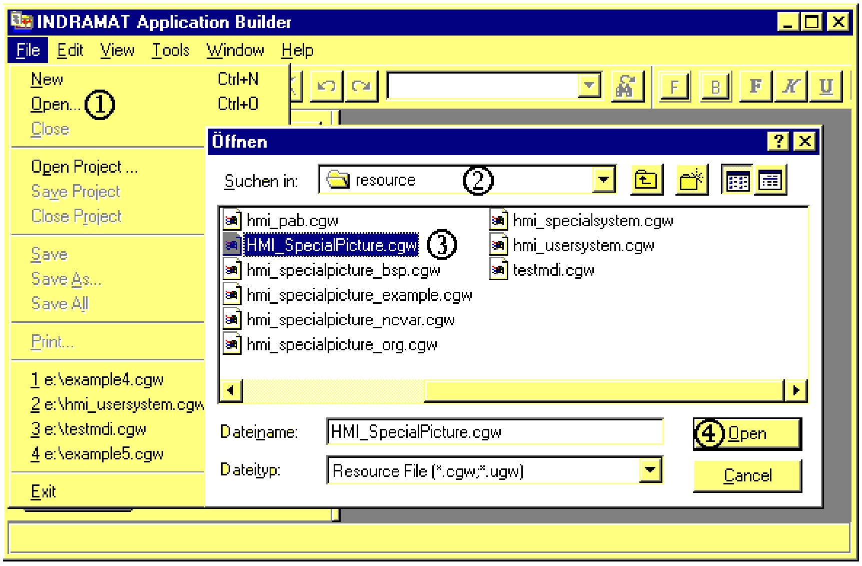


Bild11.bmp

Fig. 2-4: Opening a cgw file in the Application Builder, Part 1

(1) Select <Open…> in the “File” menu.

(2) Select the ...\MT-CNC\INDRAMAT\SYSTEM200\CustomData\Resource\ folder.

(3) Highlight and

(4) <Open> the “HMI_SpecialPicture.cgw” file.

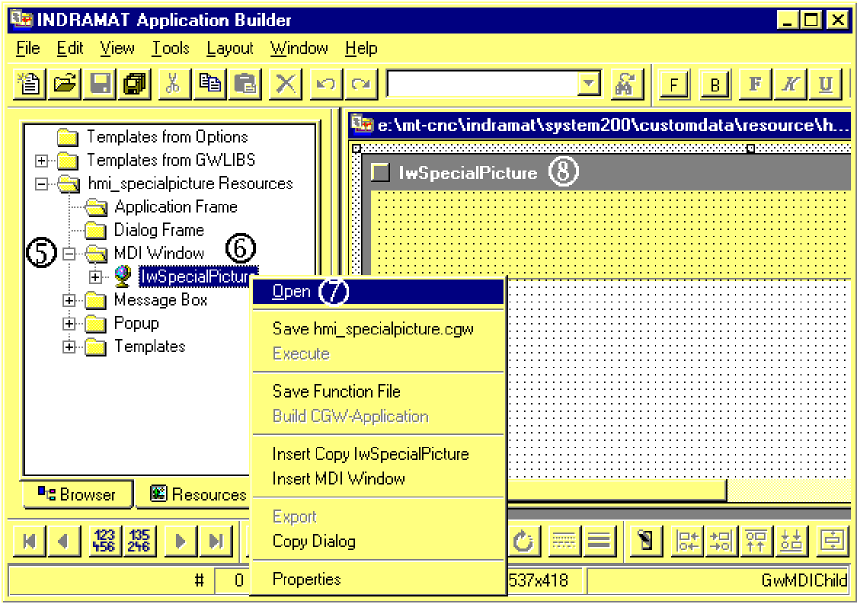


Bild11-1.bmp

Fig. 2-5: Opening a cgw file in the Application Builder, Part 2

Next, the IwSpecialPicture MDI window must be opened in the graphic editor of the Application Builder:

(5) Open the “MDI Window” by **clicking** the node (+) or by **highlighting** it and pressing <Enter>.

(6) Open “IwSpecialPicture” by pressing the right mouse button (the context-sensitive menu pops up) or by **highlighting** it and pressing <Enter>.

(7) Click **<Open>** in the context-sensitive menu.

(8) IwSpecialPicture is displayed.

Explanations on the Navigator

In the “Resources” tab of the Navigator, the structure of a resource file is shown in the form of a tree.

The tree consists of nodes for application windows, dialog windows, MDI windows, message windows, pop-up menus, and templates.

After a resource file has been created, the nodes are empty initially. By using the context-sensitive menus of the nodes (right mouse button), windows can be inserted and then opened in the dialog editor for editing.

The root of a user interface system is an object of the application window class. An empty application window already contains a Container provided as the background object for the objects of your user-specific design.

The first object in any branch of the hierarchy tree is also called parent object; each successor is called child. Children can, in turn, be a parent object themselves. All children of one parent are called brothers and sisters.

Each object knows its parent (parent()) and its children (child()), but not its brothers or sisters. This hierarchy results in a unique behavior during opening and closing of the individual windows.

Likewise, all dialog windows, message windows, and pop-up menus, which can be called from an application window, are children of this application window. After having been called up, they will be inserted in the hierarchy tree as children of the particular window.

An MDI window (Multiple-Document Interface) is a special dialog window. It is the user interface of an application which supports several child windows within one environment window (Container).

An MDI application can only be realized using the Application Builder after an MDIContainer has first be assigned to a frame or a background object (e.g. IwContainer). This MDIContainer is responsible for managing the child windows.

2.2 “HMI_SpecialPicture.cgw” (“Logo” as Example)

Starting from the files mentioned above, a variety of applications can be created.

“Logo” example in HMI

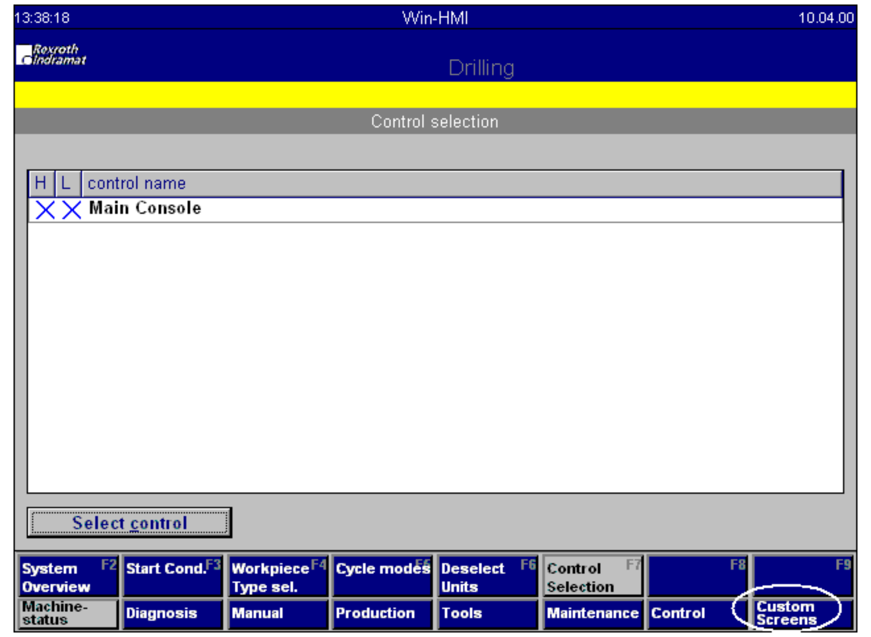
The first example shows a designated FKey bar and the Info windows of the Application Builder.

Requirements: For starting up, the supplied files listed below must be copied to the pertinent folders:

- [Drive]:\Mt-cnc\INDRAMAT\System200\CustomData\resource\ **HMI_Specialpicture.cgw**
- [Drive]:\Mt-cnc\INDRAMAT\System200\CustomData\resource\ **HMI_Specialpicture.fkl**
- [Drive]:\Mt-cnc\INDRAMAT\System200\CustomData\resource\ **HMI_Specialpicture_DE.txt**
- [Drive]:\MT-CNC\SYSTEM200\CustomData\Bitmap\ **Indramat.bmp**
- [Drive]:\MT-CNC\SYSTEM200\CustomData\Bitmap\ **Rexroth.bmp**

Note: Save files with identical names by renaming them in the Windows Explorer !

Starting the HMI user interface



H	L	control name
╳	╳	Main Console

Bild1.bmp

Fig. 2-6: Opening screen of the HMI interface

Calling Custom Screens with “OP9”

⇒ Press the <OP9> key (or click the <Custom Screens> button). The following screen is displayed:

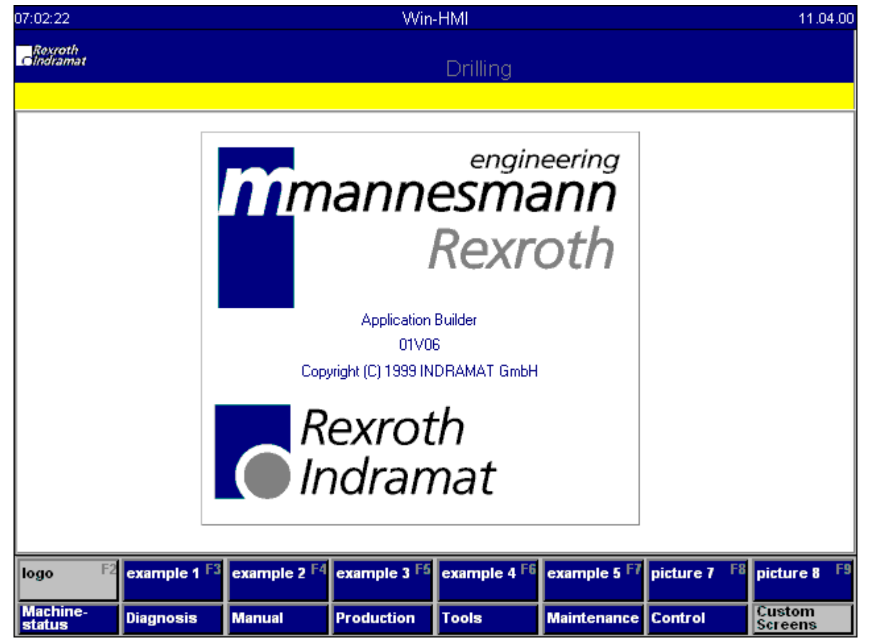


Bild2.bmp

Fig. 2-7: Logo example in HMI

It is also possible to call up the opening screen from the various menus of the “Custom Screens” (still empty at present):

⇒ Press the <F2> key or click the <Logo> button: The screen shown above appears.

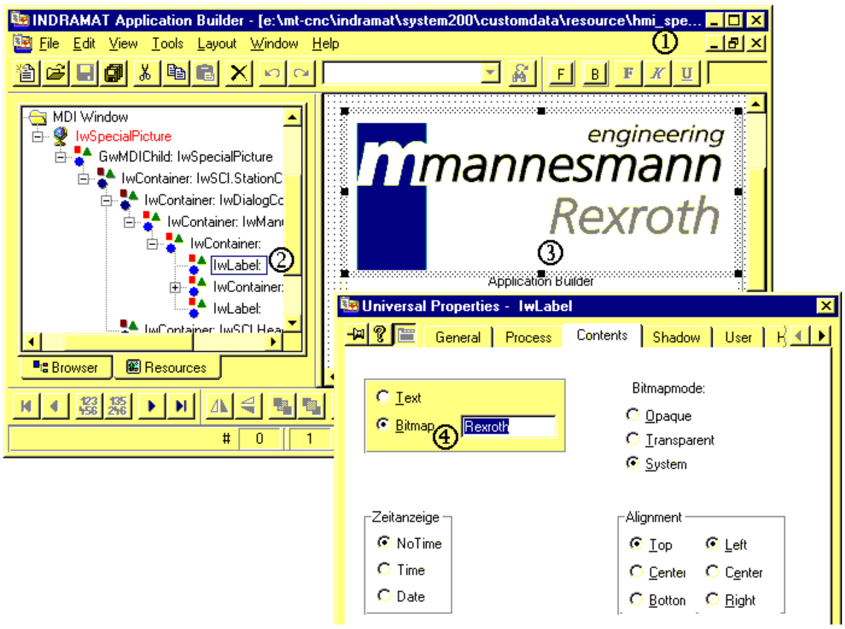
“Logo” example in the Application Builder

Requirements: The **HMI_SpecialPicture.cgw** file is open in the Application Builder.

Inserted Bitmaps

The resources of the Specialpicture.cgw file contain an **IwSpecialpicture** MDI window. This MDI window contains the Info message (see Fig. 2-7).

The internal structure of the Info message can be viewed in the Navigator (see Fig. 2-8):



○ Text
● Bitmap Rexroth

Bitmapmode:
○ Opaque
○ Transparent
● System

Zeitanzeige
● NoTime
○ Time
○ Date

Alignment
● Top ○ Cente ○ Botton
● Left ○ Center ○ Right

Bild36ix.bmp

(1): HMI_SpecialPicture.cgw opened
(2): IwLabel in the Navigator
(3): IwLabel in the window
(4): Rexroth.bmp

Fig. 2-8: “Logo” in the Application Builder

Operator steps for displaying the inserted bitmaps:

⇒ Click IwLabel ② in the Navigator.
The IwLabel ③ is marked in the window.

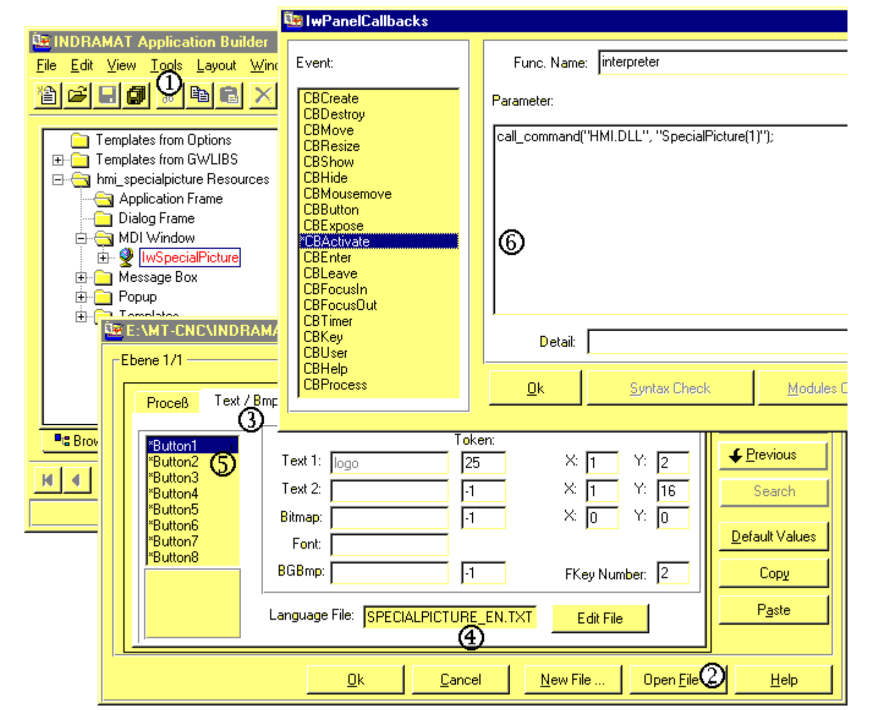
⇒ Double-click IwLabel ② in the Navigator.
The Properties window of IwLabel is opened.

⇒ Open the **Options** tab.
The bitmap name **Rexroth** is displayed (4).

Enter the name of the bitmap that is to be displayed in the IwLabel selected in the edit window (4).

CBActivate Callback of <F2>

The Info window is opened by pressing the <F2> button. Internally, this is achieved by the CBActivate callback of Button1 (F2). The callback is entered using the FKL Editor.



Func. Name: interpreter

Parameter: call_command("HMI.DLL", "SpecialPicture(1)");

Detail:

		Token:				
Text 1:	logo	25	X:	1	Y:	2
Text 2:		-1	X:	1	Y:	16
Bitmap:		-1	X:	0	Y:	0
Font:						
BGBmp:		-1		FKey Number:		2

Language File: SPECIALPICTURE_EN.TXT

Bild9ix.bmp

(1): Calling up the FKL Editor.
(2): Opening the FKL file.
(3): Selecting the Text / Bmp tab.
(4): Pertinent language file.
(5): Highlighting the Button1 <Logo>.
(6): Callback of Button1

Fig. 2-9: Specialpicture.fkl in the Editor

Operator steps for displaying the CBActivate callback of Button1:

⇒ Call up the FKL Editor (1).
The FKL Editor window is opened.

⇒ Click the <Open File> button (2) and select **HMI_Specialpicture.fkl.**

⇒ Select the Text / Bmp tab (3).
The pertinent language file (4) and the button assignment are displayed.

⇒ Double-click Button1 (5).
The callback selection list appears.

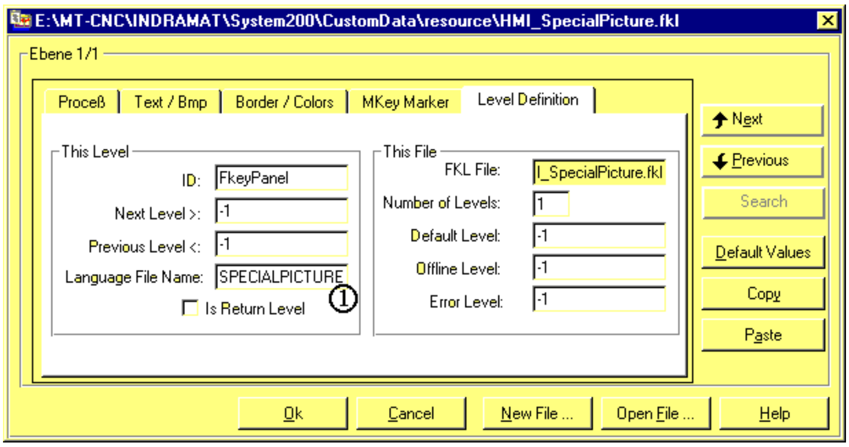
⇒ Click CBActivate (6).
The content of CBActivate is displayed:

```
call_command("HMI.DLL,"SpecialPicture(1)");
```

The calls for the HMI special pictures are described below on Page 2-11.

Language file assignment

The texts for the FKeys are stored in the **HMI_Specialpicture_DE.txt** language file. The assignment of this language file to our FKeys is defined in the HMI_SpecialPicture.fkl file:


E:\MT-CNC\INDRAMAT\System200\CustomData\resource\HMI_SpecialPicture.fkl

This Level

ID: FkeyPanel

Next Level >: -1

Previous Level <: -1

Language File Name: SPECIALPICTURE

□ Is Return Level

This File

FKL File: I_SpecialPicture.fkl

Number of Levels: 1

Default Level: -1

Offline Level: -1

Error Level: -1

Bild33ix.bmp

(1): HMI_SPECIALPICTURE language file

Fig. 2-10: “Level Definition” tab in the FKL Editor

Operator steps for defining the language file:

⇒ Call up the FKL Editor.
The FKL Editor window is opened.

⇒ Open **HMI_Specialpicture.fkl** with <Open File>.

⇒ Select the “Level Definition” tab.
The picture shown above appears (see Fig. 2-10).

The edit window (1) displays the language file assigned, or it can be used to define a new assignment.

Note The text files are designed for multilingual use. In our example, “HMI_SpecialPicture” has been entered as the language file (see (1)). In fact, this file exists in German language and, in this case, has the name "HMI_SpecialPicture_de.txt".

	German	**English**
Folder	...\CustomData\text\de\	...\CustomData\text\en\
File name	HMI_SpecialPicture_**de**.txt	HMI_SpecialPicture_**en**.txt
Selection of the language file (①)	HMI_SpecialPicture	

Fig. 2-11: Example of a bilingual file

The language selection itself is set in the main menu of the control unit:

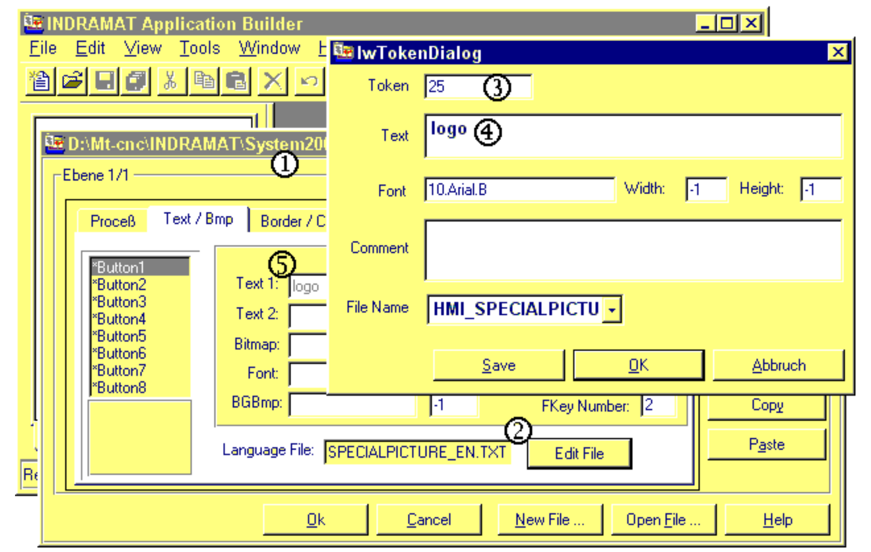
Requirements: The main menu (see Fig. 1-2) is open.

⇒ Press <SHIFT>+<F1> to select the SETUP menu.
The window for selecting the language appears.

⇒ For instance, press <F2> for the English language.

⇒ Exit the main menu.
All of the control programs (MUI, GUI, HMI, SPS, etc.) are displayed in English language.

TokenDialog The text of the button is entered using the TokenDialog:



Token: 25

Text: logo

Font: 10,Arial,B Width: -1 Height: -1

Comment:

File Name: HMI_SPECIALPICTU

Text 1: logo

Text 2:

Bitmap:

Font:

BGBmp: -1

FKey Number: 2

Language File: SPECIALPICTURE_EN.TXT

Bild34ix.bmp

(1): HMI_SpecialPicture.fkl
(2): HMI_SpecialPicture_DE.txt language file
(3): Token 25 for Button1
(4): Text entry
(5): Display in the fkl file

Fig. 2-12: IwTokenDialog for entering the FKey text

Operator steps for editing the language file:

Requirements: FKL Editor with open HMI_SpecialPicture.fkl (1)

⇒ Open the **Text / Bmp** tab.

Button1 is highlighted; the **HMI_Specialpicture_DE.txt** (2) file which is assigned as language file in the example is displayed.

⇒ Click the **<Edit File>** button.

The **IwTokenDialog** window appears.

Note Starting with Token=25, the text file contains the texts for the F buttons.

⇒ Enter Token=25 in the edit window (3).

The text window displays the button text **Logo** (4). The text in this window can be changed.

At the same time, the text is displayed on the tab (5).

2.3 Calling the HMI Special Pictures

Starting with 18V06, special pictures can be called up by means of the HMI.DLL file, by programming specific call_command calls in the HMI_SpecialPicture.fkl file.

This method can also be used to call up interpreter pictures in other OP ranges, by programming the call_command calls in the corresponding FKL files. When the pictures are shown, the appropriate FKL file for the respective OP range is activated.

Call syntax (method 1)

The picture must be contained in the HMI_SpecialPicture.cgw file; the ID of the picture must be defined. Example:

```
call_command("HMI.DLL,"SpecialPicture(1)");
```

The number in parentheses can be within a range of 1..99. The picture associated with the call must possess a defined ID:

No. in the call	ID of the MDI window
1	IwSpecialPicture
2	IwSpecialPicture2
...	...
99	IwSpecialPicture99

Fig. 2-13: Defined IDs for MDI windows in HMI_Specialpicture.cgw

The “Example HMI_SpecialPicture.fkl” FKL example file has the following assignment:

Button	Assignment of CBActivate
F2	**call_command(“HMI.DLL“, “SpecialPicture(1)“);**
F3	call_command(“HMI.DLL“, “SpecialPicture(2)“);
F4	call_command(“HMI.DLL“, “SpecialPicture(3)“);
F5	call_command(“HMI.DLL“, “SpecialPicture(4)“);
F6	call_command(“HMI.DLL“, “SpecialPicture(5)“);
F7	call_command(“HMI.DLL“, “SpecialPicture(6)“);
F8	call_command(“HMI.DLL“, “SpecialPicture(7)“);
F9	call_command(“HMI.DLL“, “SpecialPicture(8)“);

Fig. 2-14: Assignment between FKey and MDI window

Call syntax (method 2)

The picture can be contained in any CGW file and can possess a freely selectable ID.

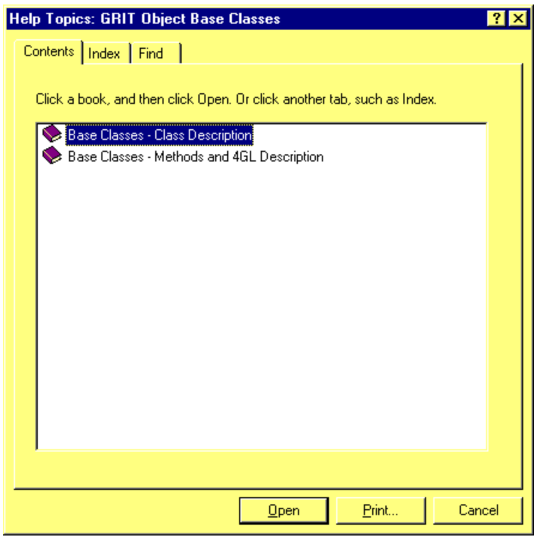
```
call_command("HMI.DLL","SpecialPicture(ANYFILE.CGW,ANYID)");
```

In the example above, the picture “ANYID” is called from the “ANYFILE.CGW” file.

2.4 Online Help for User Interface

The following hlp files are available as Online Help:

• Cc_core.hlp	Object base classes
• Cc_ncore.hlp	Extended device classes



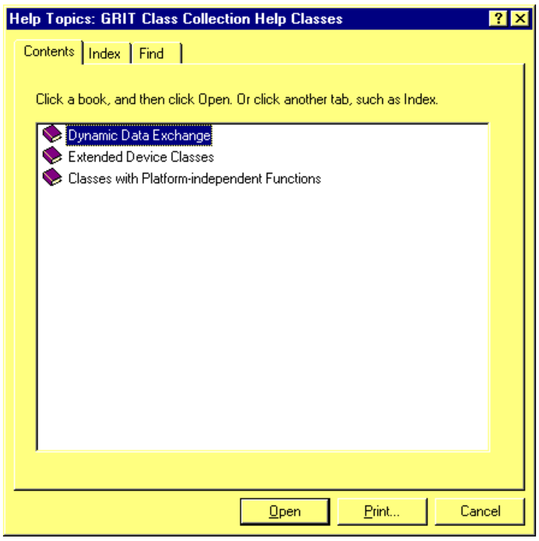
Help Topics: GRIT Object Base Classes

Click a book, and then click Open. Or click another tab, such as Index.

Base Classes - Class Description
Base Classes - Methods and 4GL Description

Bild82.bmp

Fig. 2-15: Object base classes



Help Topics: GRIT Class Collection Help Classes

Click a book, and then click Open. Or click another tab, such as Index.

Dynamic Data Exchange
Extended Device Classes
Classes with Platform-independent Functions

Bild82-1.bmp

Fig. 2-16: Extended device classes

2.5 Creating a New Resource File (cgw file)

The **Example1…Example3** applications which have been prepared for entry into the Application Builder can be found in the **HMI_Example.cgw** file. **Example4** and **Example5** have their own cgw file. All of the examples are called from the **HMI_Example.fkl** file. The calls are summarized in Fig. 2-17.

F Key	Button	Text/BMP	CBActivate
<F2>	*Button1	Logo	call_command("HMI.DLL", "SpecialPicture(1)");
<F3>	*Button2	Example1	call_command("HMI.DLL", "SpecialPicture(HMI_Example.cgw,Example1)");
<F4>	*Button3	Example2	call_command("HMI.DLL", "SpecialPicture(HMI_Example.cgw,Example2)");
<F5>	*Button4	Example3	call_command("HMI.DLL", "SpecialPicture(HMI_Example.cgw,Example3)");
<F6>	*Button5	Example4	call_command("HMI.dll","SpecialPicture(example4.cgw,IwHMI_Con_Bsp2)");
<F7>	*Button6	Example5	call_command("HMI.dll","SpecialPicture(example5.cgw,IwHMI_Con_Bsp2_1)");

Fig. 2-17: Content of HMI_SpecialPicture.fkl

The CBActivate callback for <F2> is written according to the 1st method for the call syntax and the others according to the 2nd method (see 2-11).

Instructions on the exercise below

At this point, we want to describe how a new cgw file is created, which satisfies the requirements of the HMI user interface. New names are assigned to retain the existing applications and calls. If it is intended to call up the new example from the HMI user interface as well, the call would have to be changed in **HMI_Example.fkl.** This change is not made here.

			Comparison of names
<F3>	*Button2	Example1	call_command("HMI.DLL", "SpecialPicture(HMI_Example.cgw,Example1)");
			call_command("HMI.DLL", "SpecialPicture(HMI_Example_**ueb**.cgw,Example1)");

Fig. 2-18: File and window names for the exercise

Creating a resource file (exercise)

Since it is intended to incorporate the new user interface in the HMI user interface, it is easier to create a new cgw file by copying an already existing dialog window (MDI window) from the HMI resources to the new cgw file. By copying the window, the HMI-typcial properties of the new windows, such as size and structure of the window, call properties, etc., are applied. You have already got to know an HMI window, i.e. “lwSpecialPicture”, in the “HMI_SpecialPicture.cgw” resource file. This window is used as a template for the new window:

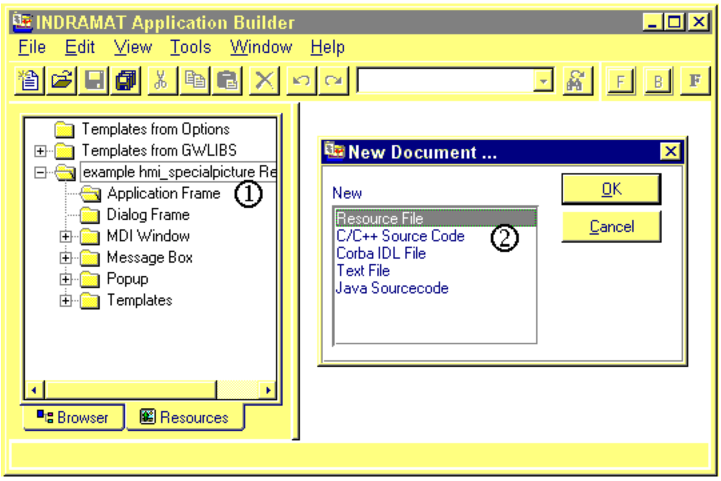
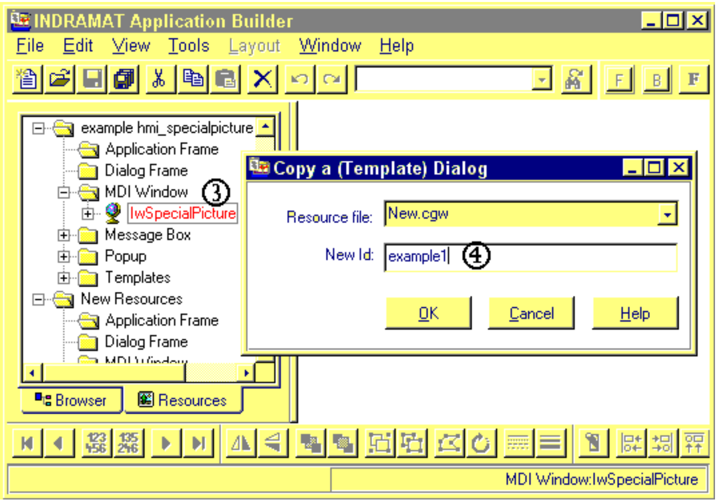


Bild38ix.bmp

Fig. 2-19: Creating a new resource file, step 1

(1) In the Application Builder, load the existing “example HMI_SpecialPicture.cgw” resource file, which contains the “IwSpecialPicture” window to be copied.

(2) Select the “New” item in the “File” menu to open the “New Document…” selection window. Select “Resource File” in the list that appears. Then click <OK> to automatically create the new “New.cgw” resource file.



Resource file: New.cgw

New Id: example1

Bild39ix.bmp

Fig. 2-20: Creating a new resource file, step 2

(3) Copy the “IwSpecialPicture” MDI window to the “New.cgw” resource file. Highlight “IwSpecialPicture” and then select “Copy Dialog” in the context-sensitive menu (right mouse button).

(4) The “Copy a (Template) Dialog” selection window appears. In this window, a new ID is automatically assigned to the dialog window to be copied. This ID must be changed to “Example1”. Select the “New.cgw” target application in the “Resource file”. Click <OK> to complete the copying process.

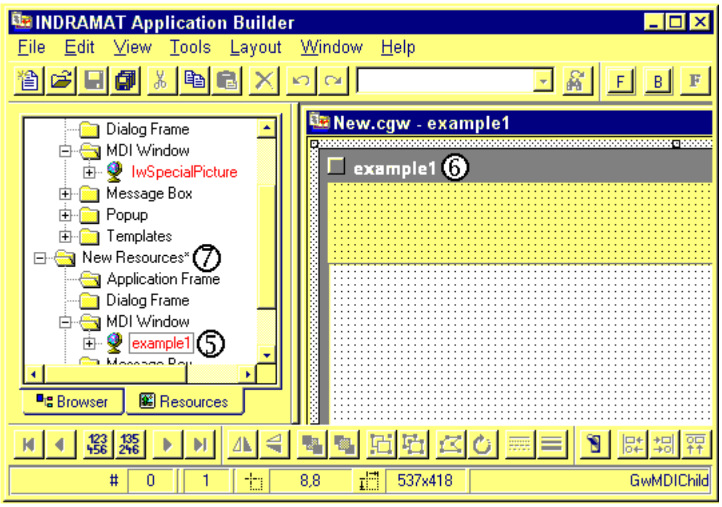


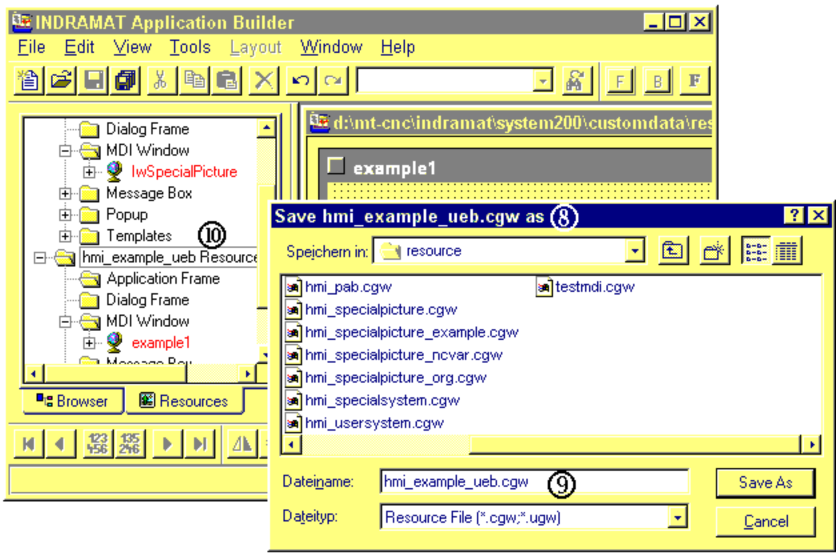
Bild40ix.bmp

Fig. 2-21: Creating a new resource file, step 3

(5) The copied window is displayed by using “Open” in the context-sensitive menu of “Example1”:

(6) “Example1” MDI window in the New.cgw file.

(7) Highlight **New Resources***, and select **Save As…** in the **File** menu.



Speichern in: resource

Dateiname: hmi_example_ueb.cgw

Dateityp: Resource File (*.cgw;*.ugw)

Bild43ix.bmp

Fig. 2-22: Creating a new resource file, step 4

(8) Select the [drive]:\Mt-cnc\INDRAMAT\System200\CustomData\resource\ folder.

(9) Enter the **HMI_Example_ueb.cgw** file name and click <Save As>.

(10)The new file name will be immediately displayed in the Navigator.

You can view the resulting active user interface using the **Preview** button in the “Layout” toolbar:

“Empty” MDI window of HMI

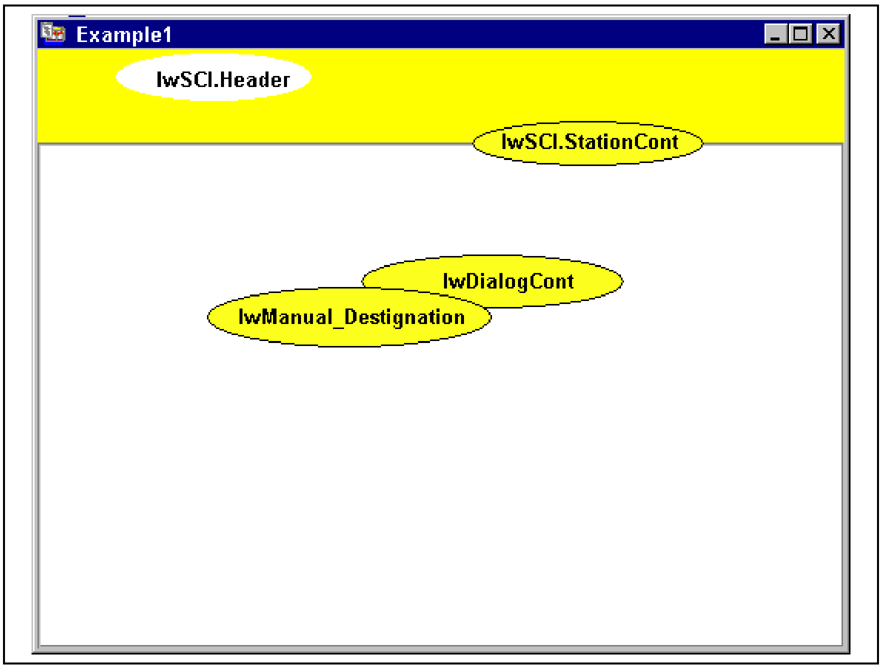


Fig. 2-23: MDI window in the preview of the Application Builder

The “IwSpecialPicture” MDI window which has been prepared for the HMI user interface consists of the “IwSCI.StationCont” IwContainer, which is further divided in “IwDialogCont” and “IwSCI.HeaderCont”. In the example above, there is also the subordinate “IwManual_Destignation” IwContainer. This subdivision is irrelevant for the example.

Calling up the new window in the HMI user interface

If you wish to start the new **HMI_Example_ueb.cgw** file in the HMI user interface without call changes (fkl file), you must rename two files, e.g.:

Old name	New name
HMI_Example.cgw	HMI_Example_org.cgw
HMI_Example_ueb.cgw	HMI_Example.cgw

Fig. 2-24: Example of renaming

Exercise test

DESKTOP.exe

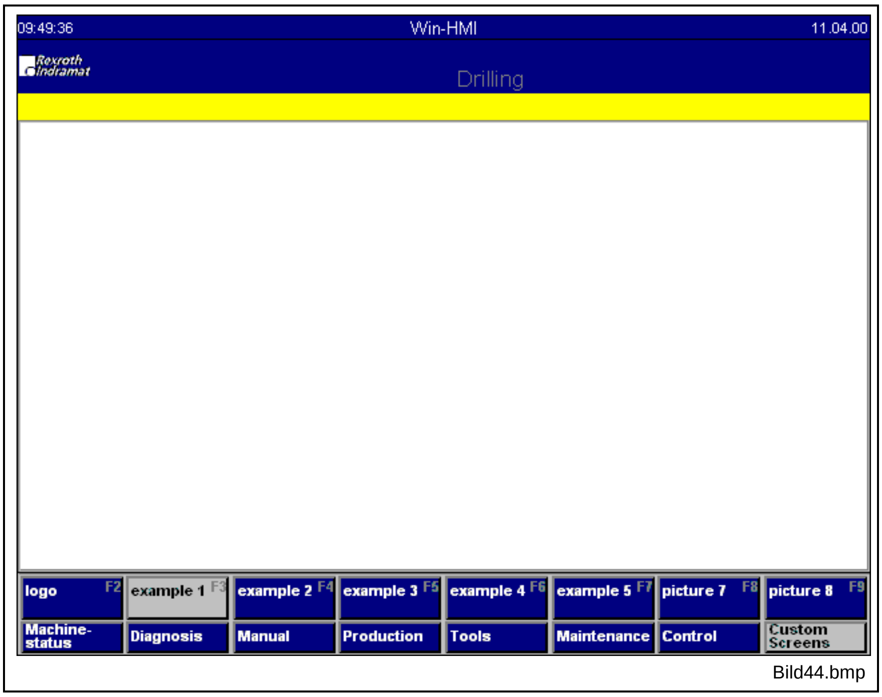


Bild44.bmp

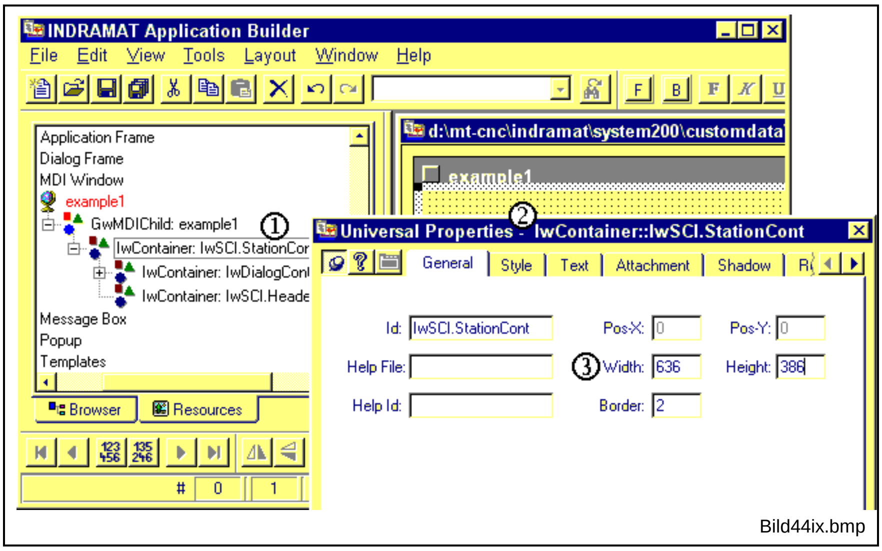
Fig. 2-25: MDI window in the HMI

Placing user interface objects in the window

The INDRAMAT Application Builder provides you with a great number of graphical objects, such as windows, containers, buttons, entry fields, list elements, menus, character fields, and character elements for the layout of the user interfaces of your applications. The **GRIT Object** toolbar of the dialog editor permits quick and easy generation of the objects on your user interface.

Adjusting the container size

The size of the new window should be the same as the free section of the HMI user interface:


INDRAMAT Application Builder
File Edit View Tools Layout Window Help
Application Frame
Dialog Frame
MDI Window
example1
GwMDIChild: example1
IwContainer: IwSCI.StationCor
IwContainer: IwDialogCont
IwContainer: IwSCI.Heade
Message Box
Popup
Templates
Browser
Resources
d:\mt-cnc\indramat\system200\customdata
example1
Universal Properties - IwContainer::IwSCI.StationCont
General
Style
Text
Attachment
Shadow
1
2
3

Id: IwSCI.StationCont
Pos-X: 0
Pos-Y: 0
Help File:
Width: 636
Height: 386
Help Id:
Border: 2

Bild44ix.bmp

Fig. 2-26: Adjusting the window width and height to the HMI user interface

(1) Display the “IwSCI.StationCont” container by selecting it in the Navigator.

(2) Display the Properties window of the container using the context-sensitive menu.

(3) Change the width of the window from 524 (MKeys are not taken into consideration in this measure) to 636.

There are two methods for inserting new objects in the window: Point&Click and Drag&Drop.

First, use the **Point&Click** method to place an object on the application window in this example.

IWContainer

⇒ Click on the **<IWContainer>** button of the **GRIT Objects** toolbar. A container serves as an universal background object. You can use it to place any number of additional objects in it.

⇒ Position the mouse pointer in the left upper corner of the dialog box.

⇒ Start to generate a container while holding down the left mouse button. After you have released the left mouse button, the container will be generated.

Setting the grit

⇒ Move the container to the position (36, 35) while holding down the left mouse button. The respective position of the container is indicated in pixel in the status bar. (If the grit is activated, it can be deactivated by using **<Raster on/off>.**)

⇒ Place the mouse pointer on the lower right sizing button of the container and, while holding down the left mouse button, drag it to a size of 250 x 250 pixels. The respective size is indicated in pixel in the status bar.

IwPushButton

⇒ Click the **<IwPushButton>** button of the **GRIT Objects** toolbar. A pushbutton is a command button which can be marked with text or a bitmap.

⇒ Now, use the **Drag&Drop** method to position the new pushbutton in the application window (while holding down the left mouse button), at any point to the right of the container.

Your application window will now look like this:

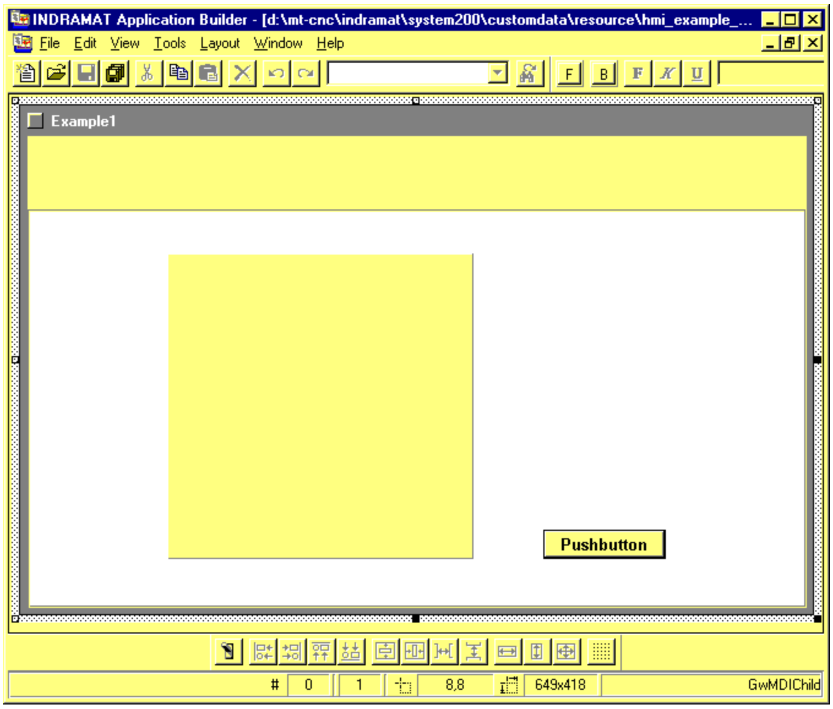

INDRAMAT Application Builder - [d:\mt-cnc\indramat\system200\customdata\resource\hmi_example_...
File Edit View Tools Layout Window Help
F B F K U
Example1
Pushbutton
# 0 1 8,8 649x418 GwMDIChild

Bild45.bmp

Fig. 2-27: Inserting objects

Aligning objects

The objects of a user interface can be aligned in many ways.

Depending on how you wish to align the objects, you must select at least two or three objects. One of the objects always forms the basis of the alignment. The other objects are aligned in relation to that particular object.

As is appropriate, the layout mechanisms only take effect for objects with a joint parent object.

In the example, it is intended to align the container and the pushbutton on the same horizontal level and to center them. If you wish to align objects on the same level, at least two objects must be selected.

⇒ Press and hold **<Ctrl>** and click first Pushbutton and then Container.

The previously selected object – here the container – is represented with the sizing buttons filled in. The container is the reference object for aligning the objects selected.

⇒ You can change the reference object subsequently by holding down **<Ctrl>** while selecting the particular object once again.

⇒ **Alternative:** Select the objects by marking a rectangle around the object using the mouse. Determine the object serving as reference for the alignment by simultaneously pressing **<Ctrl>** and clicking the mouse.

Align at Bottom

⇒ Click the **<Align at Bottom>** button of the **Layout** toolbar, or select the **Align Controls** and **Bottom** items from the **Layout** menu.

The pushbutton aligns with the bottom edge of the container.

Both objects are still selected. It is now intended to center them in the window. Centering relates to a single object only. If several objects are selected, they are treated as groups.

Center Horizontally

⇒ Click the **<Center Horizontally>** button of the **Layout** toolbar, or select the **Center in Dialog zentrieren** and **Horizontal** items from the **Layout** menu.

The layout of the user interface is completed now. Store the “HMI_Example.cgw” file using the “Save” item in the “File” menu. The resource file to be stored must be highlighted in the Navigator (because severel resources may be open).

DESKTOP.exe

Exercise test:

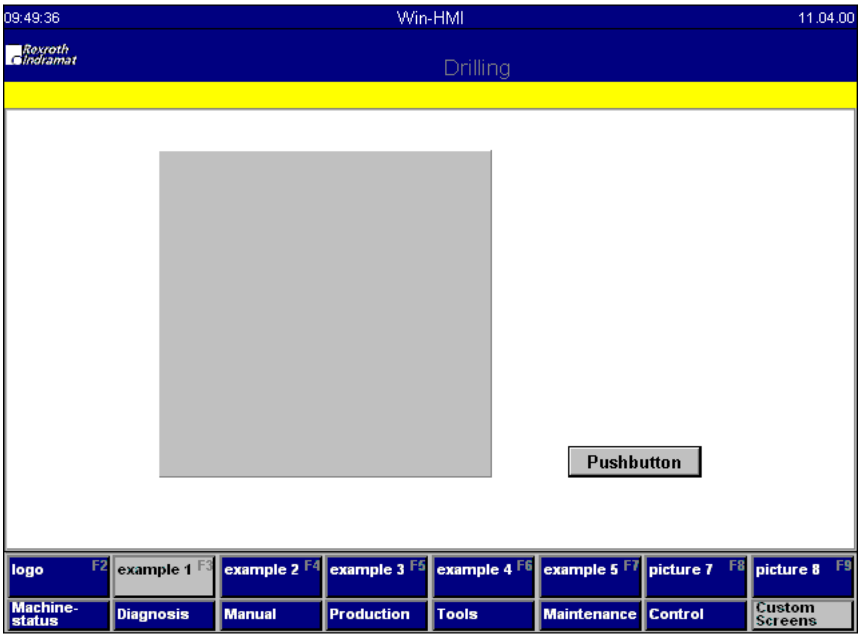


Bild46.bmp

Fig. 2-28: Custom-designed “HMI_Example.cgw” resource file

The exercise is now completed.

Note Rename the files with their original names again for the following examples:

Current name	New name
HMI_Example.cgw	HMI_Example_ueb.cgw
HMI_Example_org.cgw	HMI_Example.cgw

Fig. 2-29: Renaming HMI_Example.cgw

3 Easy Process Properties

3.1 General Information

The **Easy Properties** dialog is provided for entry of frequently used attributes of specific methods. It is a reduced version of the **Universal Properties** dialog in that there are less tabs.

In the **Easy Properties** dialog, process connections are is achieved by means of the **Easy Process Properties** dialog.

Operation of the **Easy Process Properties** dialog is described in the sections below.

3.2 Calling up the Easy Process Properties Dialog

Requirements: A new cgw file is created in the Application Builder.

Creating a new resource file:

⇒ Select **File/New.**
The list for selection of the file type appears.

⇒ Select **Resource File** and apply it with <OK>.
A tree structure is displayed in the Navigator.

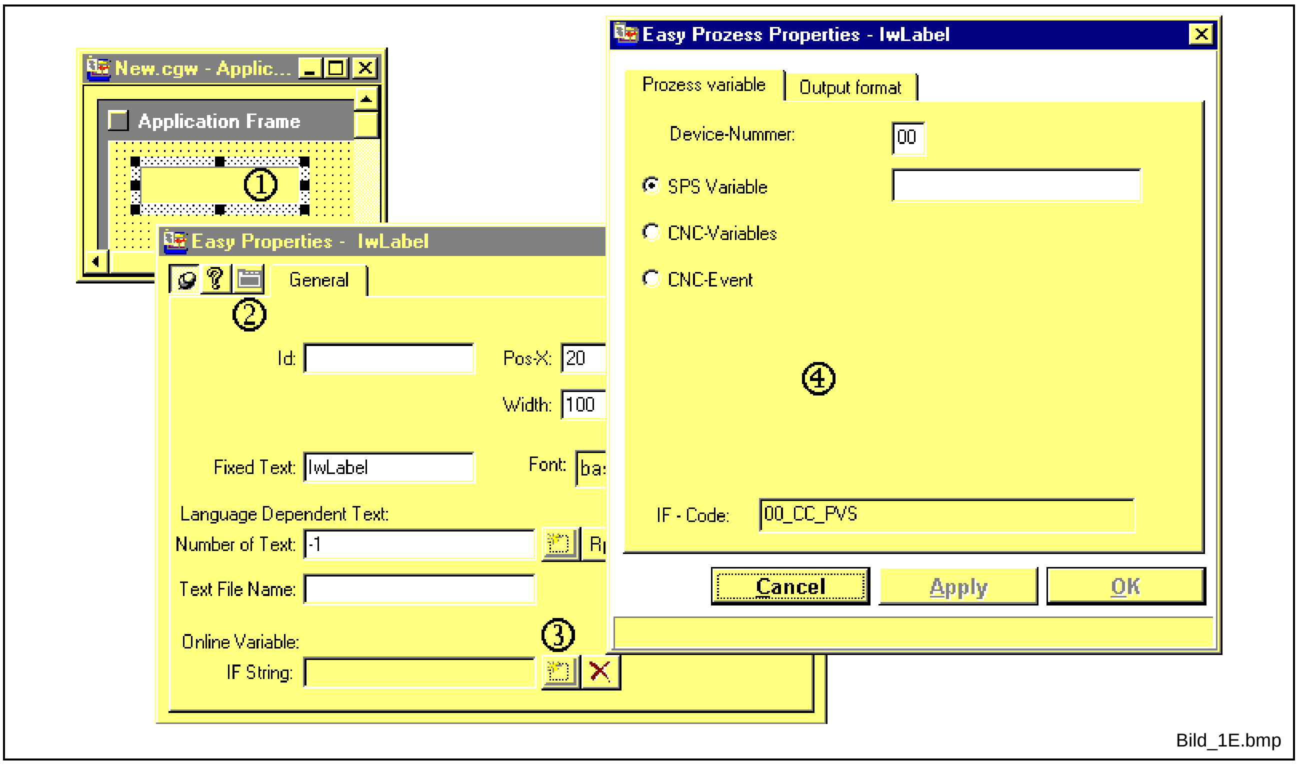
⇒ Select **Application Window** using the left mouse button and then press the right mouse button to open the **Popup menu.**

⇒ Click **Insert Application Window.**
An **Application Window** is inserted in the tree.

⇒ Double-click the new **Application Window** item.
The application window is opened in the working area.

⇒ Activate **View\GRIT Objects** (✓) so that the bar will be displayed.

⇒ From this object bar, drag **IwLabel** into the working area.
The **IwLabel** object is placed in the working area, see (1) in Fig. 3-1:



(1): IwLabel in the application window
(2): Switching from Easy to Universal Properties and vice versa
(3): Button for calling up the process connection
(4): Easy Process Properties dialog

Fig. 3-1: Menu call of the Easy Process Properties dialog (for IwLabel)

Operation

The **Easy Process Properties** dialog is called up as follows (by the example of **IwLabel**), see Fig. 3-1:

⇒ Double-click **IwLabel** (1).
The Properties dialog appears.

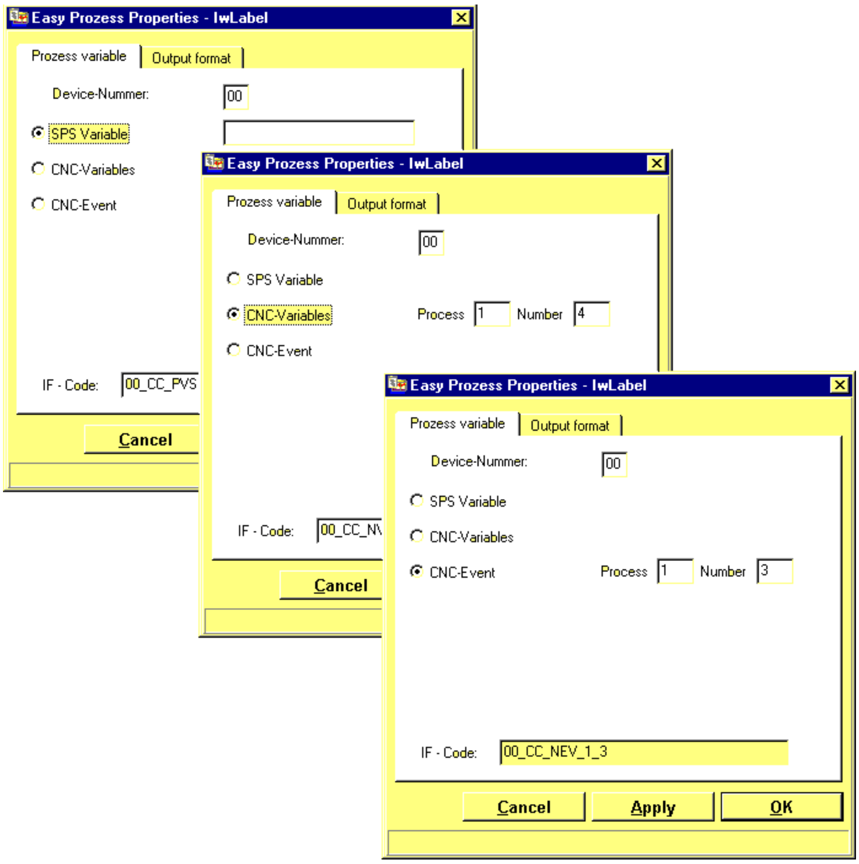
⇒ Switch from the Easy to the Universal Properties dialog and vice versa (2).
Fig. 3-1 shows the Easy Properties dialog.

⇒ Click <Open> of the process connection (3).
(4) shows the Easy Process Properties dialog.

3.3 Defining Variable Names (for Interface String)

SPS, CNC, CNC Event Variables

Note: In the Easy Properties dialog, only **one** variable can be defined per object.



Device-Nummer: 00
● SPS Variable
○ CNC-Variables
○ CNC-Event
IF - Code: 00_CC_PVS

Device-Nummer: 00
○ SPS Variable
● CNC-Variables Process 1 Number 4
○ CNC-Event
IF - Code: 00_CC_NV

Device-Nummer: 00
○ SPS Variable
○ CNC-Variables
● CNC-Event Process 1 Number 3
IF - Code: 00_CC_NEV_1_3

Bild_2E.bmp

(1): Defining the SPS Variable
(2): Defining the CNC Variable
(3): Defining the CNC Event variable
(4): IF-Code Edit button

Fig. 3-2: Selection of the variable type and entry of the variable name

Description The text entered in the Edit button is automatically added to the IF-Code. If you switch from one Radio button to the other, the entry masks appropriate for the required entries are shown.

Operation

⇒ Select the variable using the Radio button (1), (2), or (3).
The appropriate entry mask appears in the Edit button (4).

⇒ Enter the *Name* or *Process_Name* in the Edit button (1), (2), or (3).
The entry is displayed in the IF-Code Edit button (4).

Variable	IF-Code	Example
SPS Variable	*Device*_**CC_PVS**_*Name*	*Name* = Test1
CNC Variable	*Device*_**CC_NVS**_*Prozess_Nummer*	
CNC Event variable	*Device*_**CC_NEV**_*Prozess_Nummer*	

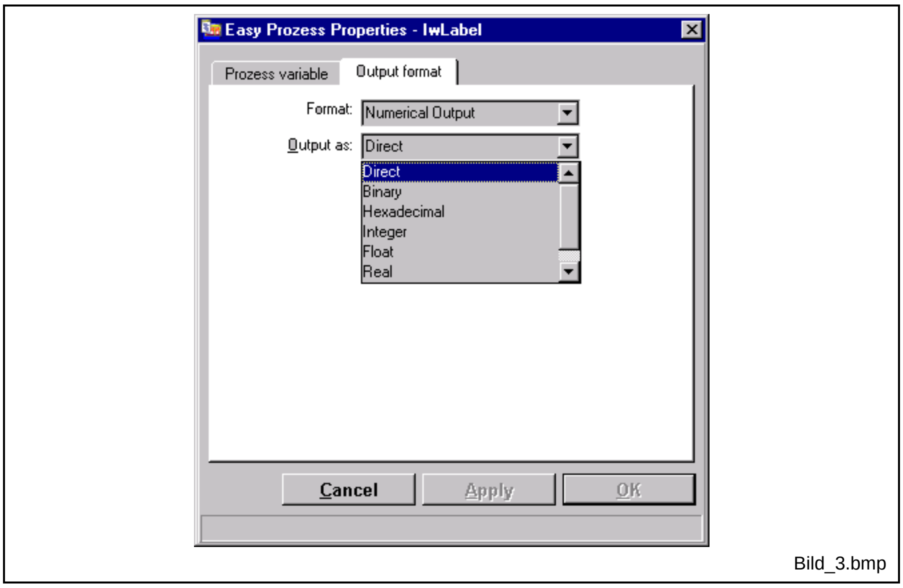
Fig. 3-3: Automatically provided IF-Code

⇒ Click **<Apply>** to apply the IF-Code.

⇒ Press **<OK>** to apply the current entry and to exit the dialog.

3.4 Output Format / Interpreter Function Call

Overview



Easy Prozess Properties - IwLabel

Format: Numerical Output

Output as: Direct

Fig. 3-4: Selection of the output format

Output format

The output format is the actual action which is triggered by the process variable. There are three different formats:

- **Numerical Output**
  All numerical output formats convert a value, which may be a number (Integer, Long, etc.) or a string, into a different output format (Hexadecimal, Binary, etc.), or they output the value directly to the object.
- **Text / Color**
  Here, instead of a control value, a text is output depending on the control value and the color of background and foreground is set.
- **Bitmap**
  Here, bitmaps are displayed in relation to process variables.

Selection of types

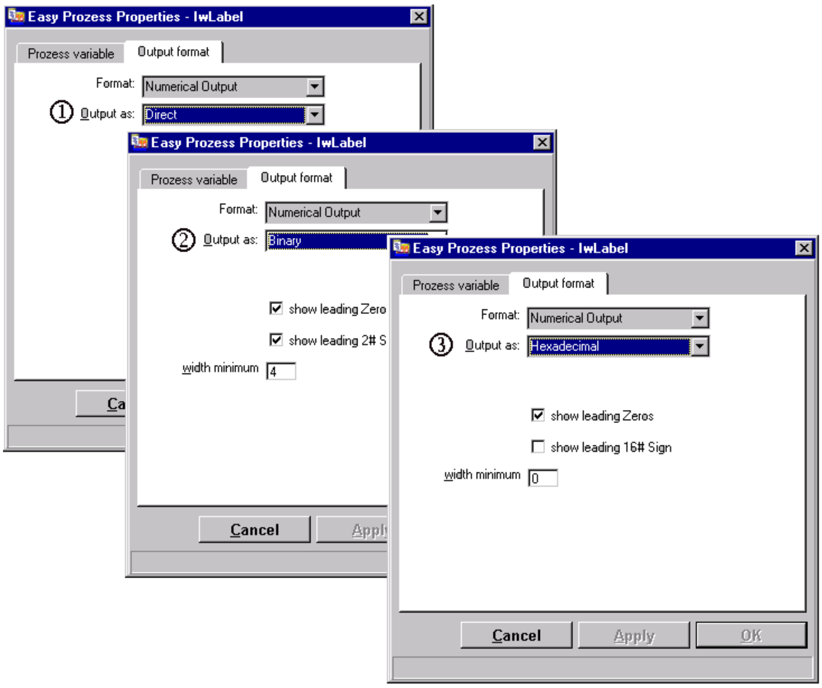
The following types of output formats are allowed:

Numerical Output	Text / Color	Bitmap
Direct Binary Hexadecimal Integer Float Real Time	Type Boole Type Value Type Range	Type Boole Type Value Type Range

Fig. 3-5: Various types of output formats

Numerical Output

The numerical output can be made as a direct number, a binary number, a hexadecimal number, an integer number, a number with floating point, a real number, or it can be represented as a time parameter.



Format: Numerical Output
① Output as: Direct

Format: Numerical Output
② Output as: Binary
☑ show leading Zero
☑ show leading 2# S
width minimum 4

Format: Numerical Output
③ Output as: Hexadecimal
☑ show leading Zeros
□ show leading 16# Sign
width minimum 0

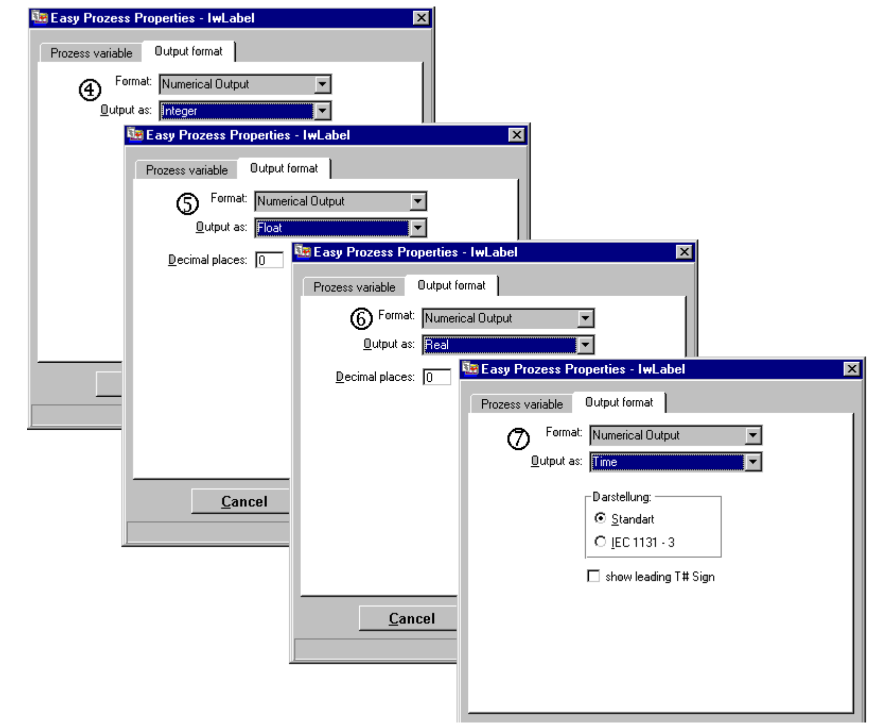
Bild_4E.bmp

(1): Direct numerical output
(2): Binary numerical output
(3): Hexadecimal numerical output

Fig. 3-6: Numerical Output: Direct, Binary, Hexadecimal

Description

- **Direct**
  The contents of the process variables are directly displayed on the object (**set_text** interpreter method).
- **Binary**
  Numerical values are converted into the binary format and are output on the current object using the **set_formated_text** method.
- **Hexadecimal**
  Numerical values are converted into the hexadecimal format and are output on the current object using the **set_formated_text** method.



④ Format: Numerical Output
Output as: Integer

⑤ Format: Numerical Output
Output as: Float
Decimal places: 0

⑥ Format: Numerical Output
Output as: Real
Decimal places: 0

⑦ Format: Numerical Output
Output as: Time
Darstellung:
● Standart
○ IEC 1131 - 3
□ show leading T# Sign

Bild_5E.bmp

(4): Integer numerical output
(5): Float numerical output
(6): Real numerical output
(7): Time numerical output

Fig. 3-7: Numerical Output: Integer, Float, Real, Time

Description

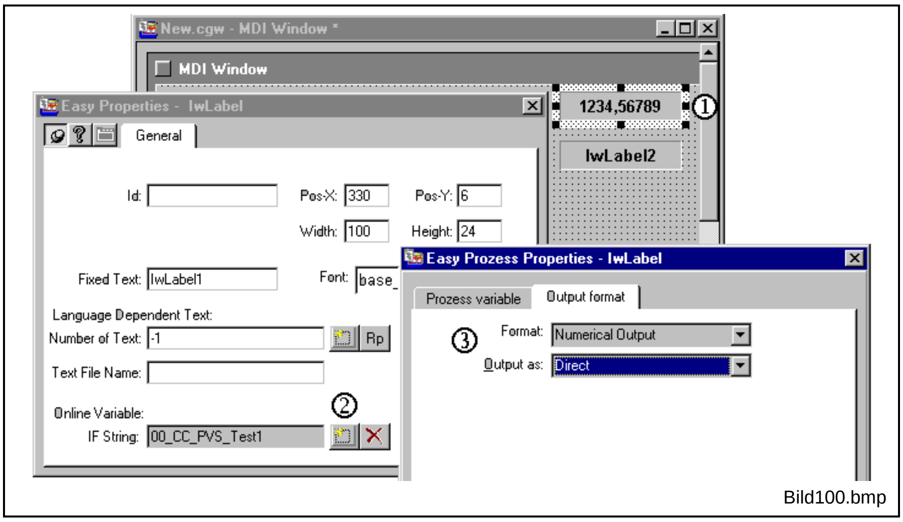
- **Integer**
  The values of the process variables are displayed as integer numbers.
- **Float**
  The values of the process variables are displayed as the fixed-point part with exponent, i.e. the number 0,123456123 is represented as 1,23456123e-001 in the object.
- **Real**
  The values of the process variables are displayed as a real number. Example: 34,55584 is represented as 34,55 when there are 2 digits after the point.
- **Time**

Language Dependent function

In the Easy Process Properties dialog, language-dependent means that, for instance, the set_token_text() method is called instead of the set_text() method. Then the entry from the additional edit fields are added to this method. If “Language Dependent” is selected, only numbers should be entered in these fields.

Example of operation

It is intended to display the **Test1** SPS variable in the **IwLabel1** label. The content of the SPS variable is a numerical value, which is to be displayed as a direct number. Fig. 3-8 illustrates the sequence of the settings required:



Id:
Pos-X: 330
Pos-Y: 6
Width: 100
Height: 24
Fixed Text: IwLabel1
Font: base_
Language Dependent Text:
Number of Text: -1
Text File Name:
Online Variable:
IF String: 00_CC_PVS_Test1

Format: Numerical Output
Output as: Direct

Bild100.bmp

(1): IwLabel1
(2): Calling up Easy Process Properties
(3): Output format

Fig. 3-8: Example of Numerical Output

⇒ Double-click **IwLabel1 (1)** to open the Properties dialog.

Fig. 3-8 shows the Easy Properties dialog.

The process connection window contains the previously defined **Test1** SPS variable.

⇒ Click the **Insert/Edit** button **(2)** to open the Easy Properties dialog .

⇒ Select the **Output format** tab.

⇒ Select **Numerical Output** as format. The output is to be **Direct (3).**

1234,56789 **(1)** is displayed as a dummy in Iwlabel1. It is represented as direct numerical output.

⇒ Output as: **Binary** number

Dummy: 10011010010

⇒ Output as: **Hexadecimal** number

Dummy: 4d2

⇒ Output as: **Integer** number

1234

⇒ Output as: number with **floating point**, 3 digits after the point

1234,568

⇒ Output as: **Real** number, 3 digits after the point

1,235e + 003

⇒ Output as: **Time**

T is displayed in the IwLabel1 label.

Text / Color: Type Boole

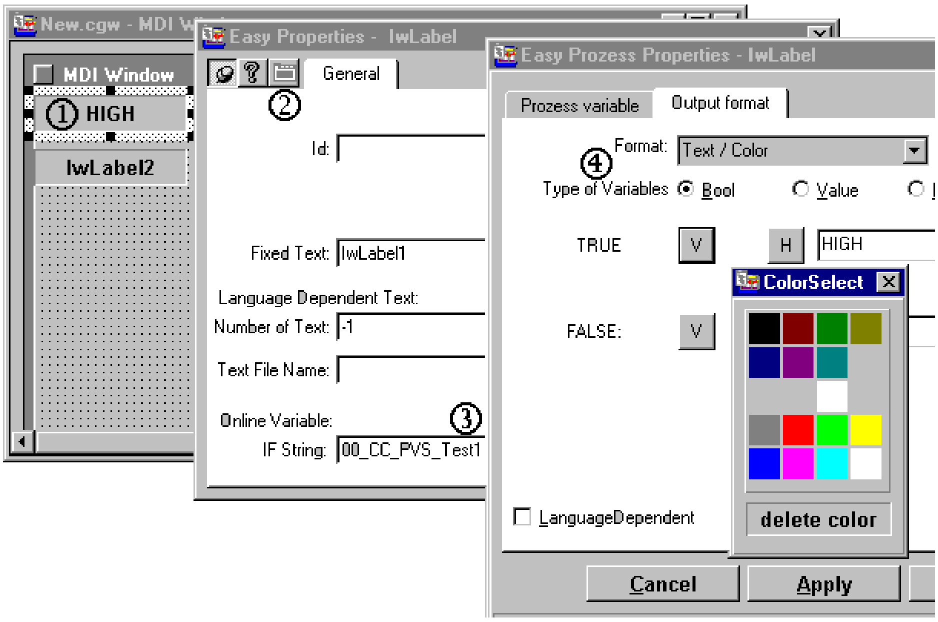
If a Boole type variable (value: low or high) is used, the text or the color of certain objects in the user interface can be changed.

Example of operation

The **IwLabel1** is activated by the Boolean SPS variable **Test1.** The following properties are defined:

Test1 variable	Label text	Foreground color	Background color
if Test1 = high:	HIGH	black	green
if Test1 = low:	LOW	white	red

Fig. 3-9: Test1 variable as Boole type



Id:

Fixed Text: IwLabel1

Language Dependent Text:
Number of Text: -1
Text File Name:

Online Variable:
IF String: 00_CC_PVS_Test1

Format: Text / Color
Type of Variables ● Bool ○ Value ○ I

TRUE V H HIGH

FALSE: V

□ LanguageDependent

Bild101.bmp

(1): IwLabel1
(2): Easy⇔Universal switching button
(3): Calling up Easy Process Properties
(4): Selecting format and variable type
(5): Entering text and color

Fig. 3-10: Example of Boole type text and color

⇒ Double-click **IwLabel1 (1)**.
The Properties dialog is opened.

⇒ Press the button **(2)** to switch from Easy to Universal Properties and vice versa.

⇒ Press the button **(3)** to open the Easy Process Properties dialog.
The Test1 SPS variable has already been defined as the process connection beforehand.

⇒ Set the **Text / Color** format and the **Boole** type of variables on the Output format tab **(4).**
The entry objects for text and color are displayed **(5).**

⇒ Entry of properties if Test1 = true:
Text => HIGH; select V = "black" and H = "green" from the color palette.
Iwlabel1 displays the properties selected **(1).**

Text / color: Type Value

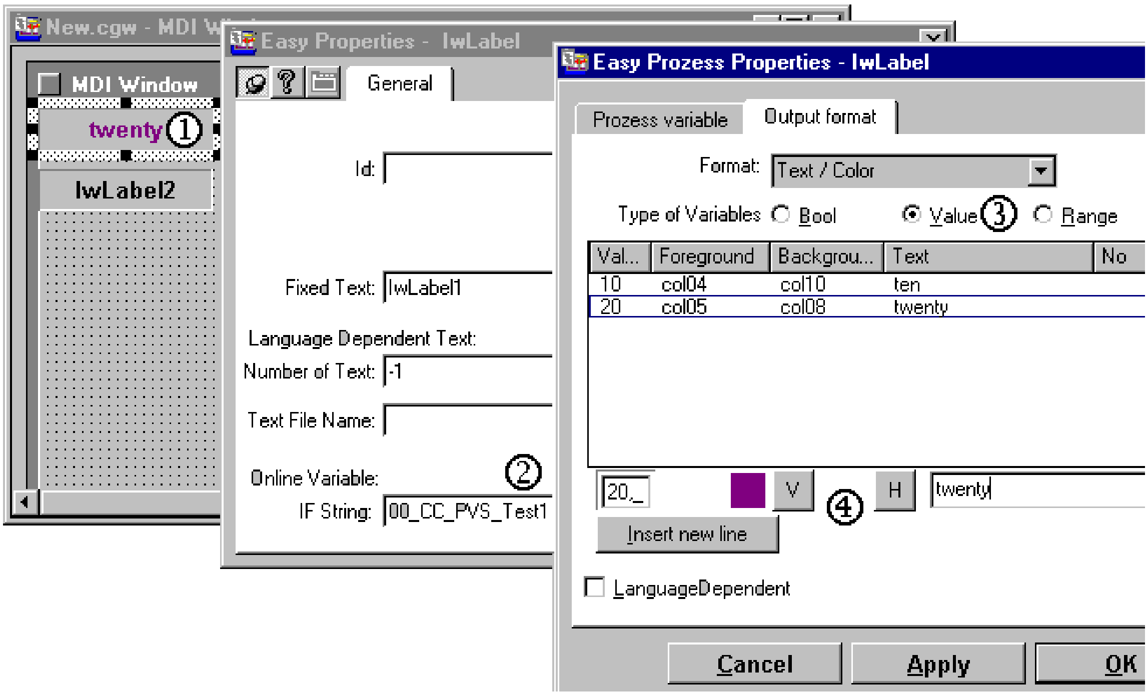
If a Value type variable is used, the text or the color of various objects in the user interface can be changed.

Example of operation

The **IwLabel1** is activated by the Boolean SPS variable **Test1.** The following properties are defined:

Test1 variable	Label text	Foreground color	Background color
if Test1 = 10:	Ten	white	red
if Test1 = 20:	Twenty	black	green

Fig. 3-11: Test1 variable as Value type



Id:

Fixed Text: IwLabel1

Language Dependent Text:

Number of Text: -1

Text File Name:

Online Variable:

IF String: 00_CC_PVS_Test1

Format: Text / Color

Type of Variables ○ Bool ● Value ○ Range

Val...	Foreground	Backgrou...	Text	No
10	col04	col10	ten	
20	col05	col08	twenty	

20._ twenty

□ LanguageDependent

Bild102.bmp

(1): IwLabel1
(2): Calling up Easy Process Properties
(3): Value type of variable
(4): Selecting the properties

Fig. 3-12: Example of the Value type text and color

⇒ Double-click **IwLabel1 (1).**
The Properties dialog is opened.

⇒ Press the button **(2)** to open the Easy Process Properties dialog.
The Test1 SPS variable has already been defined as the process connection beforehand.

⇒ Set the **Text / Color** format and the **Value** type of variables on the Output format tab **(3).**
The entry objects for text and color are displayed **(4).**

⇒ Activate <Insert new line>.
The first line in the table is selected.

⇒ Select a foreground color **(V)**, a background color **(H)**, and a text for each value.

The colors and the text are displayed in IwLabel1 **(1)**.

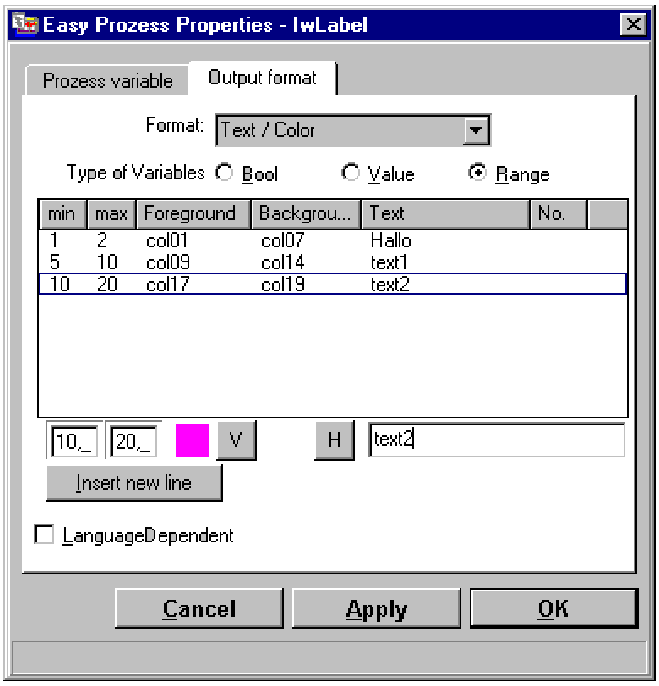
Text / Color: Type Range

If a Range type variable is used, the text or the color of certain objects in the user interface can be changed.

Example of operation

See Text / color: Type Value , however Value => Range

In case of the Range type, a **min...max** range is entered in the place of the value in case of the Value type.



Easy Prozess Properties - IwLabel

Prozess variable | Output format

Format: Text / Color

Type of Variables ○ Bool ○ Value ● Range

min	max	Foreground	Backgrou...	Text	No.
1	2	col01	col07	Hallo	
5	10	col09	col14	text1	
10	20	col17	col19	text2	

10._ 20._ 🟪 V H text2

TextFarbe03.bmp

Fig. 3-13: Example of the Range type text and color

Bitmap: Type Boole, Type Value, Type Range

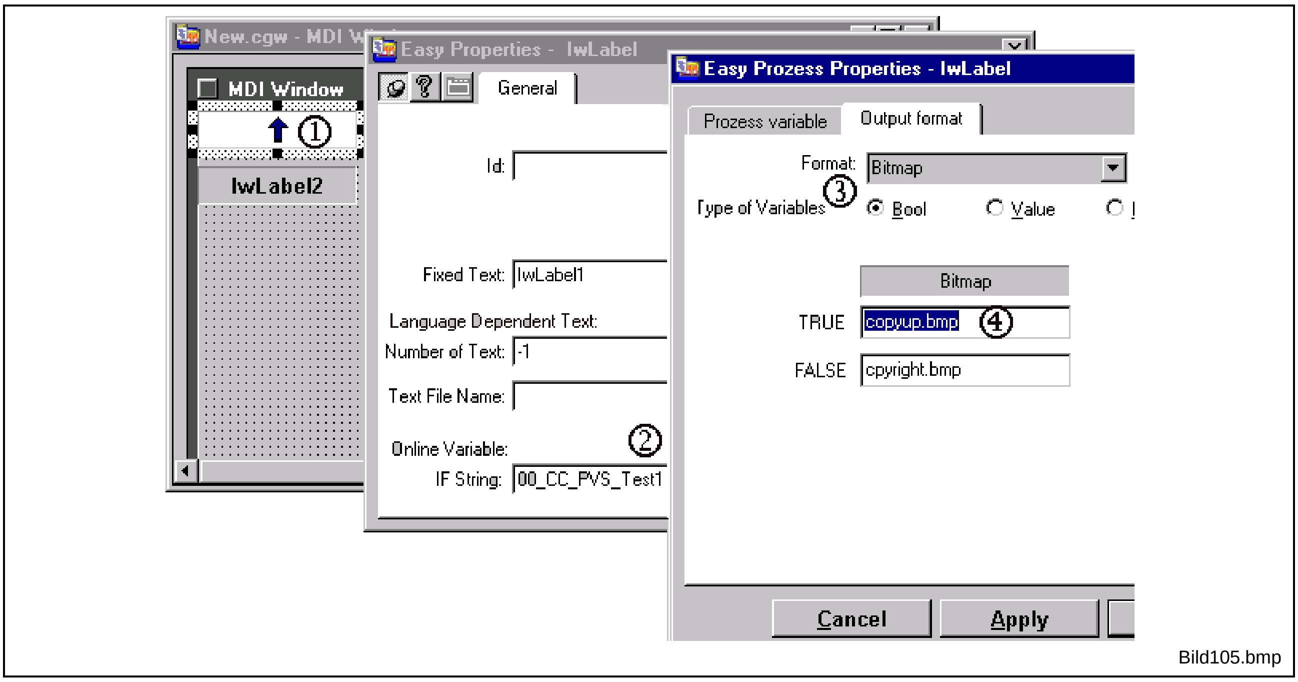
If Boole type, Value type, or Range type variables are used, bitmaps instead of texts can be superimposed in certain user interface objects.

Example of operation

The **IwLabel1** object is activated by the Boolean SPS variable **Test1:**

Test1 variable	Bitmap	Folder
Test1 = TRUE:	copyup.bmp	[Drive]:\Mt-CNC\INDRAMAT \System200\BasicData\Bitmap
Test1 = FALSE:	cpyright.bmp	

Fig. 3-14: Files for bitmaps



Id:
Fixed Text: IwLabel1
Language Dependent Text:
Number of Text: -1
Text File Name:
Online Variable:
IF String: 00_CC_PVS_Test1

Format: Bitmap
Type of Variables: ● Bool ○ Value ○ I
TRUE copyup.bmp
FALSE cpyright.bmp

Bild105.bmp

(1): IwLabel1
(2): Calling up Easy Process Properties
(3): Boole type variable
(4): File name

Fig. 3-15: Example of Boole type bitmap

⇒ Double-click **IwLabel1 (1)**.
The Properties dialog is opened.

⇒ Press the Edit Process Connection button **(2)** to open the Easy Process Properties dialog.
The dialog is opened. Test1 has already been defined as the SPS variable beforehand.

⇒ Select **Bitmap** as format and **Boole** as type of variables **(3).**

⇒ Enter the file name *copyup.bmp* **(4)**, and press <Enter>.
The bitmap is displayed in the **IwLabel1** label for control purposes **(1)**.

⇒ In case of FALSE, enter the file name *cpyright.bmp*, and press <Enter>.
The bitmap is displayed in the **IwLabel1** label for control purposes.

⇒ Press <Apply> to store your entries.

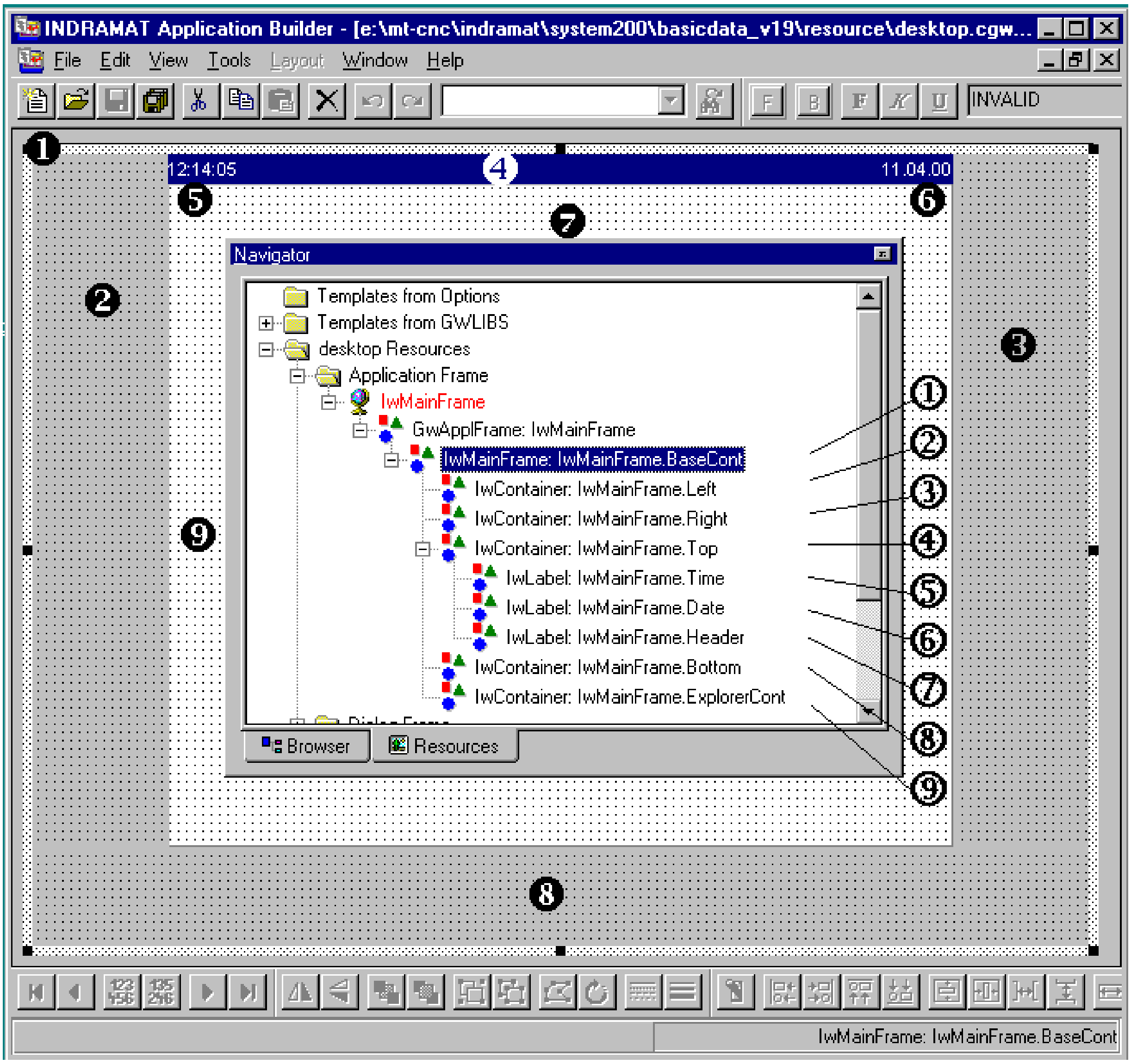
4 Structure of the HMI Desktop

4.1 Desktop.cgw

This section is an overview of where and how the resource files (CGW files) you have created using the Application Builder are incorporated.

The HMI user interface is defined in the “Desktop.cgw” program. The “IwMainFrame” application window contains several containers, which form the structure of the user interface. The figure below shows the IDs assigned to the containers.

“IwMainFrame” application window



INDRAMAT Application Builder - [e:\mt-cnc\indramat\system200\basicdata_v19\resource\desktop.cgw...

INVALID

12:14:05 11.04.00

Navigator

- Templates from Options
- Templates from GWLIBS
- desktop Resources
  - Application Frame
    - IwMainFrame
      - GwApplFrame: IwMainFrame
        - IwMainFrame: IwMainFrame.BaseCont
          - IwContainer: IwMainFrame.Left
          - IwContainer: IwMainFrame.Right
          - IwContainer: IwMainFrame.Top
            - IwLabel: IwMainFrame.Time
            - IwLabel: IwMainFrame.Date
            - IwLabel: IwMainFrame.Header
          - IwContainer: IwMainFrame.Bottom
          - IwContainer: IwMainFrame.ExplorerCont

IwMainFrame: IwMainFrame.BaseCont

Bild90.bmp

Fig. 4-1: “IwMainFrame” application window

Exercise: Starting “IwMainFrame” with “wbtest.exe”

The environment of the Application Builder includes the “wbtest.exe” test program, which can be used to test currently running application windows. The functionality deposited in the callbacks with the Grit-internal 4GL program is available immediately.

Call:

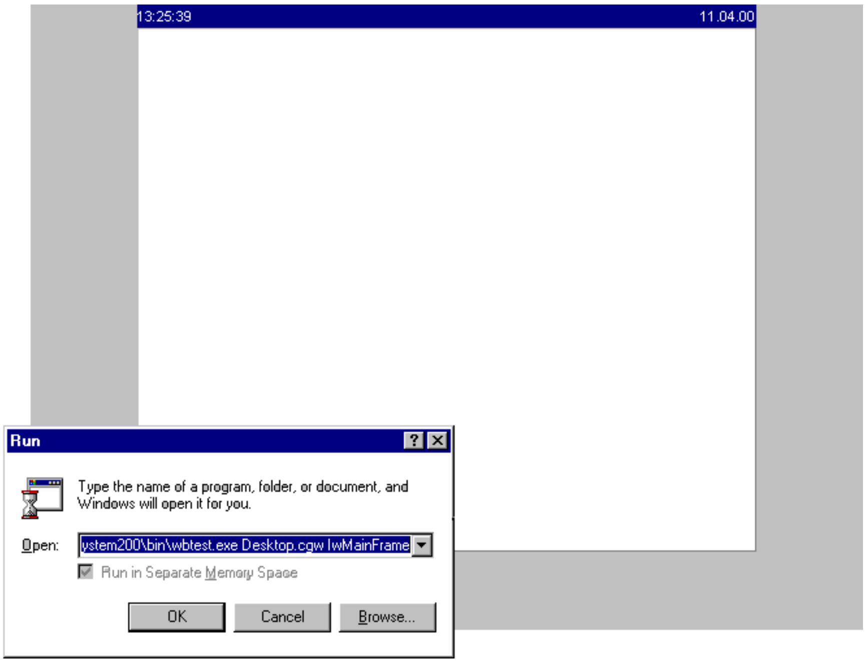
```
wbtest.exe [resource file.cgw [Starting mask]]
```

In the place of the **resource file,** enter the name of the resource file to be loaded with the filename extension “.cgw”.

In the place of the **starting mask,** enter the name of the application window to be processed upon start of the program. If the starting mask does not exist, it is not possible to start the mask system.

Example:

```
D:\mt-cnc\indramat\system200\bin\wbtest.exe
     desktop.cgw IwMainFrame
```



13:25:39

11.04.00

Run

Type the name of a program, folder, or document, and Windows will open it for you.

Open: ystem200\bin\wbtest.exe Desktop.cgw IwMainFrame

☑ Run in Separate Memory Space

Bild93.bmp

Fig. 4-2: “IwMainFrame” started with “wbtest.exe”

Enter the file call in the “Run” window of the Windows NT start menu.

Dialog Windows in the “desktop.cgw” File

The “desktop.cgw” file possesses four dialog windows: three for the various key assignments and one “IwExplorer” dialog window. The dialog windows are represented in the Application Builder in sections:

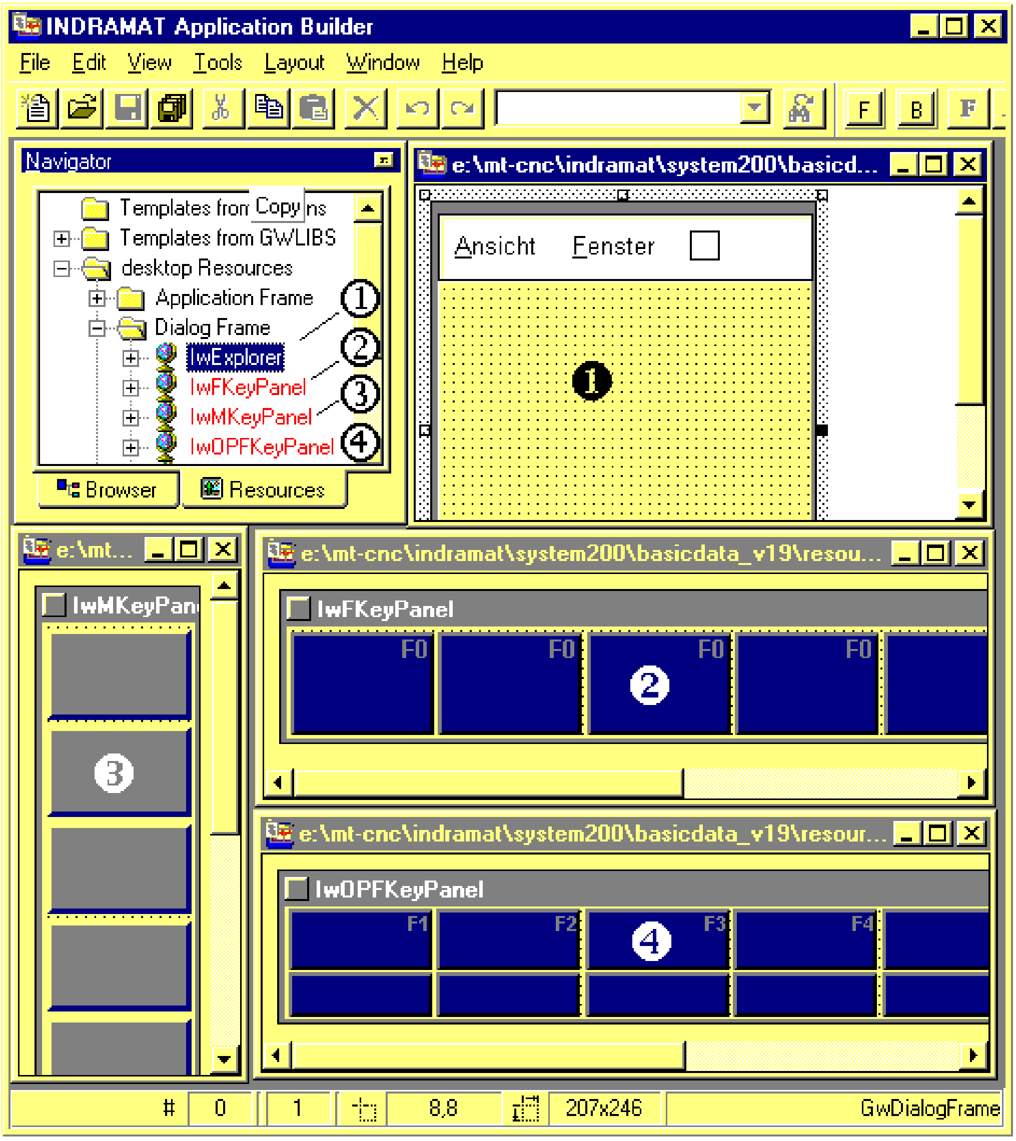

INDRAMAT Application Builder
File Edit View Tools Layout Window Help
Navigator
Templates from Copy ns
Templates from GWLIBS
desktop Resources
Application Frame
Dialog Frame
IwExplorer
IwFKeyPanel
IwMKeyPanel
IwOPFKeyPanel
1
2
3
4
Browser
Resources
e:\mt-cnc\indramat\system200\basicd...
Ansicht Fenster
1
e:\mt...
IwMKeyPan
3
e:\mt-cnc\indramat\system200\basicdata_v19\resou...
IwFKeyPanel
F0
F0
F0
F0
2
e:\mt-cnc\indramat\system200\basicdata_v19\resour...
IwOPFKeyPanel
F1
F2
F3
F4
4
# 0 1 8,8 207x246 GwDialogFrame

Bild91.bmp

Fig. 4-3: Dialog window in “desktop.cgw”

“IwExplorer” Dialog Window

The figure below shows the structure of the “IwExplorer” dialog window. Among other containers, this dialog window contains the “IwExplorer.MDICont” MDIContainer. The point in the ID does not have any functional meaning, but is only intended to point out that the MDIConainer is assigned to the “IwExplorer” dialog window.

In the examples, this container has been used for accommodation of the special pictures. The figure below illustrates the relations between the various objects:

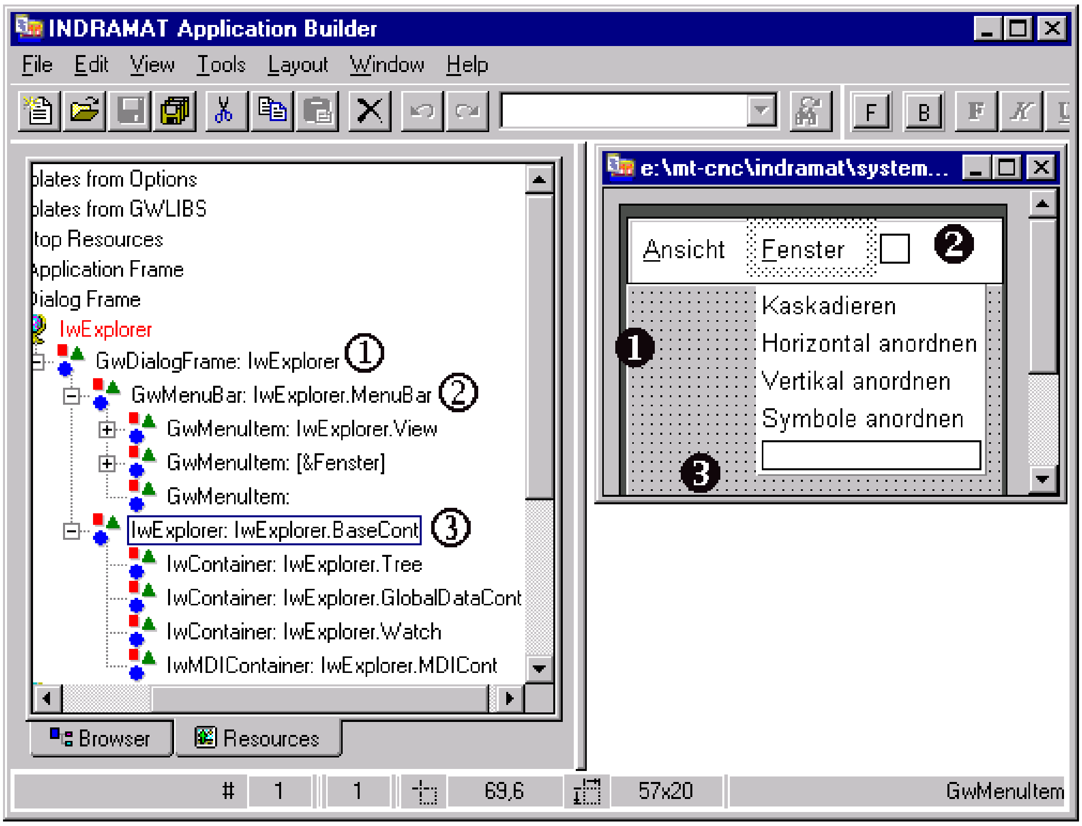

INDRAMAT Application Builder
File Edit View Tools Layout Window Help
F B
plates from Options
plates from GWLIBS
top Resources
Application Frame
Dialog Frame
IwExplorer
GwDialogFrame: IwExplorer ①
GwMenuBar: IwExplorer.MenuBar ②
GwMenuItem: IwExplorer.View
GwMenuItem: [&Fenster]
GwMenuItem:
IwExplorer: IwExplorer.BaseCont ③
IwContainer: IwExplorer.Tree
IwContainer: IwExplorer.GlobalDataCont
IwContainer: IwExplorer.Watch
IwMDIContainer: IwExplorer.MDICont
Browser
Resources
# 1 1 69,6 57x20 GwMenuItem
e:\mt-cnc\indramat\system...
Ansicht
Fenster
❷
❶
Kaskadieren
Horizontal anordnen
Vertikal anordnen
Symbole anordnen
❸

Bild92.bmp

Fig. 4-4: “IwExplorer” dialog window

The “IwExplorer” dialog window with the GwDialogFrame “IwExplorer” (1) possesses the “IwExplorer.MenuBar” menu bar (2) and the “IwExplorer.BaseCont” IwExplorer (3).

The “IwExplorer” object consists of three IwContainers and one IwMDIContainer with the ID “IwExplorer.MDICont”. The latter is used for accommodation of the special pictures.

4.2 Parent Windows and Child Windows

The properties of windows are determined not only by their class, but also by the relations of the windows among each other. These relations affect the behavior and the quality of window instances to a large extent. One of the possible relations among the windows is the parent-child relation. A window can possess one or several child windows. A child window can possess only one parent window.

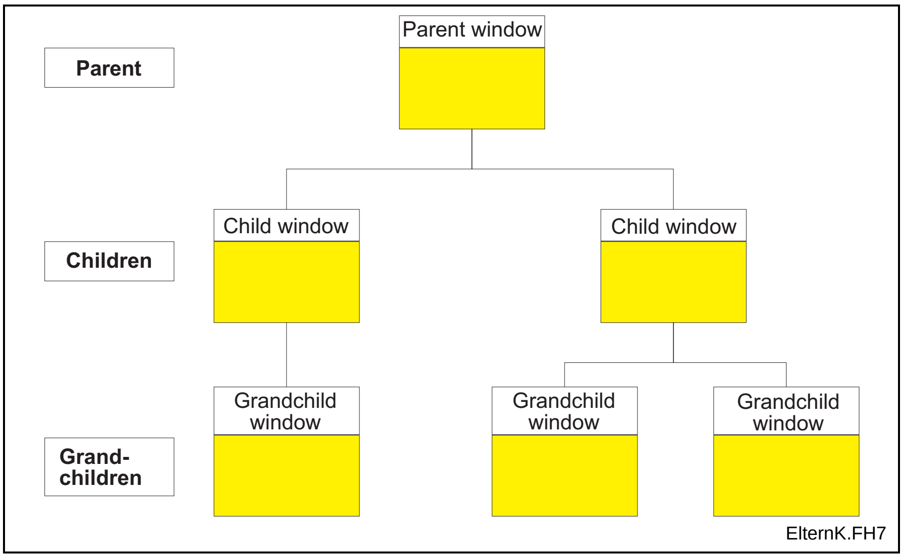


Fig. 4-5: Parent-child relation among windows

The relations among the various windows are defined in the HMI Desktop. Integration of the special pictures is defined.

4.3 MDI (Multi Document Interface)

Word processing programs are a good example of the principle of operation of MDI applications. While the word processing program itself appears in a main window (MDI Frame window), each document to be processed is displayed in its own window (MDI child window). This allows parallel processing of documents. In addition, the simultaneous view of several documents facilitates the rapid data exchange by means of the clipboard or the drag-and-drop function (see MDI window).

Three different window types form part of MDI applications:

MDI Frame window: This window is the main window (frame) of the application. It is the parent window of the MDI Client window.

MDI Client window: This window is the “invisible” instance between the MDI Frame window and the MDI child windows. The exchange of data between the main window and the child windows is exclusively processed via the MDI Client window.

MDI child windows: These windows are children of the MDI Client window and, thus, the grandchildren of the MDI Frame window. They contain the documents to be managed.

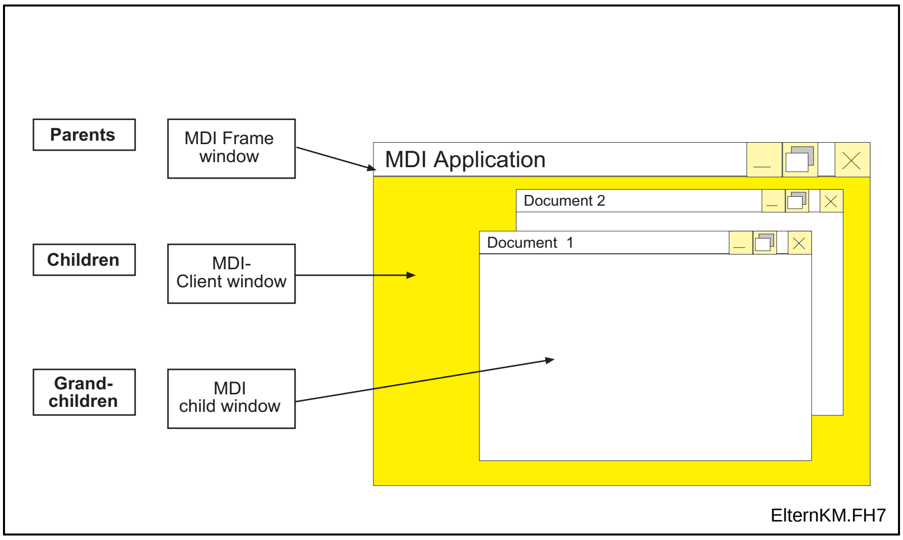


Fig. 4-6: Window types participating in the MDI

The “IwExplorer.MDICont” IwMDIContainer is used for the special pictures of the HMI user interface. It accommodates the MDI child windows to be created by the user.

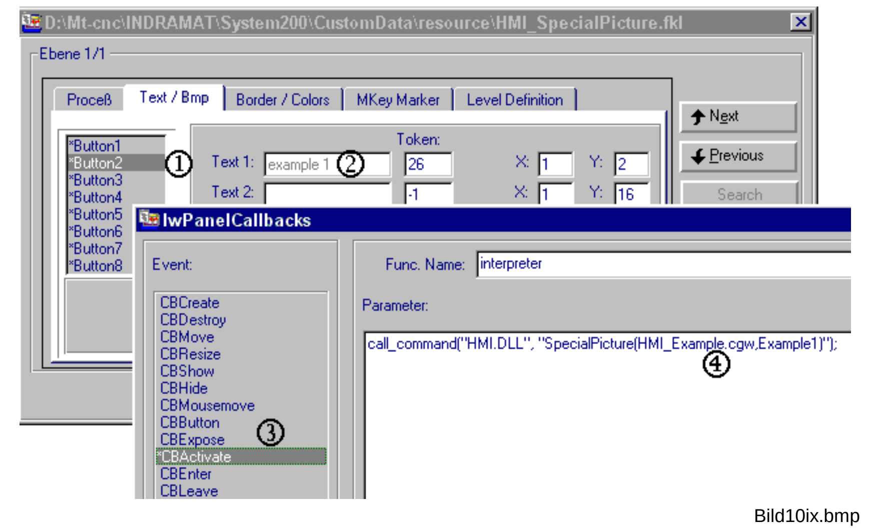
5 Examples with Process Connection

5.1 Example1: SPS and NC Variable

Loading HMI_Example.cgw

Taking a name from the fkl file

You can take the names of the cgw files from the **Specialpicture.fkl** file. These names contain additional examples:



Token:

Text 1: example 1 — 26 — X: 1 — Y: 2

Text 2: — -1 — X: 1 — Y: 16

Func. Name: interpreter

Parameter:

```
call_command("HMI.DLL", "SpecialPicture(HMI_Example.cgw,Example1)");
```

Bild10ix.bmp

(1): Selecting Button2
(2): Button2 is lettered with Example1
(3): Double-clicking (1) opens the CB Editor with *CBActivate
(4): CBActivate calls HMI_Example.cgw

Fig. 5-1: fkl editor showing the cgw files used

Loading and opening HMI_Example.cgw

Example 1 is contained in the HMI_Example.cgw file. The file must be filed in the CustomData folder:

⇒ Copy the file to
D:\Mt-cnc\INDRAMAT\System200\CustomData\resource\
HMI_Example.cgw

The Navigator in the Application Builder shows the following examples:

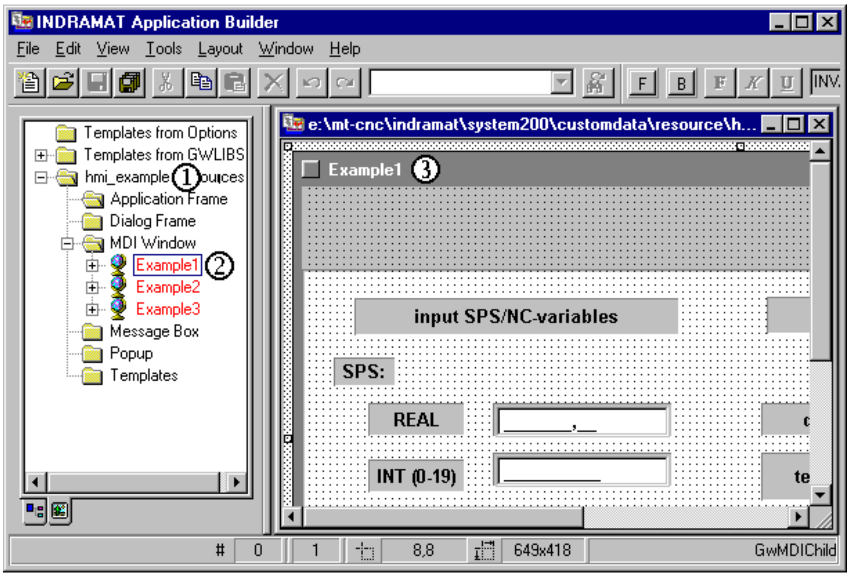


Bild11ix.bmp

(1): Opened HMI_example.cgw file
(2): MDI window Example1
(3): Opened Example1 MDI window

Fig. 5-2: HMI_Example.cgw in the Application Builder

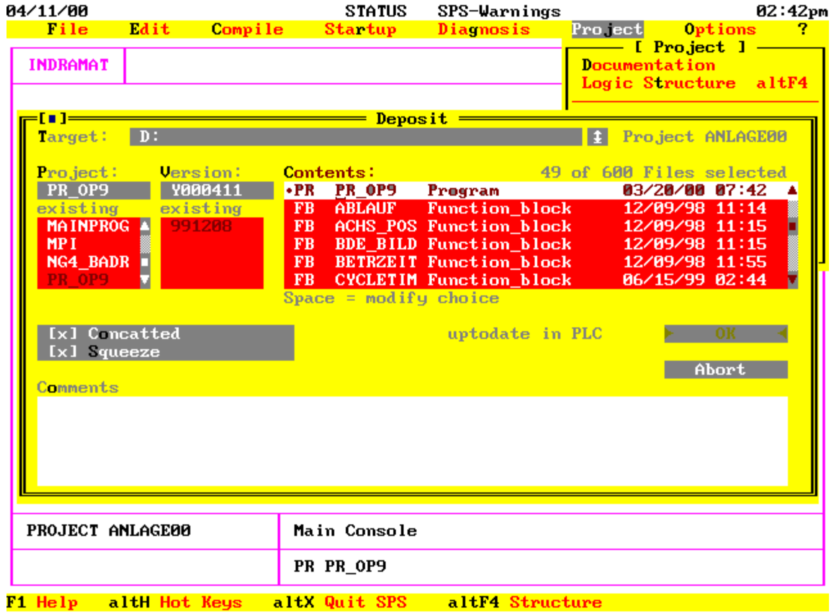
Loading the SPS Program

The variables defined in the cgw files for process connection must be known in the SPS. The **Pr_op9\991208** SPS program contains the variables required. It must be loaded into the SPS:

⇒ Copy the SPS program in the following folder:
D:\SPS_ARC\Pr_op9\991208

⇒ Open the SPS user interface.

⇒ Open the **Project\Deposit\Load** menu.
A picture similar to that below is opened:



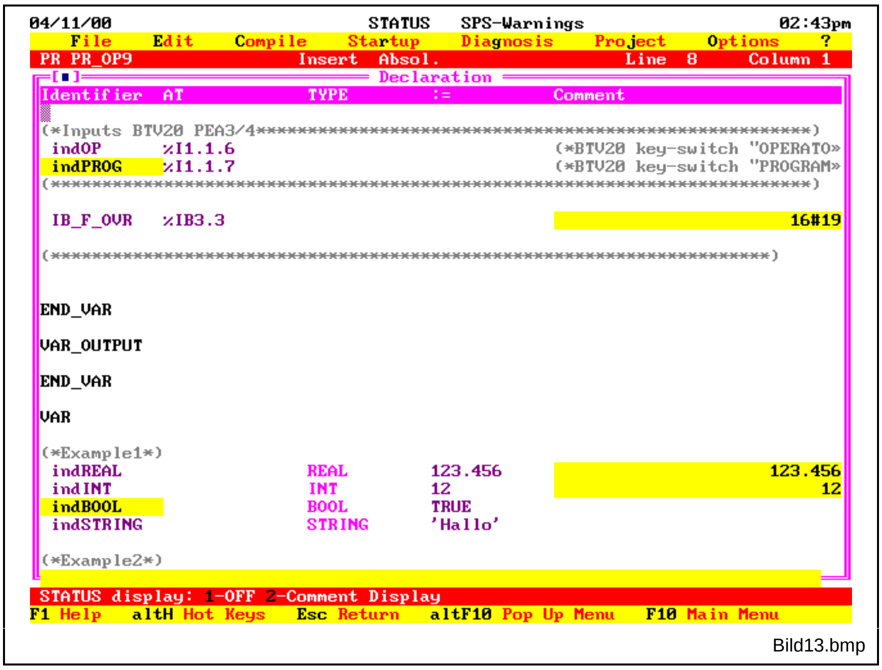
```
04/11/00                          STATUS   SPS-Warnings                    02:42pm
     File     Edit     Compile     Startup     Diagnosis     Project     Options    ?
                                                            ──── [ Project ] ────
 INDRAMAT                                                    Documentation
                                                             Logic Structure   altF4
 ═[■]════════════════════════════ Deposit ════════════════════════════════
  Target:  D:                                              ↕  Project ANLAGE00

  Project:       Version:       Contents:                      49 of 600 Files selected
   PR_OP9         Y000411       •PR  PR_OP9    Program            03/20/00 07:42
  existing       existing        FB  ABLAUF    Function_block     12/09/98 11:14
   MAINPROG       991208         FB  ACHS_POS  Function_block     12/09/98 11:15
   MPI                           FB  BDE_BILD  Function_block     12/09/98 11:15
   NG4_BADR                      FB  BETRZEIT  Function_block     12/09/98 11:55
   PR_OP9                        FB  CYCLETIM  Function_block     06/15/99 02:44
                                Space = modify choice

   [x] Concatted                              uptodate in PLC           OK
   [x] Squeeze
                                                                       Abort
  Comments


 PROJECT ANLAGE00                 Main Console

                                  PR PR_OP9
F1 Help    altH Hot Keys    altX Quit SPS    altF4 Structure
```

Bild12.bmp

Fig. 5-3: Loading the SPS program

⇒ Press <Ctrl>+<F9> to transfer the SPS program you have loaded from the user interface to the SPS controller.

The declaration in the STATUS display is as follows:



```
04/11/00                        STATUS   SPS-Warnings                    02:43pm
    File    Edit    Compile    Startup    Diagnosis    Project    Options    ?
 PR PR_OP9                 Insert  Absol.                  Line  8     Column 1
 [■]════════════════════════════════ Declaration ═══════════════════════════════
Identifier  AT                TYPE        :=          Comment
▒
(*Inputs BTV20 PEA3/4******************************************************)
 indOP      %I1.1.6                                   (*BTV20 key-switch "OPERATO»
 indPROG    %I1.1.7                                   (*BTV20 key-switch "PROGRAM»
(**************************************************************************)

 IB_F_OVR   %IB3.3                                                          16#19

(**********************************************************************)


END_VAR

VAR_OUTPUT

END_VAR

VAR

(*Example1*)
 indREAL                          REAL         123.456                    123.456
 indINT                           INT          12                              12
 indBOOL                          BOOL         TRUE
 indSTRING                        STRING       'Hallo'

(*Example2*)

 STATUS display: 1-OFF 2-Comment Display
F1 Help    altH Hot Keys    Esc Return    altF10 Pop Up Menu    F10 Main Menu
```

Bild13.bmp

Fig. 5-4: Declaration in the SPS

NC Variable

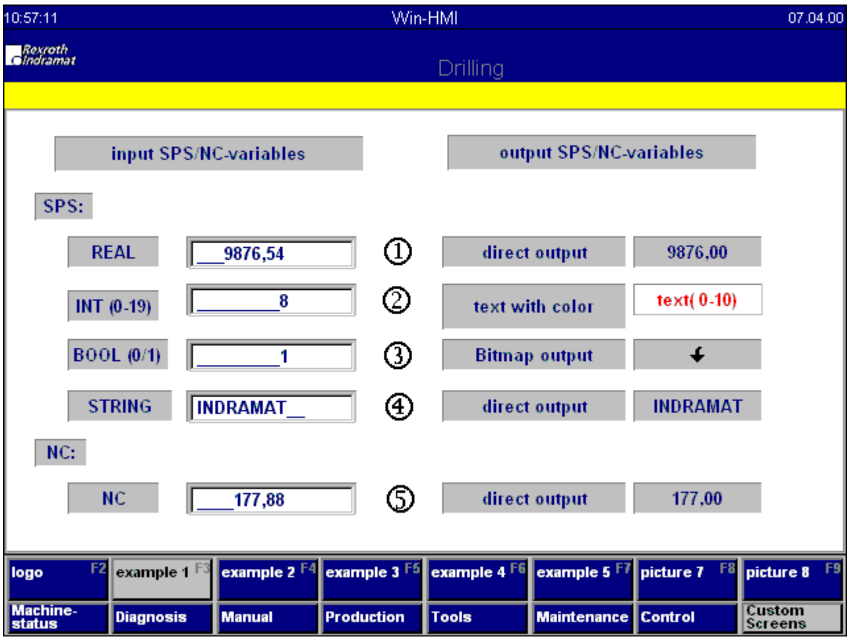
Each NC process possesses 256 NC variables. These variables must not be declared, because they are a fixed integral part of the NC controller.

The notation (see function interface) is as follows:

Syntax	Function	Example
CR_NVS_①_②	Reading	00_CR_NVS_6_10
CW_NVS_①_②	Writing	00_CW_NVS_6_10
CC_NVS_①_②	Cyclic reading	
①Process No. [0...6] ②Variable No. [0...255]		

Fig. 5-5: Syntax for NC variable

Starting WIN-HMI



10:57:11 Win-HMI 07.04.00

Drilling

input SPS/NC-variables			output SPS/NC-variables	
SPS:				
REAL	___9876,54	①	direct output	9876,00
INT (0-19)	________8	②	text with color	text(0-10)
BOOL (0/1)	________1	③	Bitmap output	↓
STRING	INDRAMAT__	④	direct output	INDRAMAT
NC:				
NC	____177,88	⑤	direct output	177,00

Bild3ix.bmp

(1):	left: entry of real number	right: direct output
(2):	left: entry of integer	right: text with color
(3):	left: entry of Boolean number	right: Bitmap output
(4):	left: entry of string	right: direct output
(5):	left: entry of real number	right: direct output

Fig. 5-6: Example1 in the WIN-HMI

Program Code in the Application Builder

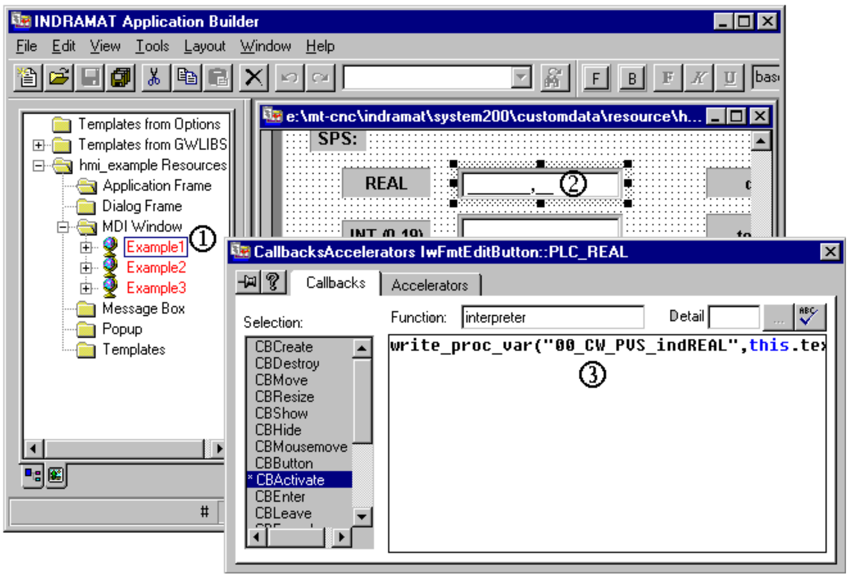
Requirements: **HMI_Example.cgw** must be open in the Application Builder.

⇒ Open the **Example1** MDI window by <double-clicking> ①.

⇒ <Right mouse button> on IwFmtEditButton to enter real number.②
A pop-up menu is opened.

⇒ <Click> the **Callbacks&Accelerators** menu.

⇒ <Click> **CBActivate.**
The callback is displayed in the edit window.③, see Fig. 5-7.



```
write_proc_var("00_CW_PVS_indREAL",this.te
```

Bild17ix.bmp

(1): Selection in the Navigator
(2): IwFmtEditButton for real number entry
(3): CBActivate for real number entry with IwFmtEditButton

Fig. 5-7: Display of the callbacks for entry

IwFmtEditButton: PLC_REAL CBActivate

```
write_proc_var("00_CW_PVS_indREAL",this.text());
```

IwFmtEditButton: PLC_INT CBActivate

```
write_proc_var("00_CW_PVS_indINT",this.text());
```

IwFmtEditButton: PLC_BOOL CBActivate

```
write_proc_var("00_CW_PVS_indBOOL",this.text());
```

IwFmtEditButton: PLC_STRING CBActivate

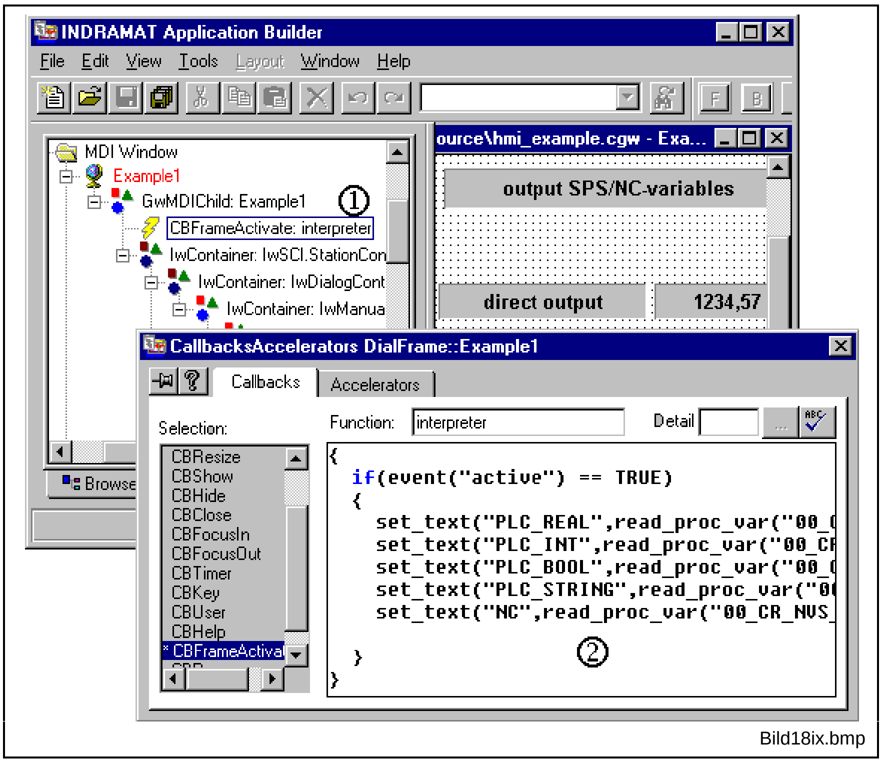
```
write_proc_var("00_CW_PVS_indSTRING",this.text());
```

IwFmtEditButton: PLC_NC CBActivate

```
write_proc_var("00_CW_NVS_6_10",this.text());
```

CBFrameActivate for first reading after opening

After Example1 has been opened, the variables of the SPS or NC are for once read out by means of **CBFrameActivate** and written to the EditButtons for entry. This method ensures that the previously written state (before closing) forms the basis for continuation.



```
{
  if(event("active") == TRUE)
  {
    set_text("PLC_REAL",read_proc_var("00_
    set_text("PLC_INT",read_proc_var("00_C
    set_text("PLC_BOOL",read_proc_var("00_
    set_text("PLC_STRING",read_proc_var("0
    set_text("NC",read_proc_var("00_CR_NVS_

  }
}
```

Bild18ix.bmp

(1): CBFrameActivate callback in the Navigator
(2): CBFrameActivate in the edit window

Fig. 5-8: CBFrameActivate of Example1

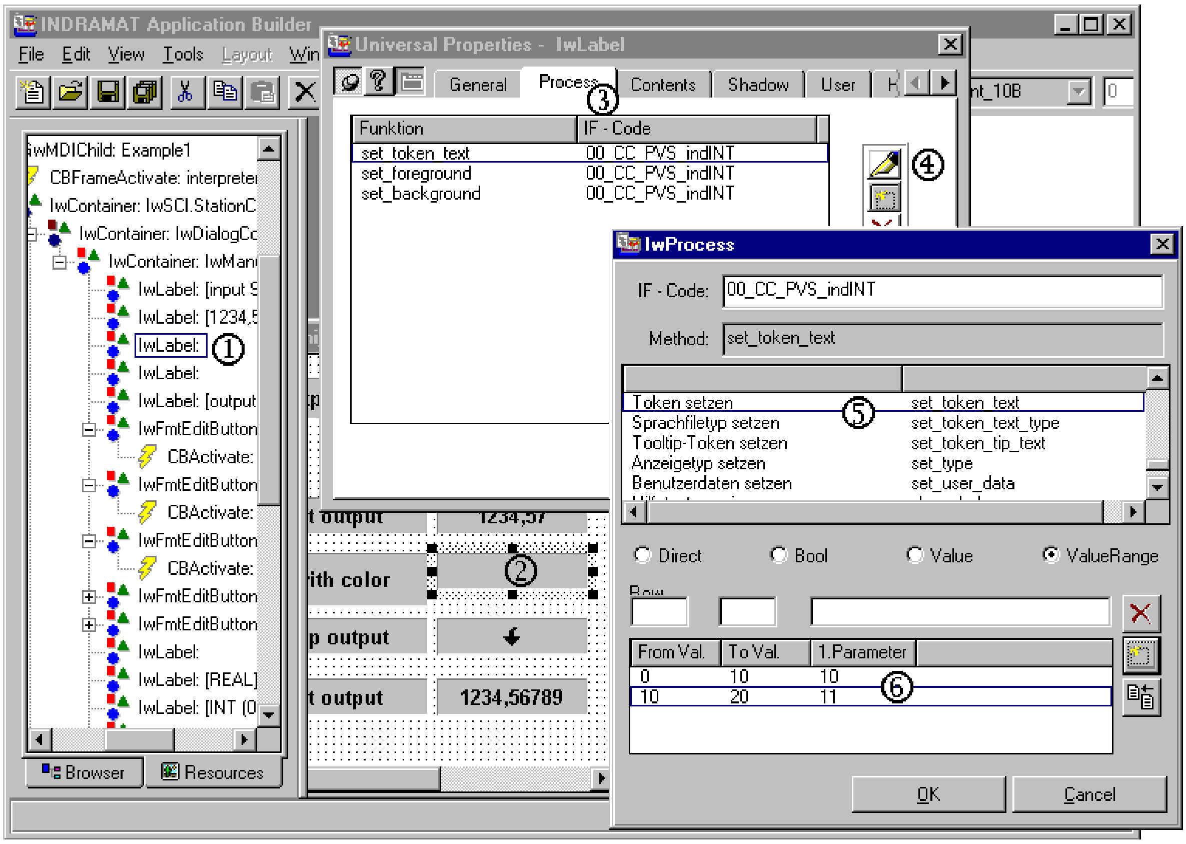
⇒ Double-click (1).
The callback is displayed in an edit window.

GwMDIChild: Example1 CBFrameActivate

```
{
  if(event("active") == TRUE)
  {
    set_text("PLC_REAL",read_proc_var("00_CR_PVS_indREAL"));
    set_text("PLC_INT",read_proc_var("00_CR_PVS_indINT"));
    set_text("PLC_BOOL",read_proc_var("00_CR_PVS_indBOOL"));
    set_text("PLC_STRING",read_proc_var("00_CR_PVS_indSTRING"));
    set_text("NC",read_proc_var("00_CR_NVS_6_10"));
  }
}
```

Continuous reading of the variables from the SPS and NC and their formated output

The output of variables is assumed by **IwLabel**s. The pertinent functions have been determined under Properties. The functions for an IwLabel (②) are represented in the dialogs of the Application Builder, Fig. 5-9.


INDRAMAT Application Builder
File Edit View Tools Layout Win
Universal Properties - IwLabel
General Process Contents Shadow User
IwProcess
Direct Bool Value ValueRange
Row
OK Cancel
Browser Resources
1 2 3 4 5 6

Funktion	IF - Code
set_token_text	00_CC_PVS_indINT
set_foreground	00_CC_PVS_indINT
set_background	00_CC_PVS_indINT

IF - Code: 00_CC_PVS_indINT

Method: set_token_text

Token setzen	set_token_text
Sprachfiletyp setzen	set_token_text_type
Tooltip-Token setzen	set_token_tip_text
Anzeigetyp setzen	set_type
Benutzerdaten setzen	set_user_data

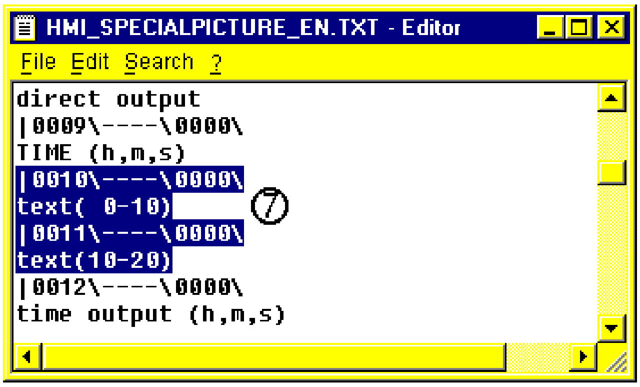
○ Direct ○ Bool ○ Value ● ValueRange

From Val.	To Val.	1.Parameter
0	10	10
10	20	11

Bild20ix.bmp

(1): IwLabel in the Navigator for output of text and color
(2): IwLabel for output of text and color
(3): Universal Properties dialog
(4): Calling up the edit window
(5): Process connection edit window
(6): Parameters for the text file

Fig. 5-9: Properties dialogs in the Application Builder



```
direct output
|0009\----\0000\
TIME (h,m,s)
|0010\----\0000\
text( 0-10)
|0011\----\0000\
text(10-20)
|0012\----\0000\
time output (h,m,s)
```

Bild21.bmp

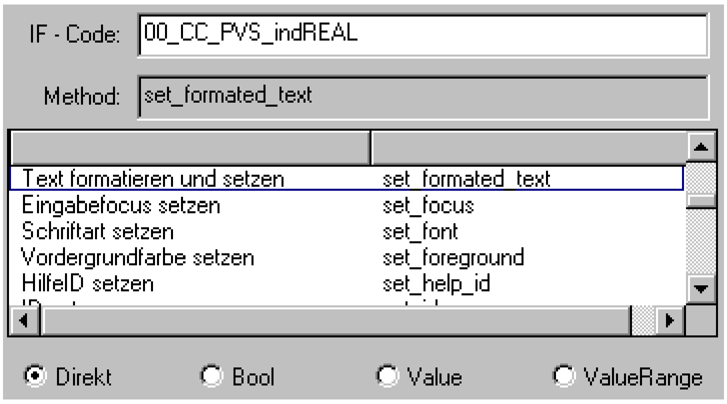
(7): Parametes for the text file

Fig. 5-10: “HMI_Specialpicture_DE.txt” text file

The text to be output from the text file for the output example of “multilingual text with color” is represented in Fig. 5-10.

Functions used for output

Direct output of a real number:



IF - Code: 00_CC_PVS_indREAL

Method: set_formated_text

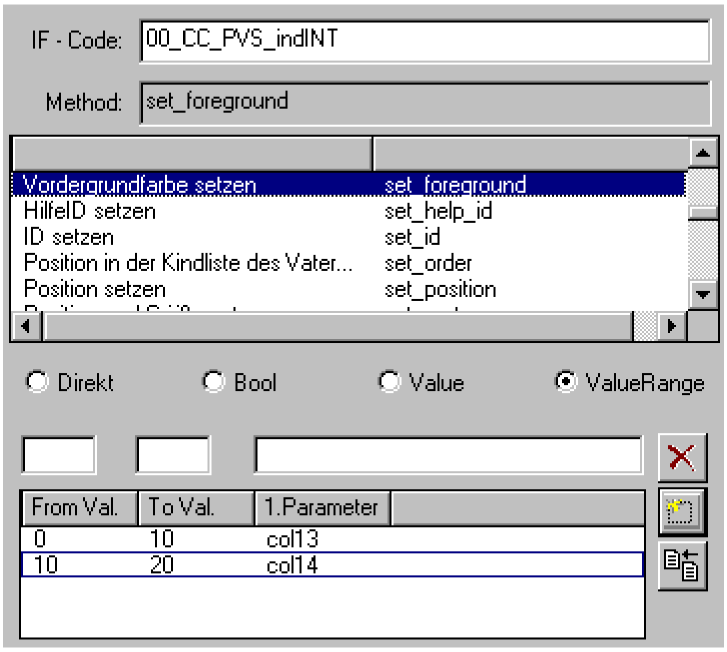
Text formatieren und setzen	set_formated_text
Eingabefocus setzen	set_focus
Schriftart setzen	set_font
Vordergrundfarbe setzen	set_foreground
HilfeID setzen	set_help_id

● Direkt ○ Bool ○ Value ○ ValueRange

Bild22.bmp

Fig. 5-11: Direct real number output

Multilingual text with color with integer:



IF - Code: 00_CC_PVS_indINT

Method: set_foreground

Vordergrundfarbe setzen	set_foreground
HilfeID setzen	set_help_id
ID setzen	set_id
Position in der Kindliste des Vater...	set_order
Position setzen	set_position

○ Direkt ○ Bool ○ Value ● ValueRange

From Val.	To Val.	1.Parameter
0	10	col13
10	20	col14

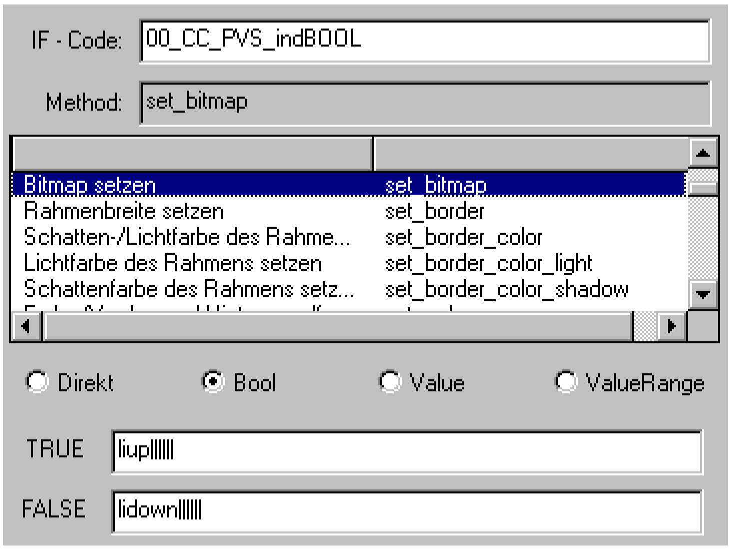
Bild23.bmp

Fig. 5-12: Setting the foreground color with the Value range

Three methods form part of this output:

1. set_token_text see Fig. 5-9
2. set_foreground see Fig. 5-12
3. set_background in analogy to the set_foreground method

Bitmap output with Bool:



IF - Code: 00_CC_PVS_indBOOL

Method: set_bitmap

Bitmap setzen	set_bitmap
Rahmenbreite setzen	set_border
Schatten-/Lichtfarbe des Rahme...	set_border_color
Lichtfarbe des Rahmens setzen	set_border_color_light
Schattenfarbe des Rahmens setz...	set_border_color_shadow

○ Direkt ● Bool ○ Value ○ ValueRange

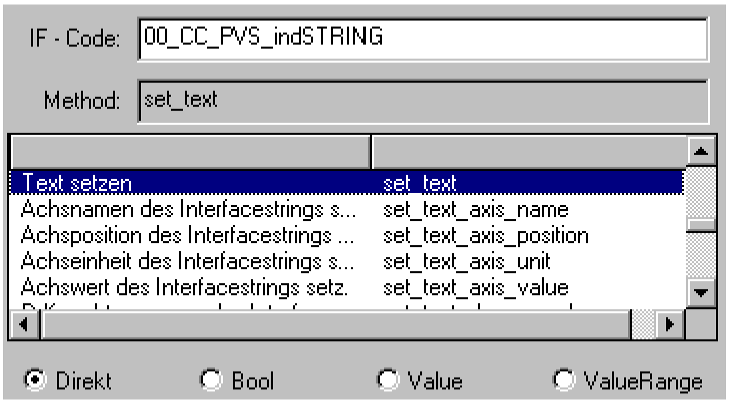
TRUE liup|||||||

FALSE lidown|||||||

Bild24.bmp

Fig. 5-13: Setting the Bitmap

Direct output of a string:



IF - Code: 00_CC_PVS_indSTRING

Method: set_text

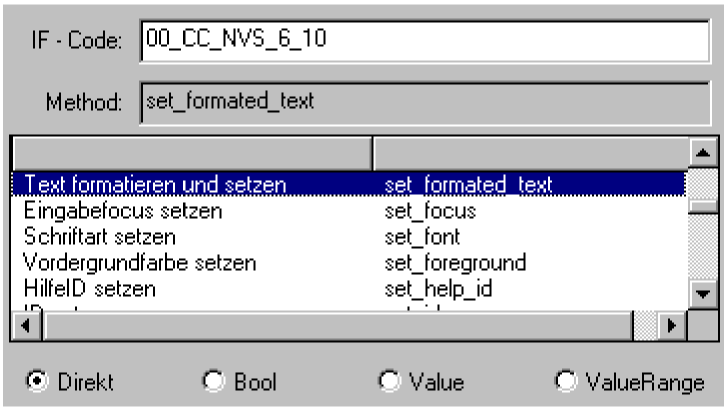
Text setzen	set_text
Achsnamen des Interfacestrings s...	set_text_axis_name
Achsposition des Interfacestrings ...	set_text_axis_position
Achseinheit des Interfacestrings s...	set_text_axis_unit
Achswert des Interfacestrings setz.	set_text_axis_value

● Direkt ○ Bool ○ Value ○ ValueRange

Bild25.bmp

Fig. 5-14: Direct string output

Direct output of an NC variable:



IF - Code: 00_CC_NVS_6_10

Method: set_formated_text

Text formatieren und setzen	set_formated_text
Eingabefocus setzen	set_focus
Schriftart setzen	set_font
Vordergrundfarbe setzen	set_foreground
HilfeID setzen	set_help_id

● Direkt ○ Bool ○ Value ○ ValueRange

Bild26.bmp

Fig. 5-15: Direct real number output

5.2 Example2: Structure Variable

Example 2 is contained as the **Example2** MDI window in HMI_Example.cgw, see Fig. 5-2. A structure element is used for data exchange between the HMI user interface and the SPS. The structure definition has been loaded together with the SPS program (see Example1).

Structure Display in the SPS

Requirements: SPS program (Example1) loaded; SPS user interface activated; status display ON.

⇒ Open the **Edit\Declaration** menu.

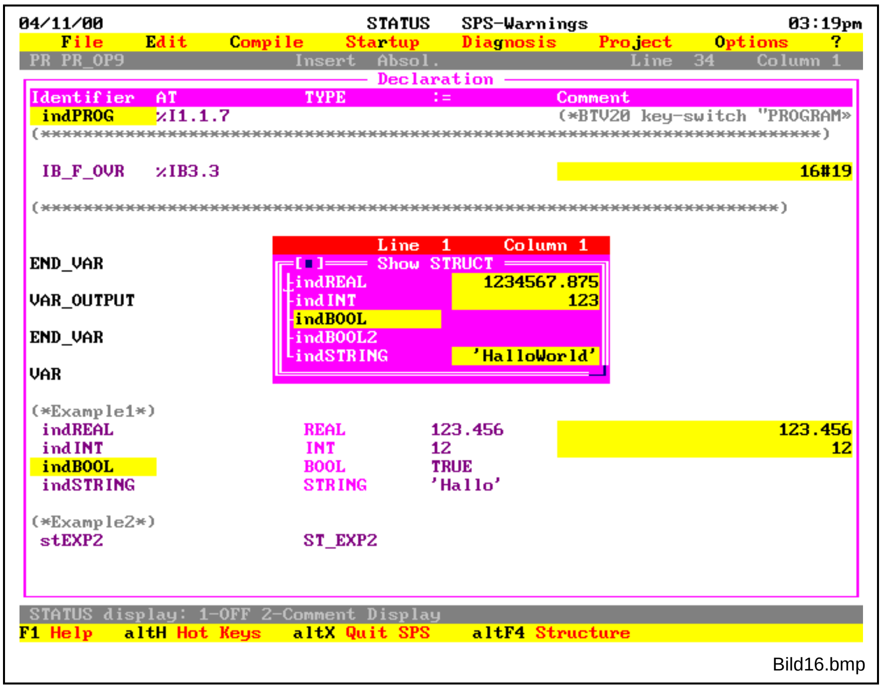
⇒ Place the cursor on the **stEXP2** structure variable.

⇒ Press <Shift>+<F3>.

A dialog with the multi-element **stEXP2** variable is opened.

⇒ Click the <OK> button.

A picture similar to that below appears:



```
04/11/00                           STATUS   SPS-Warnings                  03:19pm
     File      Edit      Compile      Startup      Diagnosis      Project      Options      ?
 PR PR_OP9                          Insert  Absol.                 Line  34    Column 1
                                    Declaration
Identifier  AT              TYPE            :=             Comment
 indPROG    %I1.1.7                                        (*BTV20 key-switch "PROGRAM»
(*****************************************************************************)

 IB_F_OVR   %IB3.3                                                         16#19

(************************************************************************)

                                    Line  1       Column 1
END_VAR                         [■]═══ Show STRUCT
                                 indREAL           1234567.875
VAR_OUTPUT                       indINT                    123
                                 indBOOL
END_VAR                          indBOOL2
                                 indSTRING          'HalloWorld'
VAR

(*Example1*)
 indREAL                   REAL       123.456                           123.456
 indINT                    INT        12                                     12
 indBOOL                   BOOL       TRUE
 indSTRING                 STRING     'Hallo'

(*Example2*)
 stEXP2                    ST_EXP2

 STATUS display: 1-OFF 2-Comment Display
F1 Help   altH Hot Keys   altX Quit SPS   altF4 Structure
```

Bild16.bmp

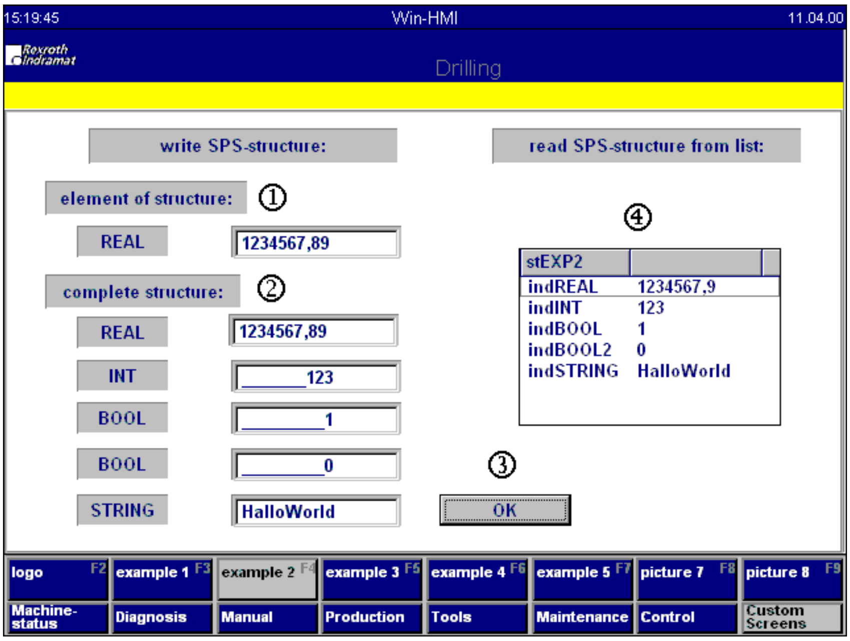
Fig. 5-16: Status display of the “stEXP2” structure

The values of the variable have been entered in the WIN-HMI user interface beforehand, see Fig. 5-17.

Starting WIN-HMI

Requirements: The Special Pictures menu of the WIN-HMI is open.

⇒ Press the <Example2> (<F4>) button.
A picture similar to that below is opened:



REAL: 1234567,89

REAL: 1234567,89
INT: 123
BOOL: 1
BOOL: 0
STRING: HalloWorld

stEXP2	
indREAL	1234567,9
indINT	123
indBOOL	1
indBOOL2	0
indSTRING	HalloWorld

Bild4ix.bmp

(1): IwFmtEditButton::PLC_REAL
(2): responsible for structure entry: IwFmtEditButton::PLC_REAL1, IwFmtEditButton::PLC_INT, IwFmtEditButton::PLC_BOOL, IwFmtEditButton::PLC_BOOL2, IwFmtEditButton::PLC_STRING
(3): IwPushButton
(4): IwMultiColumnScrolledList::liste

Fig. 5-17: WIN-HMI, Example2

⇒ Enter *1234567,89* in the Edit button (1).

⇒ <Enter>.
The number is displayed both in the REAL button “complete structure” (2) and in the list (4).

Note: In the example, the real variable is transferred to the SPS outside of the structure as well.

⇒ Enter *123* in the Edit button for INT.

⇒ Enter *1* in the Edit button for BOOL.

⇒ Enter *0* in the Edit button for BOOL.

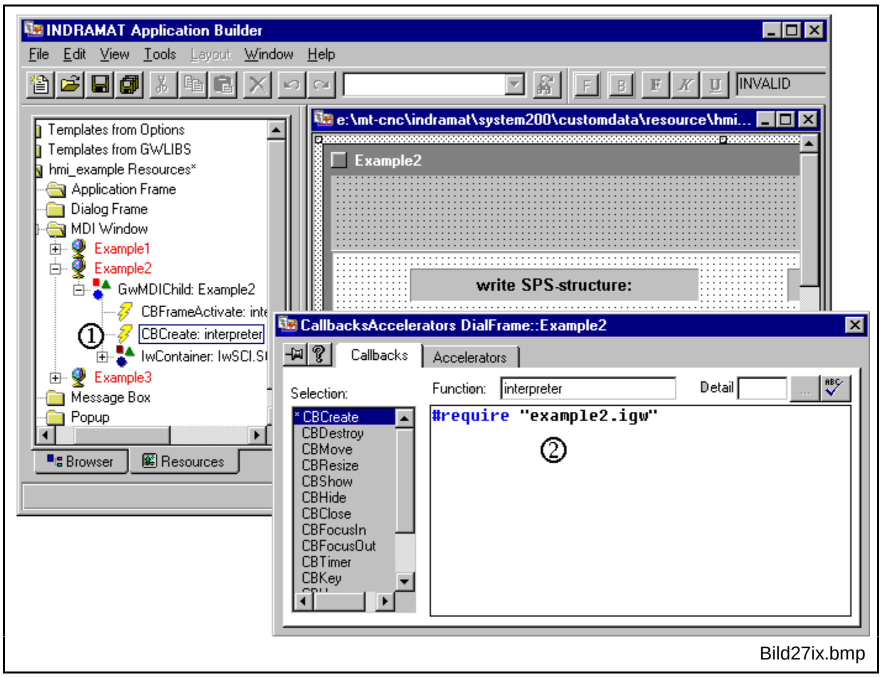
⇒ Enter *HalloWorld* in the Edit button for STRING.

⇒ Click <Apply>.
The values entered are displayed in the list (4).

Program Code for Writing the Variable to the SPS

Definition of a structure

Example2 is different from Example1 mainly in that the variables have been comprised to form a structure. One advantage is the quicker data exchange between the WIN-HMI and the SPS or NC. The **writePLCstruct** function has been created for writing a structure to the SPS. This function is contained in the **example2.igw** file.

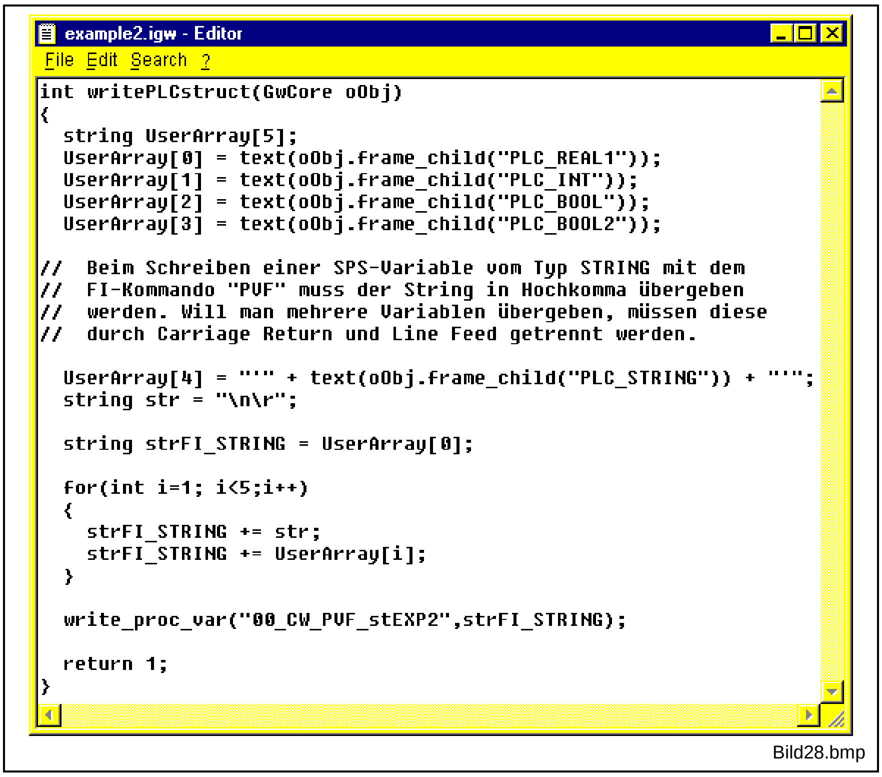


```
#require "example2.igw"
```

Bild27ix.bmp

(1): GwMDIChild: Example2
(2): CBCreate

Fig. 5-18: The igw-file contains the “writePLCstruct” function



```
int writePLCstruct(GwCore oObj)
{
  string UserArray[5];
  UserArray[0] = text(oObj.frame_child("PLC_REAL1"));
  UserArray[1] = text(oObj.frame_child("PLC_INT"));
  UserArray[2] = text(oObj.frame_child("PLC_BOOL"));
  UserArray[3] = text(oObj.frame_child("PLC_BOOL2"));

//  Beim Schreiben einer SPS-Variable vom Typ STRING mit dem
//  FI-Kommando "PVF" muss der String in Hochkomma übergeben
//  werden. Will man mehrere Variablen übergeben, müssen diese
//  durch Carriage Return und Line Feed getrennt werden.

  UserArray[4] = "'" + text(oObj.frame_child("PLC_STRING")) + "'";
  string str = "\n\r";

  string strFI_STRING = UserArray[0];

  for(int i=1; i<5;i++)
  {
    strFI_STRING += str;
    strFI_STRING += UserArray[i];
  }

  write_proc_var("00_CW_PVF_stEXP2",strFI_STRING);

  return 1;
}
```

Bild28.bmp

Fig. 5-19: Function for writing a structure to the SPS

Syntax	Function	Example
CR_PVF_①	Reading	
CW_PVF_①	Writing	00_CW_PVF_stEXP2
CC_PVF_①	Cyclic reading	00_CC_PVF_stEXP2
①Structure name		

Fig. 5-20: Syntax for a structure variable

Below, those callbacks are listed which are responsible for writing the individual variables to the SPS:

GwMDIChild: Example2
CBFrameActivate

The following CB is processed by activating GwMDIChild: **Example2:**

```
{
  if(event("active") == TRUE)
  {
    set_text("PLC_REAL",read_proc_var("00_CR_PVS_stEXP2.indREAL"));
    set_text("PLC_REAL1",read_proc_var("00_CR_PVS_stEXP2.indREAL"));

    set_text("PLC_INT",read_proc_var("00_CR_PVS_stEXP2.indINT"));
    set_text("PLC_BOOL",read_proc_var("00_CR_PVS_stEXP2.indBOOL"));
    set_text("PLC_BOOL2",read_proc_var("00_CR_PVS_stEXP2.indBOOL2"));
    set_text("PLC_STRING",read_proc_var("00_CR_PVS_stEXP2.indSTRING"));

    start_timer("liste",500);
  }
  else
  {
    stop_timer("liste");
  }
}
```

The contents of the variables from the SPS are written to the six IwFmtEditButtons upon activation of Example2.

The PLC_REAL button has a special position within the display window: the real value entered here is formated, written to the SPS, and displayed in the PLC_REAL1 button.

IwFmtEditButton: PLC_REAL
CBActivate

```
{
string str = awptof(this.text());
write_proc_var("00_CW_PVS_stEXP2.indREAL",str);

set_text("PLC_REAL1",this.text());
}
```

IwFmtEditButton: PLC_REAL
CBFocusOut

```
set_text("PLC_REAL1",this.text());
```

Use the **Apply** IwPushButton to transfer the values entered to the SPS:

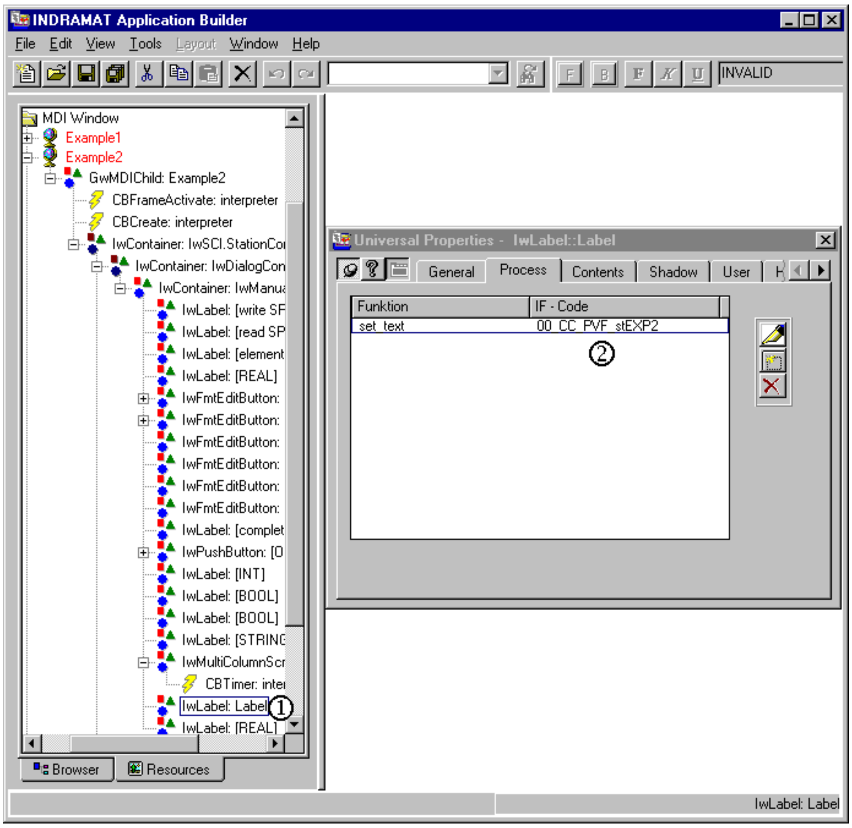
IwPushButton
CBActivate

```
writePLCstruct(this);
```

The function executing the transfer to the SPS is called up.

Program Code for Reading the Variable from the SPS

Here, the **Label** IwLabel's property of continuously reading the **stEXP2** structure from the SPS is utilized. The label is used for transmission only (not for output), and it is outside of the visible area.



Funktion	IF - Code
set_text	00_CC_PVF_stEXP2

Bild29ix.bmp

(1): Iwlabel: Label (for data transfer only)
(2): IF-Code for process connection

Fig. 5-21: Data transfer as a property of an IwLabels

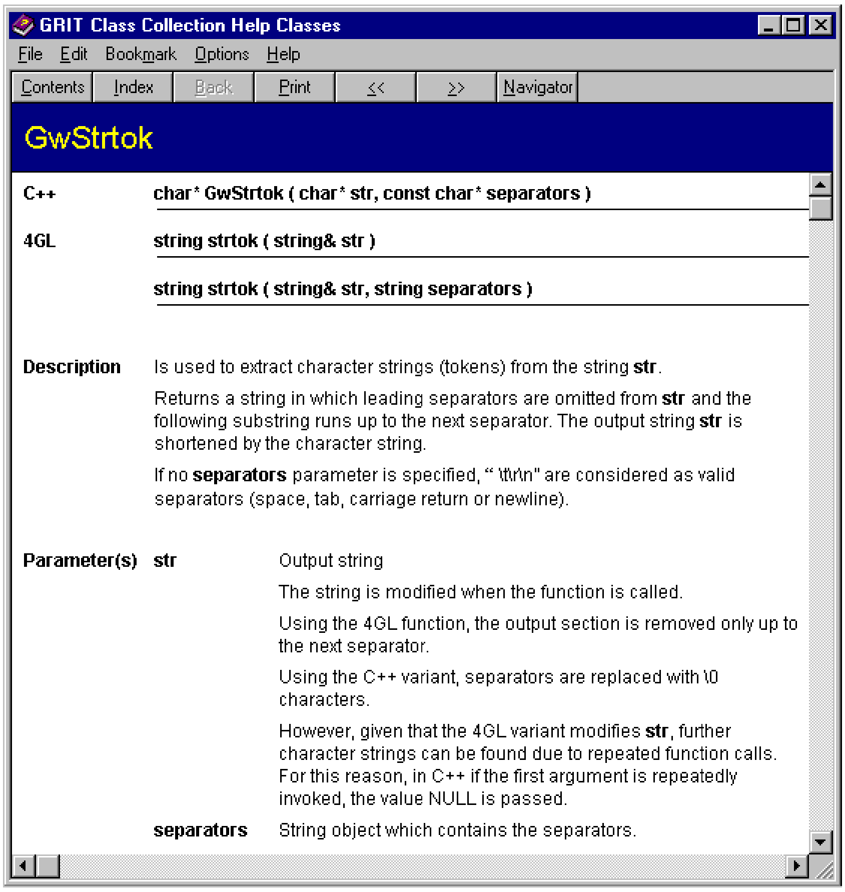
The **00_CC_PVF_stEXP2** IF-Code is used to continuously read the variables from the SPS. The contents are contained in a string of the **Label** IwLabel.

The structure variables are displayed in the IwMultiColumnScolledList: **liste.** The values are refreshed every 500 msec.

IwMultiColumnScrolledList: liste CBTimer

```
{
  string strFI_STRING = text("Label");
  string strTemp;
  for(int i=0; i<5; i++)
  {
      strTemp = strtok(strFI_STRING,"\n\r");

      this.set_item(strTemp,i,1);
  }
}
```

The **text ("Label")** string is taken apart in the five single variables by utilizing the **strtok** method. The description can be found in the Help menu:


GRIT Class Collection Help Classes
File Edit Bookmark Options Help
Contents Index Back Print << >> Navigator

GwStrtok

C++ **char* GwStrtok (char* str, const char* separators)**

4GL **string strtok (string& str)**

string strtok (string& str, string separators)

Description Is used to extract character strings (tokens) from the string **str**.

Returns a string in which leading separators are omitted from **str** and the following substring runs up to the next separator. The output string **str** is shortened by the character string.

If no **separators** parameter is specified, " \t\r\n" are considered as valid separators (space, tab, carriage return or newline).

Parameter(s) **str** Output string

The string is modified when the function is called.

Using the 4GL function, the output section is removed only up to the next separator.

Using the C++ variant, separators are replaced with \0 characters.

However, given that the 4GL variant modifies **str**, further character strings can be found due to repeated function calls. For this reason, in C++ if the first argument is repeatedly invoked, the value NULL is passed.

separators String object which contains the separators.

Bild30.bmp

Fig. 5-22: Help for the “strtok” method

5.3 Example3: Machine Data and Focus Change

Example3 is used to test user machine data. The three variables for these machine data must be defined both in the HMI user interface and the NC controller. Only then can the contents be exchanged via the function interface.

User Machine Data

The notation (see function interface) is as follows:

Syntax	Function	Example
CR_MTD_①_②_③_④_⑤ CR_MTD1_①_②_③_④_⑤	Reading	00_CR_MTD1_200_1_0_1_12
CW_MTD_①_②_③_④_⑤	Writing	00_CW_MTD_200_1_0_1_12
CC_MTD_①_②_③_④_⑤ CC_MTD1_①_②_③_④_⑤	Cyclic reading	00_CC_MTD1_200_1_0_1_12
①Page No. [1...299] ②Run variable1 [-/+1000] ③Run variable2 [-/+1000] ④Element No. [1...1000] ⑤Data type [1...13]		

Fig. 5-23: Syntax for NC variable

Importing NC machine data

A user machine dataset has been prepared for Example3, which contains the three variables used in the example. Their definition is contained in the **Example3.exp** file, which must be imported from the NC controller as follows:

Requirements: The WIN-HMI user interface is activated.
Example3.exp resides on a diskette in drive A or on the hard disk of the controller (the folder must be known).

⇒ Press <Shift>+<F4> to move to the main menu of the MUI.

⇒ Press <F3> to call the machine data.

⇒ Press <Ctrl>+<F8> to prepare the machine data.

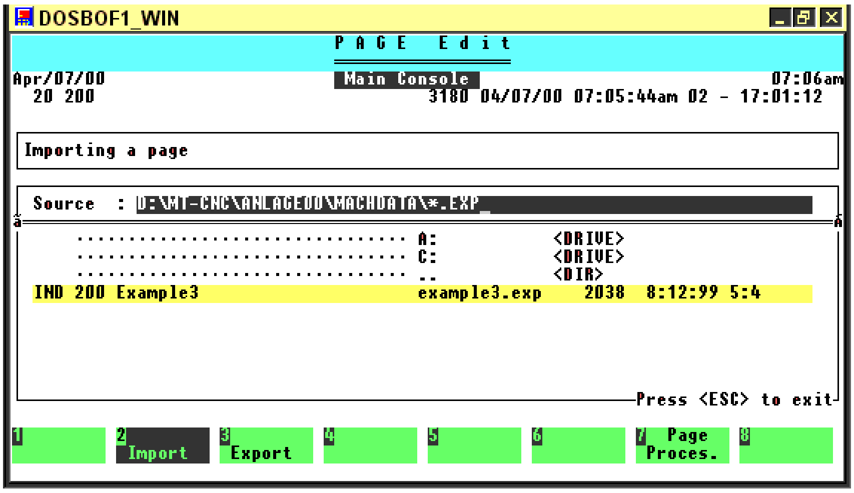
⇒ Press <F1> to create a new machine dataset.

⇒ Enter the machine dataset number, e.g. No. 20.

⇒ 200 must be entered as the name of the machine dataset.

⇒ Press <F7> to change the machine data/definition.

⇒ Press <Ctrl>+<F2> to import a machine data page.



```
DOSBOF1_WIN
                              P A G E   E d i t
Apr/07/00                     Main Console                        07:06am
  20  200                          3180  04/07/00 07:05:44am 02 - 17:01:12

 Importing a page

  Source  : D:\MT-CNC\ANLAGE00\MACHDATA\*.EXP_

      ................................ A:            <DRIVE>
      ................................ C:            <DRIVE>
      ................................ ..            <DIR>
  IND  200  Example3                  example3.exp     2038   8:12:99 5:4

                                                       Press <ESC> to exit
1       2 Import  3 Export  4       5       6       7 Page Proces.  8
```

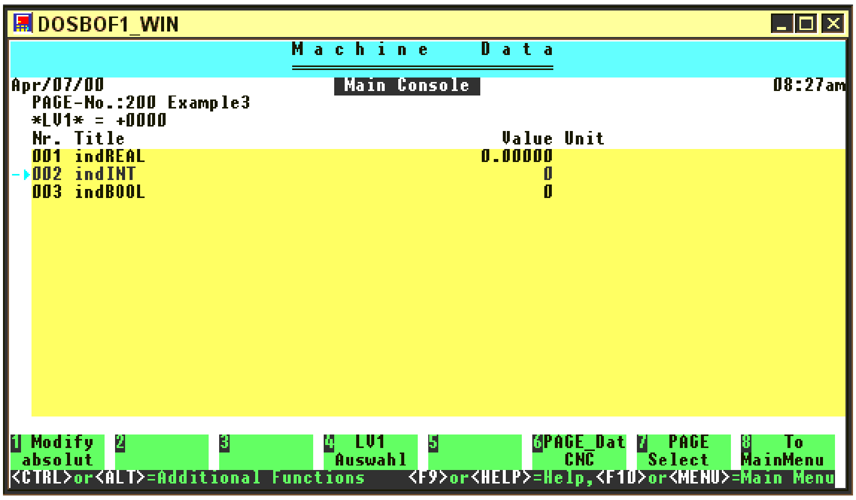
Bild15.bmp

Fig. 5-24: Importing a Page

Instruction on how to search the source

If the **Example3.exp** source file resides in a subdirectory, it will become visible, if the display area is extended by pressing the <TAB>, <DOWN> and/or <UP> keys.

⇒ Select the machine dataset and specify Page No. 200.



```
DOSBOF1_WIN
                         M a c h i n e    D a t a
Apr/07/00                     Main Console                        08:27am
  PAGE-No.:200 Example3
  *LV1* = +0000
  Nr. Title                                  Value Unit
  001 indREAL                              0.00000
→ 002 indINT                                     0
  003 indBOOL                                    0

1 Modify absolut  2  3  4 LV1 Auswahl  5  6 PAGE_Dat CNC  7 PAGE Select  8 To MainMenu
<CTRL>or<ALT>=Additional Functions    <F9>or<HELP>=Help,<F10>or<MENU>=Main Menu
```

Bild14.bmp

Fig. 5-25: Content of the Page

⇒ Press <F6> to load the Page to the controller.

⇒ Press <F8> to return to the machine data preparation.

⇒ Press <F10> to return to the main menu.

Starting WIN-HMI

Requirements: The Special Pictures menu of the WIN-HMI is open.

⇒ Click the <Example3> (<F5>) button.

The picture below is opened:

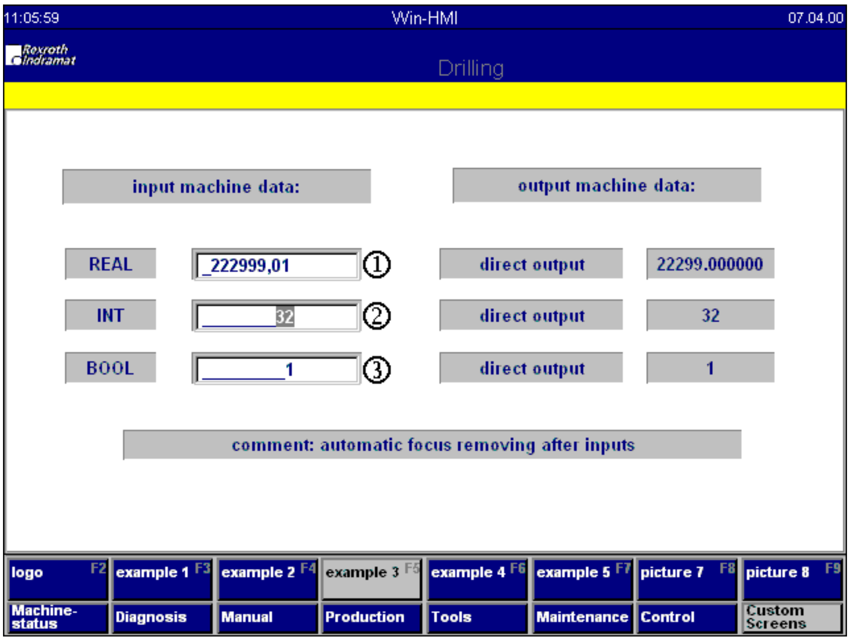


Bild5ix.bmp

(1): Entry of real number
(2): Entry of integer
(3): Entry of Boolean number (0, 1)

Fig. 5-26: Entry of machine data

⇒ Enter *222999,01* in the Edit button (1).

⇒ <Enter>.

The number 222999.000000 is displayed as “direct output”.

The Edit button (2) receives the focus.

⇒ Enter *32* in the Edit button (2).

⇒ <Enter>.

The number 32 is displayed as “direct output”.

The Edit button (3) receives the focus.

⇒ Enter *1* in the Edit button (3).

⇒ <Enter>.

The number 1 is displayed as “direct output”.

The Edit button (1) receives the focus.

Program Code for Writing the Machine Data

The task of writing the variables is assumed by the following callbacks. The different formats for entering the values is set in the properties of the IwFmtEditButtons.

The focus is transmitted utilizing the **set_focus** method.

IwFmtEditButton: PLC_REAL CBActivate

```
{
write_proc_var("00_CW_MTD_200_1_0_1_12",this.text());
set_focus("PLC_INT");
}
```

IwFmtEditButton: PLC_INT CBActivate

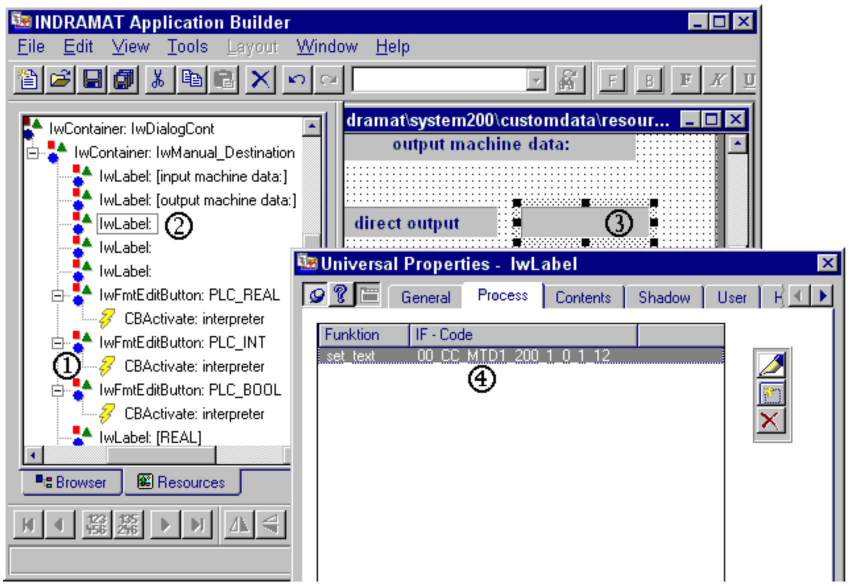
```
{
write_proc_var("00_CW_MTD_200_1_0_2_7",this.text());
set_focus("PLC_BOOL");
}
```

IwFmtEditButton: PLC_BOOL CBActivate

```
{
write_proc_var("00_CW_MTD_200_1_0_3_1",this.text());
set_focus("PLC_REAL");
}
```

Program Code for Reading the Machine Data

Reading of the machine data from the NC is realized by means of the Label properties. The content of the permanently readout machine data is directly displayed in the corresponding IwLabel.


INDRAMAT Application Builder
File Edit View Tools Layout Window Help
dramat\system200\customdata\resour...
output machine data:
direct output
Universal Properties - IwLabel
General
Process
Contents
Shadow
User
Browser
Resources
1
2
3
4

IwContainer: IwDialogCont
IwContainer: IwManual_Destination
IwLabel: [input machine data:]
IwLabel: [output machine data:]
IwLabel:
IwLabel:
IwLabel:
IwFmtEditButton: PLC_REAL
CBActivate: interpreter
IwFmtEditButton: PLC_INT
CBActivate: interpreter
IwFmtEditButton: PLC_BOOL
CBActivate: interpreter
IwLabel: [REAL]

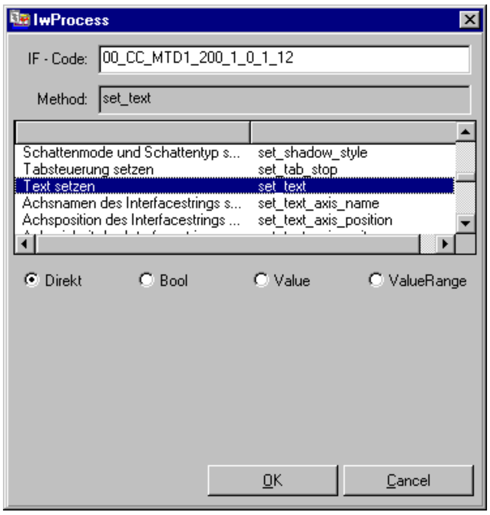
Funktion	IF - Code
set text	00 CC MTD1 200 1 0 1 12

Bild31ix.bmp

(1): 3 CBs for writing the machine data
(2): IwLabel in the Navigator
(3): IwLabel in the window
(4): Process connection of the IwLabel for the reading process

Fig. 5-27: Process connection for reading machine data

The variable is entered and the function selected in the following menu:



IwProcess

IF - Code: 00_CC_MTD1_200_1_0_1_12

Method: set_text

Schattenmode und Schattentyp s...	set_shadow_style
Tabsteuerung setzen	set_tab_stop
Text setzen	set_text
Achsnamen des Interfacestrings s...	set_text_axis_name
Achsposition des Interfacestrings ...	set_text_axis_position

● Direkt ○ Bool ○ Value ○ ValueRange

Bild32.bmp

Fig. 5-28: Entry of the IF-Code and selection of the method

Display of IwLabel for the PLC_REAL value:

⇒ Enter *00_CC_MTD1_200_1_0_1_12*

Display of IwLabel for the PLC_INT value:

⇒ Enter *00_CC_MTD1_200_1_0_2_7*

Display of IwLabel for the PLC_BOOL value:

⇒ Enter *00_CC_MTD1_200_1_0_3_1*

6 Dynamic Application

6.1 Example 4

Explanations

In Example 4, the movement of a conveyor belt is reproduced. The symbol of an MTC is moved in an IwShapeBitmap object in relation to the rotation of the drive wheels. The spokes of the drive wheels are IwShapeEllipse objects. The small control wheel is composed of IwShapeEllPie objects. The position values of the dynamic objects are deducted from an IwSlider object. The invisible slider can be moved by means of the <CursorRight> and <CursorLeft> keys.

The example is stored in the “example4.cgw” file.

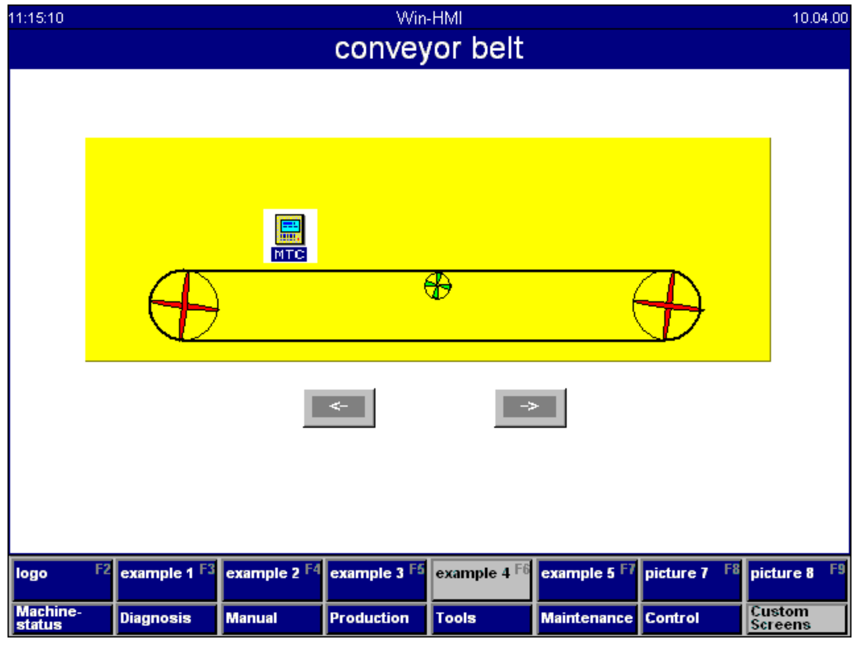


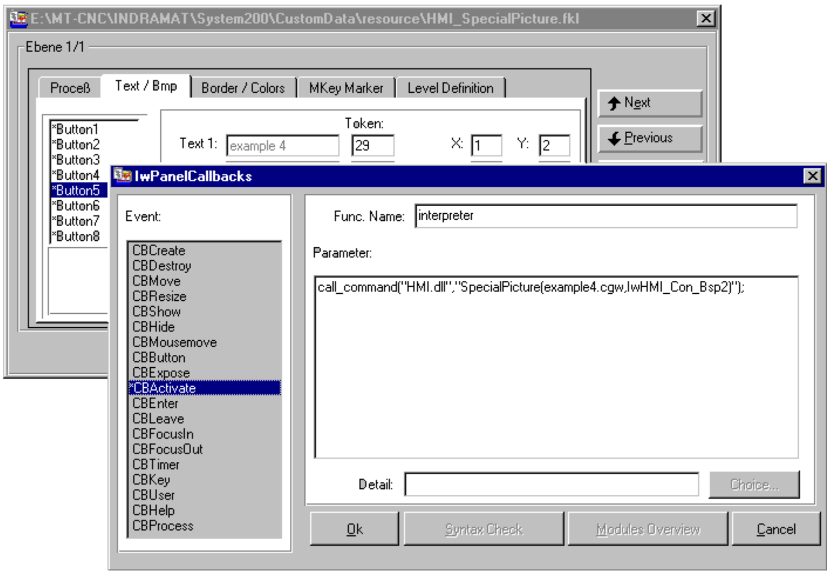
Bild6k.bmp

Fig. 6-1: Example 4 started as application

Call Mechanism for Example 4

“Example 4” is started on Button5. The call is stored in the HMI_SpecialPicture.fkl file:

HMI_SpecialPicture.fkl



E:\MT-CNC\INDRAMAT\System200\CustomData\resource\HMI_SpecialPicture.fkl

Text 1: example 4
Token: 29
X: 1
Y: 2

IwPanelCallbacks

Event: CBActivate

Func. Name: interpreter

Parameter:

```
call_command("HMI.dll","SpecialPicture(example4.cgw,IwHMI_Con_Bsp2)");
```

Detail:

Bild57.bmp

Fig. 6-2: FKL Editor with HMI_SpecialPicture.fkl

The “CBActivate” callback for “Button 5” receives the following content:

CBActivate for Button 5

```
call_command("HMI.dll","SpecialPicture(example4.cgw,IwHMI_Con_Bsp2)");
```

The syntax for this call is described on Page 2-11.

The example resides in the “example4.cgw” file, and the dialog window has the ID **IwHMI_Con_Bsp2.**

Preview of Example 4

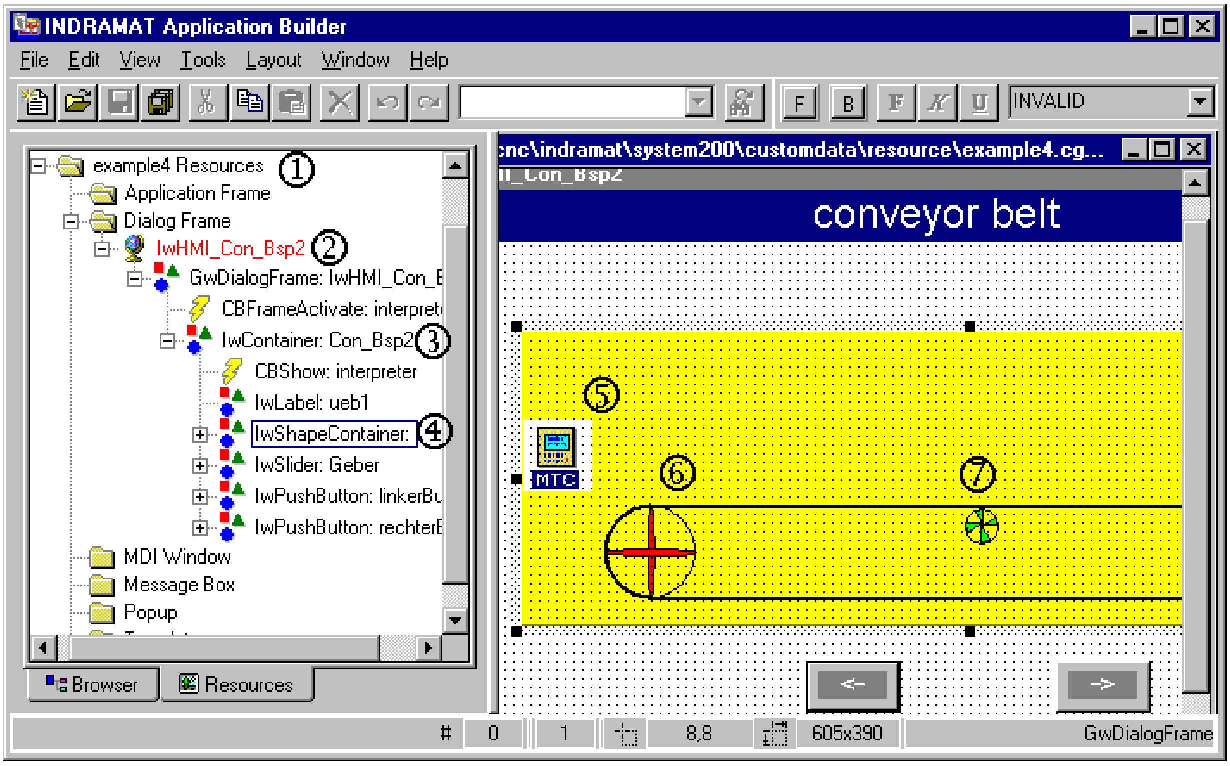


Bild51.bmp

Fig. 6-3: Example 4 in Example4.cgw

(1) The resource file is called “Example4.cgw”.

(2) This file contains an “IwHMI_Con_Bsp2” dialog window.

(3) The dialog window possesses an IwContainer.

(4) IwShapeContainer is the prerequisite for taking up Shape objects.

Important Shape objects in the IwShapeContainer:

(5) IwShapeBitmap object

(6) IwShapeEllipsen objects

(7) IwShapeEllPie objects

Inserting Shape Objects in an IwShape Container

The picture below shows the assignment of some of the objects.

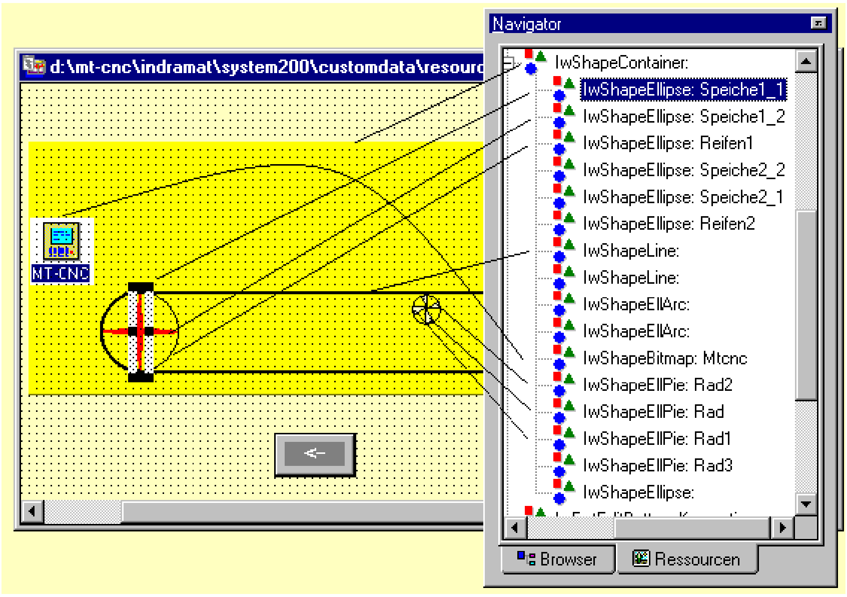


Bild55.bmp

Fig. 6-4: IDs of the Shape objects

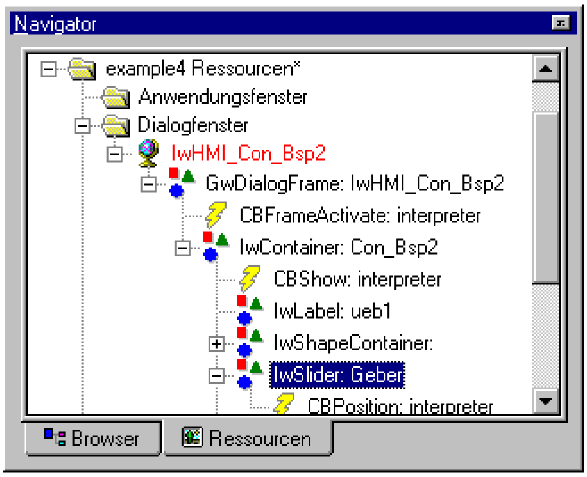
The objects are “dragged” from the toolbar for the dynamic objects to the IwShape-Container in the known manner.

IwSlider Object as Command Value Encoder (Geber)

In the “Example4”, an IwSlider object serves as the command value encoder for the movements of the Shape objects. The slider is moved by pressing the <CursorRight> and <CursorLeft> keys. Movement by means of the mouse should not be possible. For that reason, the slider can be taken out of the visible range of the IwContainer. This can be achieved, for instance, by placing the position value for Pos.X of the IwSlider object in the container in the invisible range. The results and procedures relating to the IwSlider remain activated.

Making IwSlider visible

In order to exercise moving the IwSlider to the visible range, the Y-position must be changed in the Properties window:



Slidernavi.bmp

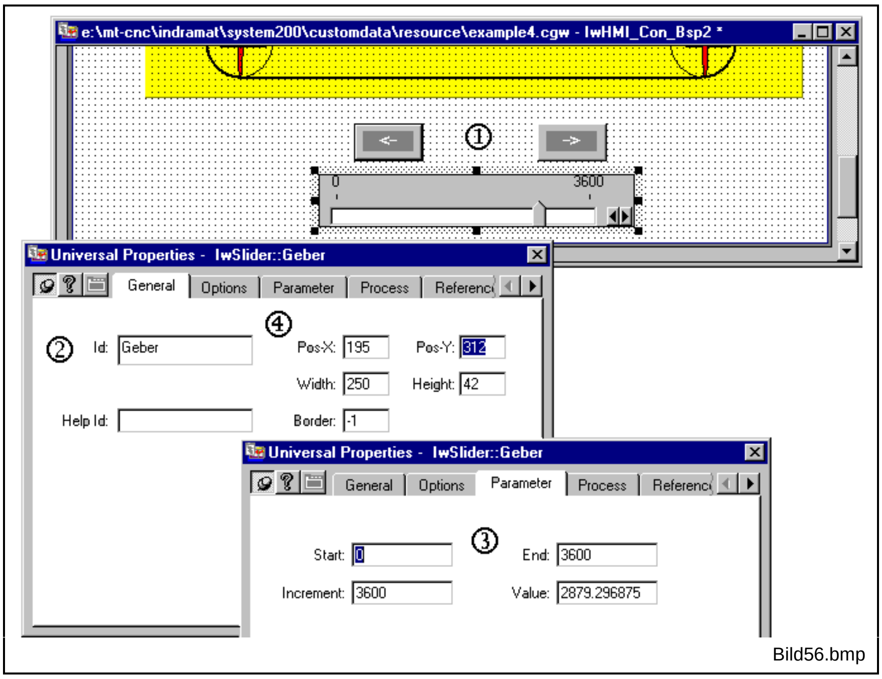
Fig. 6-5: Navigator with IwSlider

⇨ Double-click **IwSlider: Geber.**

The Properties window of IwSlider: Geber opens, see Fig. 6-6: IwSlider object (4).

⇨ Change the Y-position from –312 to 312.

IwSlider becomes visible, see Fig. 6-6: IwSlider object.


e:\mt-cnc\indramat\system200\customdata\resource\example4.cgw - IwHMI_Con_Bsp2 *
<-
->
①
0
3600
Universal Properties - IwSlider::Geber
General
Options
Parameter
Process
Referenc
②
③
④
Universal Properties - IwSlider::Geber

Id: Geber

Help Id:

Pos-X: 195

Pos-Y: 312

Width: 250

Height: 42

Border: -1

Start: 0

End: 3600

Increment: 3600

Value: 2879.296875

Bild56.bmp

Fig. 6-6: IwSlider object

(1) IwSlider object

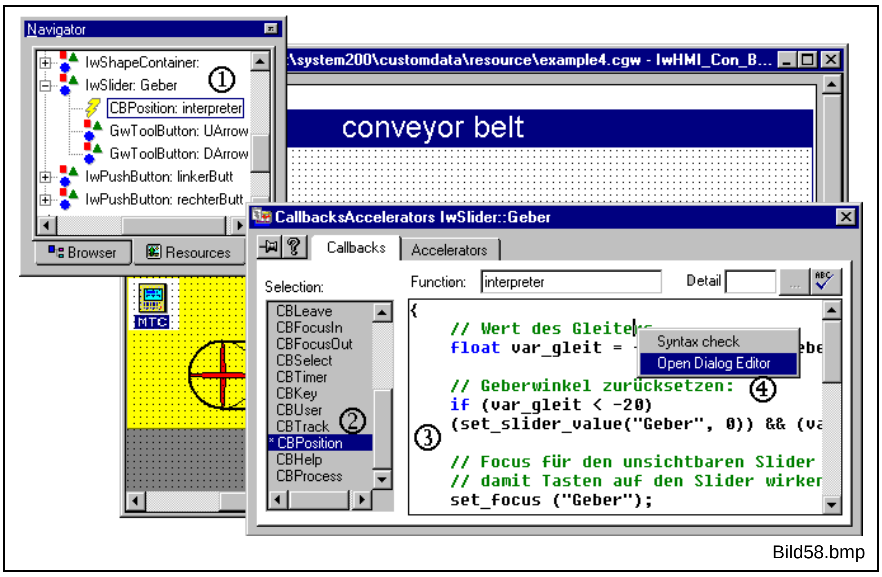
(2) Enter “Geber” (encoder) as ID.

(3) Define the slider parameters.

(4) By changing the position of the slider in the window, for instance, from PosY = 312 to PosY = - 312, the “Geber” slider is moved out of the visible range.

In the end, the IwSlider must again be moved to the invisible range.

“CBPosition” Callback



Function: interpreter

Detail

```
{
    // Wert des Gleite
    float var_gleit = -                              ebe

    // Geberwinkel zurücksetzen:
    if (var_gleit < -20)
    (set_slider_value("Geber", 0)) && (va

    // Focus für den unsichtbaren Slider
    // damit Tasten auf den Slider wirker
    set_focus ("Geber");
```

Bild58.bmp

Fig. 6-7: CBPosition of IwSlider

(1) Highlight “IwSlider: Geber”; press the right mouse button to open the context-sensitive menu; select “Callbacks&Accelerators”.

(2) Highlight “CBPosition”.

(3) Enter the callback below.

(4) The context-sensitive menu of the editor field can be used to open an editor dialog, which allows a more convenient program entry.

Complete CBPosition

```
{
        // Wert des Gleiters
        float var_gleit = -slider_value("Geber");

        // Geberwinkel zurücksetzen:
        if (var_gleit < -20)
        (set_slider_value("Geber", 0)) && (var_gleit=0);

        // Focus für den unsichtbaren Slider setzen,
        // damit Tasten auf den Slider wirken
        set_focus ("Geber");

        // Rotationswinkel setzen
        set_ellipse_rotation ( "Rad", 3*var_gleit);
        set_ellipse_rotation ( "Rad1", 3*var_gleit);
        set_ellipse_rotation ( "Rad2", 3*var_gleit);
        set_ellipse_rotation ( "Rad3", 3*var_gleit);

        set_ellipse_rotation ( "Speiche1_1", 1.57+var_gleit);
        set_ellipse_rotation ( "Speiche1_2", var_gleit);
        set_ellipse_rotation ( "Speiche2_1", 1.57+var_gleit);
        set_ellipse_rotation ( "Speiche2_2", var_gleit);

        // Positionsberechnung für die "MTCNC"
        float pos =0-28*var_gleit;
        set_position ("Mtcnc", pos, 54);
}
```

Further Items

The following picture shows those objects and callbacks which have not yet been mentioned:

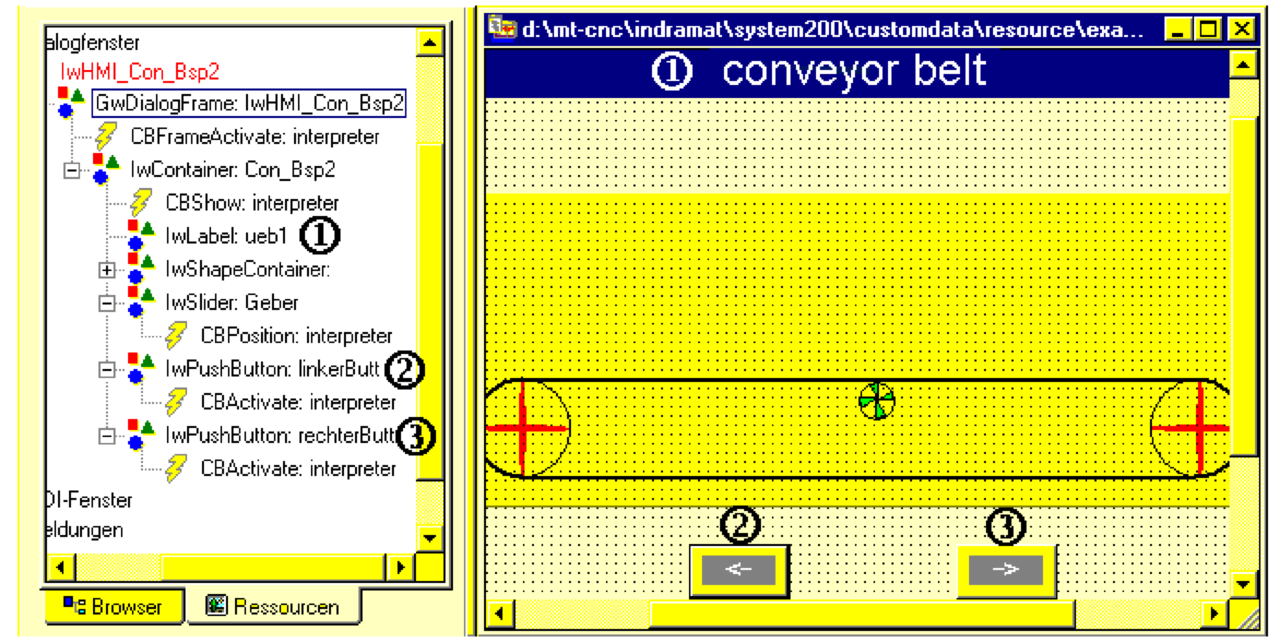


Bild59.bmp

Fig. 6-8: Further objects in the example

(1) IwLabel: ueb1
contains “conveyor belt” as text property

(2) IwPushButton: CursorLeft
contains the “plcedit_arrow_l.bmp” bitmap; apart from that, this button does not have any meaning.

(3) IwPushButton: CursorRight
contains the “plcedit_arrow_r.bmp” bitmap; apart from that, this button does not have any meaning.

The two bitmaps are a part of the INDRAMAT software and reside in the [Drive]:\mt-cnc\INDRAMAT\System200\BasicData\bitmap folder.

Both PushButtons have a “CBActivate” callback. The “GwDialogFrame” also contains the “CBFrameActivate” callback, and the “IwContainer” contains the “CBShow” callback. The content of the four callbacks is as follows:

```
{
set_focus ("Geber");
}
```

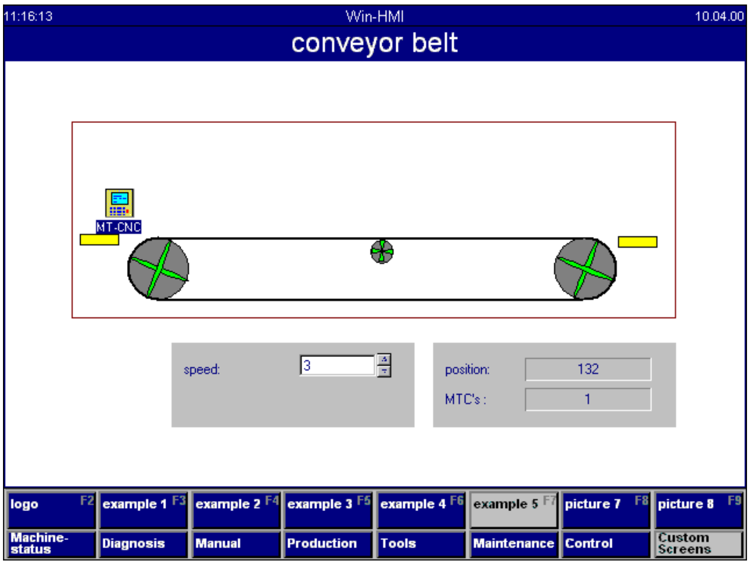
The set_focus method sets the focus to the “Geber” (encoder) object.

The focus can be moved by means of the cursor keys and the tab key. To ensure that the “Geber” remains always active, the method is used at those places where the focus can change.

6.2 Expanded Variant (Example 5)

As compared with “Example4”, “Example5” has been modified as follows:

- The command value encoder (Geber) for the movements is a timer.
- Two lifting devices have been added.
- The “IwExample5.igw” IGW file has been changed.



speed: 3

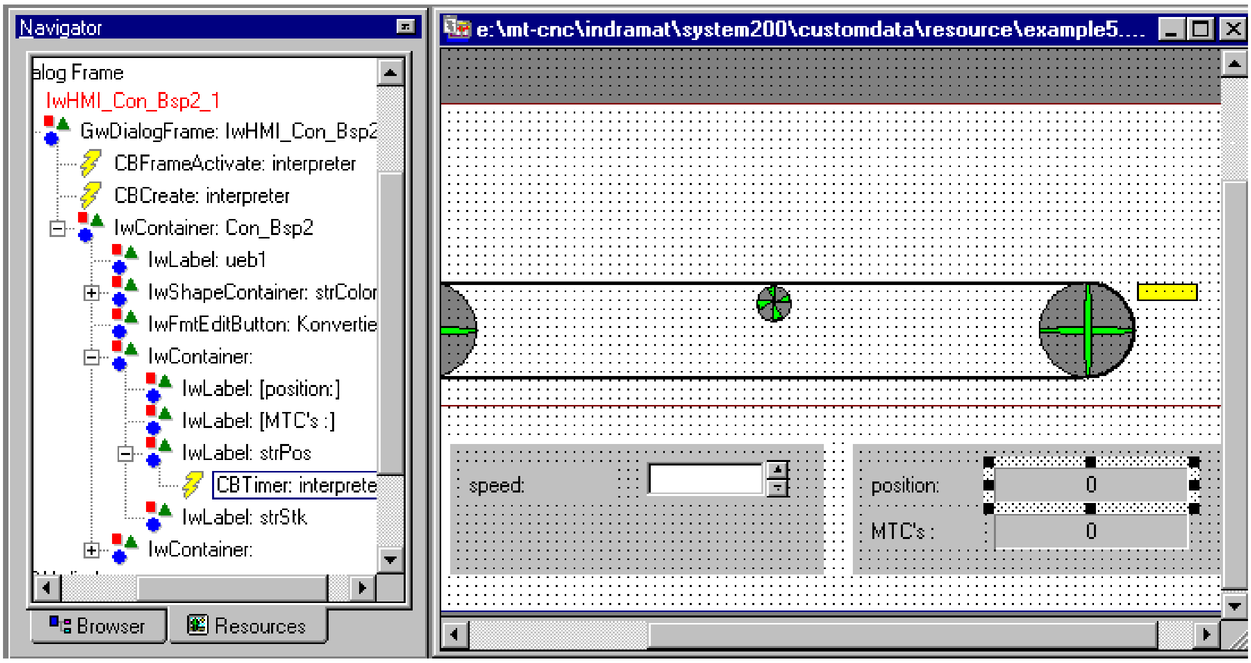
position: 132

MTC's : 1

Bild7K.bmp

Fig. 6-9: “Example5”

Object Overview


Navigator

alog Frame
IwHMI_Con_Bsp2_1
GwDialogFrame: IwHMI_Con_Bsp2
CBFrameActivate: interpreter
CBCreate: interpreter
IwContainer: Con_Bsp2
IwLabel: ueb1
IwShapeContainer: strColor
IwFmtEditButton: Konvertie
IwContainer:
IwLabel: [position:]
IwLabel: [MTC's :]
IwLabel: strPos
CBTimer: interprete
IwLabel: strStk
IwContainer:

e:\mt-cnc\indramat\system200\customdata\resource\example5....

speed:

position: 0

MTC's : 0

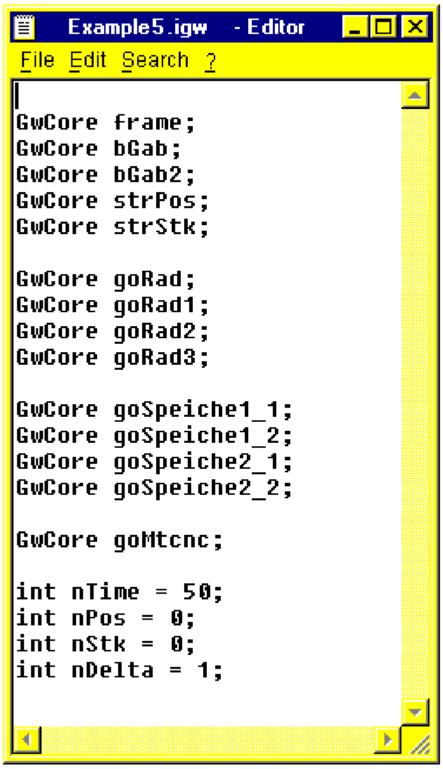
Bild61.bmp

Fig. 6-10: Object overview of “Example5”

“Example5.igw” IGW File

IGW files are text files which consist of a list of declarations and 4GL functions.

After an IGW file has been loaded for an application (CGW file), its functions are available in the entire application. In our example, this results in the advantage that the program is processed more rapidly.



```
GwCore frame;
GwCore bGab;
GwCore bGab2;
GwCore strPos;
GwCore strStk;

GwCore goRad;
GwCore goRad1;
GwCore goRad2;
GwCore goRad3;

GwCore goSpeiche1_1;
GwCore goSpeiche1_2;
GwCore goSpeiche2_1;
GwCore goSpeiche2_2;

GwCore goMtcnc;

int nTime = 50;
int nPos = 0;
int nStk = 0;
int nDelta = 1;
```

Bild62.bmp

Fig. 6-11: “IwExample5.igw” opened in the text editor

The file must be made known to the application. This can be achieved by means of #require “filename”.

The #require directive is used to load an IGW file at the beginning of file modules and on the topmost level, e.g. within the Create callback of the application window.

“CBCreate” callback

```
#require "Example5.igw"

{

bGab = child( this, "IwGabel2" );
bGab2 = child( this, "IwGabel" );
strStk = child( this, "strStk" );
strPos = child( this, "strPos" );

goRad = child( this, "Rad" );
goRad1 = child( this, "Rad1" );
goRad2 = child( this, "Rad2" );
goRad3 = child( this, "Rad3" );

goSpeiche1_1 = child( this, "Speiche1_1" );
goSpeiche1_2 = child( this, "Speiche1_2" );
goSpeiche2_1 = child( this, "Speiche2_1" );
goSpeiche2_2 = child( this, "Speiche2_2" );

goMtcnc = child( this, "Mtcnc" );
}
```

Time Control

Starting the timer: CBFrameActivate

```
{
        if (event("active")==TRUE)
        {
                start_timer( strPos, nTime );
        }
        else
        {
          stop_timer( strPos );
        }
}
```

The timer should only be active if the “Example” application is active. If a different application is active, the conveyor belt stops running.

The cycle time of the timer is set by means of the “nTime” parameter. This parameter is set to nTime = 50 in the IGW file. There, the time interval can be changed.

“CBTimer” callback

```
{
        nPos = nPos + nDelta;

        //Geberwinkel zurücksetzen:
        if ( nPos > 700 )
        {
                nPos = 0;
                nStk = nStk + 1;
                strStk.set_text( nStk );
        }

        strPos.set_text( nPos );

        //Farbe
/*      float color = nPos/5;
        float col1 = 127 + color;
        if ( col1 > 255 )
                col1 = 127;

        string strcol = make_rgb( col1, col1, 0 );
        set_background( bGab, strcol );
        set_background( bGab2, strcol );*/

        // Rotationswinkel setzen
        float dR1 = ellipse_rotation( goRad );
        dR1 = dR1 - (0.066*nDelta);
```

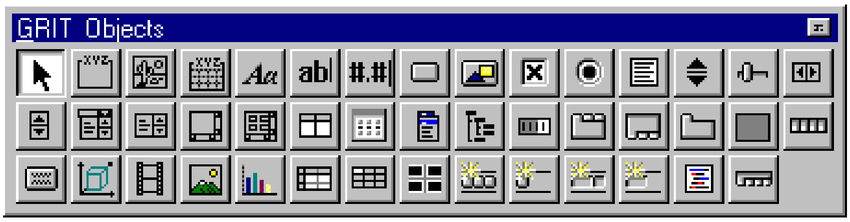
```
set_ellipse_rotation ( goRad, dR1);
set_ellipse_rotation ( goRad1, dR1);
set_ellipse_rotation ( goRad2, dR1);
set_ellipse_rotation ( goRad3, dR1);

float d1 = ellipse_rotation( goSpeiche1_1 );
float d2 = ellipse_rotation( goSpeiche1_2 );
d1 = d1 - (0.0211*nDelta);
d2 = d2 - (0.0211*nDelta);

set_ellipse_rotation ( goSpeiche1_1, d1);
set_ellipse_rotation ( goSpeiche1_2, d2);
set_ellipse_rotation ( goSpeiche2_1, d1);   //d3
set_ellipse_rotation ( goSpeiche2_2, d2);   //D4

//Positionsberechnung
//y-Bewegung
if ( nPos < 118 )
{
        int xMTCNC = 2;
        int xGabel = 7;
        int y1 = nPos;

        set_position ( goMtcnc, xMTCNC, 170 - y1 );
        set_position ( bGab , xGabel, 214 - y1 );
}
//x-Bewegung MT-CNC
if ( nPos >= 118 && nPos <= 581 )
{
        int x1 = nPos + 2 - 118;
        set_position ( goMtcnc, x1, 100-45 );
}
//Y-Bewegung Gabel nach unten
if ( nPos >= 168 && nPos < 385 )
{
        int xGabel = 7;
        int y1 = nPos-168;

        set_position ( bGab, xGabel, 97 + y1 );
}

if ( nPos > 581 && nPos < 700 )
{
        int xMTCNC = 465;
        int xGabel = 470;
        int y1 = nPos-581;

        set_position ( goMtcnc, xMTCNC, 56 - y1 );
        set_position ( bGab2, xGabel, 101 - y1 );
}
if ( nPos < 135 )
{
        int xGabel = 470;
        int y1 = -33 + nPos;
```

```
                set_position ( bGab2, xGabel, y1 );
        }
}
```

7 Appendix: Available Toolbars

7.1 Toolbars: GRIT-Objects

The following objects in the toolbar "GRIT-Objects" are available for general and static applications:



Toolbar GRIT-Objects.bmp

Fig. 7-1: Toolbar: GRIT-Objects

These objects can be generated in a resource file via Drag&Drop or Point&Click. Due to the variety of objects they were divided into functional groups:

- Background Objects,
- Control Elements,
- Table and List Objects and
- Other Input/Output Objects

Note: The prefix „Gw/Iw" before the name of an object indicates that a generated object pertains to an instance of a class of the same name of the GRIT/Indramat-class library. The online documentation "GRIT Application Developer" (Datei: contains a detailed description of the objects. main_wb.hlp).

Background Objects

A base element is required for each application, on which other objects can be placed. A background object and all objects placed thereon create a "father-child" relationship. The background objects are created as instances of the following classes.

Symbol	Class	Meaning
	IwContainer	Container are background objects on which many other objects can be placed on arbitrarily. As first object This determines the size of the application window.
	IwDrawingArea	The DrawingArea is a drawing surface, on which e.g. lines, rectangles, circles etc. can be drawn out from the application program.
	IwRowColumn	The RowColumn is a background object with the special property that allows all children of this object to be automatically aligned in lines and columns. Additionally a height and width adjustment is performed, which is depends on the highest and widest object of the line or column. The children can be downsized afterwards.
	IwDrawnPushButton	The graphical possibilities which the DrawnPushButton offers are the same as with the DrawingArea. Additionally this object can be selected as a normal PushButton.
	IwScrolledWindow	The ScrolledWindow is a window with a scroll bar. Exactly one child-object (e.g. IwContainer) is possessed. The size of this child-object determines if a scroll bars are shown. All objects that lay on a ScrolledWindow can be scrolled into the visible

		sector via the scrollbars of the ScrolledWindow.
	IwScrolledObject	This object can carry a background object, of which several stamps can be shown and can be addressed separately. The ScrolledObject is a special form of the ScrolledList. In opposite to the ScrolledList, which can only contain strings respectively texts, the ScrolledObject is capable in picking up any desirable GRIT/IW-object. These can also be objects that possess children themselves. Consequently, it is possible to even handle complex entries. A ScrolledObject can contain desirable many entries, but a specific quantity is only visible. The quantity of the total entries as well as the quantity of the visible entries can be indicated. To align the objects a selection between a horizontal and vertical scrollbar is available. The distance between the individual objects can be determined (in Pixel).
	GwMenuBar	This object functions as a background object for the creation of a menu line. Items, SubItems, ToggleItems and Separators can be used as children (menu components).
	IwTree	A tree-object to show desirable tree structures. It performs as father object for the so called TreeEntries, which forms the entries in the tree. If the quantity of visual entries exceeds the size of the tree, then the horizontal respectively the vertical scrollbars appear automatically. A „+" beside a tree entry shows that this possesses children. A tree entry is opened by clicking on the „+" or by double clicking on an assigned text respectively folder symbol of a tree entry. If a tree entry is opened then the children are not visible and the „+" is replaced by a „-". Click on „-" to close the branch of a tree. The right and the left cursor arrow key can also be used to open or close the tree entries. The upper and the lower cursor arrow key moves the marker that highlights the selected entry of the list of visual entries.
	IwTabControl	A TabControl offers the same function as a TabContainer. The difference in comparison with the TabContainer is, that the tabs are located at the top edge. If the width of all tabs of the TabPages exceed the width of the TabControls, then two buttons for paging appear automatically.
	IwTabContainer	The TabContainer is comparable with an index card box, the TabContainer is the box and the TabPages are the index cards (register cards). If the width of all tabs of the TabPages exceed the width of the TabControls, then two buttons for paging appear automatically.
	IwTabPage	A TabPage is a single page on a TabContainer respectively TabControl, which can contain any desirable GRIT/Indramat-objects. The size of a TabPage results from the size of the TabControls. A TabPage possesses a tab. The page is activated by clicking the tab (set as upper page). The TabPage can not have a focus. But if the TabControl has the focus, then it is shown on the tab of the active page.
	IwMDIContainer	The MDIContainer functions as a background object for the MDI-window. The father object window must contain a MDIContainer to create a MDI-application.
	IwPanel	A IwPanel derives from an IwContainer and manages self contained the object types IwButtonEx or IwFKeyEx. A IwPanel conducts within the Application Builder like „an object". A double click on the buttons results in calling up the Propertysheets of the IwPanel class. There the properties of the panels are defined, thus dimension, container text, background bit map, attachment, edges and so on. But also the properties of the buttons are defined via the IwPanelpropertysheet.
	IwHPGL	This object is to show HPGL-files (**H**ewlett Packard **G**raphic **L**anguage-Format). Either, it can be drawn on out from an application, or a HPGL-graphic can be shown on it.
	IwMetaFile	This object is to show files in Windows-Metafile-Format. Either, it can be drawn on out from an application, or a HPGL-graphic can be shown on it.
	IwMedia	Videos (AVI-files) can be tied in and shown with the assistance of a object in the GwMedia class.
	IwToolBar	To align the ToolButtons a toolbar functions as background and father object. A symbol bar is normally shown either under the MenuBar or in a DockingFrame. The position of children within the symbol bar depends on their dimension.
	IwStatusBar	A StatusBar is to show information and messages that should be given to the user during the run time. It appears at the lower edge of an application window.
	IwViewSelector	The ViewSelector is an object that can contain of several pages. Each page possesses a tab. In comparison with TabControl / TabContainer, ViewSelector has no children.

Fig. 7-2: Toolbar GRIT-Objects: Background Objects

Control Elements

As desired you can place control elements on a window or background object. They are to handle the application. In a menu bar the objects Item, SubItem, ToggleItem, and Separator can be generated to children by defining its type in the attribute dialog of the menu entry. A general menu entry is automatically created when an object of a MenuBar class is generated.

Symbol	Class	Meaning
ab|	IwEditButton	This object reassembles a single line entry pad. All Clipboard-methods of the Windows manager are available on this pad. Children can not be created on the EditButtons.
#.#|	IwFmtEditButton	The formatted entry pad consists of a prompt, the actual entry pad and an appendix. An entry mask can be predetermined for the entry section. Based on it a plausibility self check (e.g. entry of the date) is performed on the object.
▭	IwPushButton	A PushButton is a command switch template of the user template with text or a bit map. Switch templates can be child-objects of background objects (lwContainer, lwRowColumn). Objects of the IwPushButton class can not possess child-objects themselves.
☒	IwToggleButton	A ToggleButton matches a RadioButton. In comparison with the RadioButton, several ToggleButtons can be selected within a group.
◉	IwRadioButton	This button has exactly two conditions – on or off. The value is shown through the marking before text of the RadioButton. Of all RadioButtons in a group only one can be turned on.
⬍	IwSpinButton	The SpinButton derives from the IwScrolledList and therefore offers its functionality. SpinButtons create like the RadioButtons a group, meaning the objects have the same father. One of the SpinButtons is the master. The master has two direction arrows, which, when activated, triggers a rotation of the entry list of the selected SpinButton in the group. SpinButtons can be set sequentially to e.g. realize a date entry.
-0-	IwSlider	The slider reassembles a slide controller. An object of the IwSlider class possesses a slider and two direction arrows. The sliding range is adjusted via a beginning, end and an increment value. The orientation of the slider can be horizontal or vertical. The slide controller works continuously or discrete. The resolution results from the pixel size.
◂▸	IwScrollbarHorizontal	The Scrollbar reassembles a scrollbar. The sliding range, within the slider can be moved, is defined by a start and an end value. The possibility exists to set the orientation of the Scrollbar horizontally or vertically. The slide controller works continuously or discrete. The resolution results from the pixel size.
▴▾	IwScrollbarVertikal	
☰	IwComboBox	This object reassembles a combination of EditButton and ScrolledList. The advantage of this object is, that a selection list, which is linked to the object, can be opened. The quantity of entries of a ComboBox is of desirable size, thus only a limited number of lines are visible. The quantity can be determined and can therefore be changed. The size of the ComboBox will be adjusted accordingly. Furthermore the possibility exists to edit the active entry. If the entered characters in the edit line match exactly the beginning character set of an entry in the list of the ComboBox, then this line of the list is automatically selected and the entry in the edit line is expanded by the rest of the character set. The expansion of the edit line is shown “marked”. If no respective line is found in the list, then the “marked” text of the edit line is replaced by the entered characters and the ComboBox, should it have been in edit mode before, is switched into the normal edit mode. The individual selected lines are deselected.
▭	IwButton	The IwButton can show two texts, a background bit map and an additional bit map. The IwButton has inherited all properties of its predecessors, but versus the IwButtonEx it can be installed independently. It can not become a Default-Button.
▦	IwValueSet	The ValueSet is an object, which can show entries as text, bit map and/or color in matrix form. The entries can function as switch templates as well. As of Application

		Builder 01V06 it will be relieved by the objects IwToolBar / IwToolButton.
	IwToolButton	The ToolButton is an object, which can possess the functions of a PushButton or a ToggleButton. It is mainly used on a ToolBar.

Fig. 7-3: Toolbar GRIT-Objects: Control Elements

Tables and List Objects

Object	Class	Meaning
	IwScrolledList	ScrolledLists shows a list of text lines. A vertical scrollbar appears, if more lines are contained in the list as momentarily can be shown due to the size of the object. Is an entry longer than the width of the ScrolledList, then a horizontal scrollbar appears. ScrolledLists can not have any children.
	IwScrolledTabList	This object is a multiple column scrollable text list, in which the column headlines can be set. Otherwise it reacts analog to the ScrolledList. The titles of the columns remain when scrolling.
	IwMultiColumnScrolledList	The MultiColumnScrolledList belongs to the multiple column list objects. The contents of each column can be a bit map or a text. The end user can sort the lines of the table according to a desirable column. The titles of the columns remain when scrolling.
	IwSpreadSheet	The spreadsheet is to show information in a line – column form. Furthermore spreadsheet calculations can be performed quickly and easily with these GRIT-objects.

Fig. 7-4: Toolbar GRIT-Objects: Table and List Objects

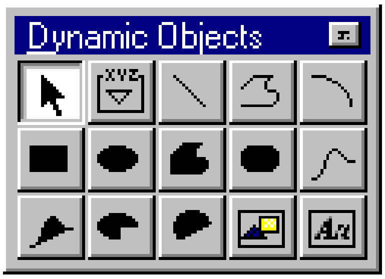
Other Input/Output Objects

Symbol	Class	Meaning
Aa	IwLabel	A label is only to show texts or a bit map and can not be selected.
▤	IwText	A text field symbolizes a multiple line entry template. An automatic line return and a scrolling of the text is performed in addition to the clipboard functions.
▭	IwProgressIndicator	In the form of an analog bar the progress of any process can be shown with a progress indicator. The process progress is either shown in percent or as an absolute value (e.g. Bytes). The start and end range can be defined freely as well as the position of the text and the color of the bar and its background.
📊	IwChart	This object is to show information that are in line and column form, in the form of a pie, bar or line diagram.
▭	IwStatusBarEntry	Similar to a label information can be given with a StatusBarEntry as a child of a StatusBar, but its variety of functions is broader than those of a label.
▤	IwTextEditor	A text editor field allows the entry of texts, syntax-oriented highlighting, undo/redo and further syntax-depending keyboard and mouse commands of known text editors.
▦	IwEdit	

Fig. 7-5: Toolbar GRIT-Objects: Other Input/Output Objects

7.2 Toolbar Dynamic Objects

The following objects are available in the toolbar "Dynamic Objects" for dynamic applications:



Toolbar_DynamischeObjekte.bmp

Abb. 7-6: Die Toolbar: Dynamische Objekte

These objects can be created in a resource file via Drag&Drop or Point&Click. They can be divided into two functional groups:

- Dynamic Container Classes and
- Graphic Base Objects

Note: The prefix „Gw/Iw" before the name of an object indicates that a generated object pertains to an instance of a class of the same name of the GRIT/Indramat-class library. The section “Dynamic Graphic Objects” of the online documentation contains a more detailed description of the objects.

Dynamic Container Classes

Functionally containers correspond with those of the GRIT-container classes. These function as background objects for any type of shape graphic object, but can also contain all GRIT-control elements (GwControl or IwControl).

Symbol	Class	Meaning
	IwShapeContainer	Background object for any type of dynamic graphic object. Base class of all container objects.

Fig. 7-7: Toolbar Dynamic Objects: Container Classes

Graphical Base Objects

In desirable quantities graphical base objects such as lines, rectangles, eclipses and others can be placed in one of the container objects and can be worked on individually or together.

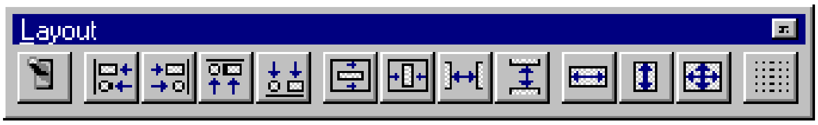
With the assistance of the toolbar dynamic objects of the dialog editor a large quantity of objects can be created in the resource file via Drag&Drop or Point&Click. In this case the symbol of the toolbar is given in the overview.

Symbol	Class	Meaning
╲	IwShapeLine	Represents a simple line which connects two points.
∽	IwShapePolyline	A row of 2 or more points which are connected with each other through straight lines. The line is not closed and not filled.
◝	IwShapeEllArc	An eclipse bow which is defined by its center point, its width and height, the start and end angle, and the actual rotation around its center point.
■	IwShapeRect	Rectangle A rectangle is internally defined by its upper left corner, its width and height, and the actual rotation around the upper left corner.
⬬	IwShapeEllipse	Eclipse The eclipse is defined by its center point, its width and height, and the actual rotation around its center point. A stretching of the eclipse is not possible.
⬟	IwShapePolygon	A closed row of 2 or more points which are connected with each other through straight lines. The line is not closed and not filled. A connecting line is automatically placed between the points if the first and the last point are not conform.
▬	IwShapeRoundRect	Rectangle with rounded corners. The parameters of the rectangle are determined as in the IwShapeRect class. The rounding grade of the corners of the to be drawn eclipse is additionally determined in the form of the X and Y-radius.
∫	IwShapeBezier	A row of cubical bezier splines which are not closed. A cubical bezier spline is defined by its start point, two control points and its end point. With a bezier curve the end point of a spline is conform with the start point of the following spline. The bezier curve is not filled.
➤	IwShapeClosedBezier	Closed row of cubical bezier splines. In opposite to the IwShapeBezier this bezier curve is closed and can be filled optionally. The start point of the first spline is at the same time the endpoint of the last spline.
◕	IwShapeEllPie	An eclipse bow where both end points are individually connected with the eclipse center point. The eclipse segment can optionally be filled.
◖	IwShapeEllChord	An eclipse bow where both end points are connected with each other through a straight line. The eclipse segment can optionally be filled.
🖼	IwShapeBitmap	This bit map object is to incorporate one of the IwBitmap supported file formats. Should a bit map object be drawn transparent or not can be predetermined. Should it be drawn transparent then the color for the to be created transparency mask can either be predetermined explicitly or the color of the 1 pixel (upper left corner) is used as transparency color.
Aa	IwShapeText	Text object consisting of one or several text lines (separated by linefeed). The text position can be to the left, to the right or centered. The surrounding rectangle is filled with a non transparent background. Is the text rotated, then the rotated surrounding rectangle is filled (not the actual horizontally and vertically aligned surrounding rectangle).

Fig. 7-8: Toolbar Dynamic Objects: Graphical Base Objects

7.3 Toolbar: Creation

The toolbar creation is to align and adapt the size of all objects (base objects as well as dynamic objects) in the dialog editor. The functions of the toolbar are also accessible via the functions of the menu creation.



Toolbar_Gestaltung.bmp

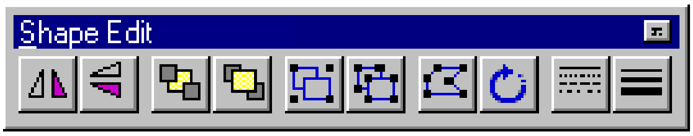
Fig. 7-9: The Toolbar “Creation”

Symbol	Meaning
	Starts a preview of the active window.
Sorts the selected objects (minimum 2) to the height of the:	
	- left edge of the reference object.
	- right edge of the reference object.
	- upper edge of the reference object.
	- lower edge of the reference object.
Sorts the selected objects (minimum 3) with the same distance between the:	
	- outer left and right object.
	- highest and lowest object.
Centers the object/object group in correlation to its father object in:	
	- horizontal direction.
	- vertical direction.
Adapts the selected objects (minimum 2) to the:	
↔	- width of the reference object.
↕	- height of the reference object.
✥	- width and height of the reference object.
	Switches a grid on/off.

Fig. 7-10: Symbols of the Toolbar “Creation”

7.4 Toolbar: Form Settings

The toolbar “Form Settings” is to exclusively manipulate dynamic objects.



Toolbar_FormenBearbeiten.bmp

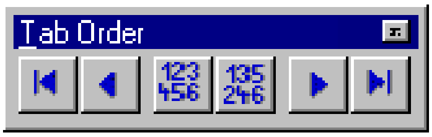
Fig. 7-11: The toolbar: “Form Settings”

Symbol	Meaning
◮	Mirrors a dynamic object on its own vertical axis.
◀	Mirrors a dynamic object on its own horizontal axis.
⧉	Sorts a dynamic object in the background.
⧉	Sorts a dynamic object in the foreground.
⊞	Combines all selected objects to a group.
⊞	Dissolves the combined group of objects.
⌗	Switches the conversion mode on/off. Individual points of dynamic objects can be moved in the conversion mode and therewith the forms of the objects can be changed.
↻	Is to rotate steplessly the dynamic objects.
┅	Selection of a line type.
≡	Selection of a line width .

Fig. 7-12: Symbol of the Toolbar: “Form Settings”

7.5 Toolbar: Tab Sequence

You can change the sequence of the objects in your child list with the support of the toolbar “Tab Sequence”. For the latter user this is of importance, because the sequence in the child list dictates the tab sequence.



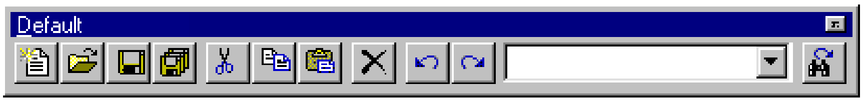
Toolbar_Tabulator-Reihenfolge.bmp

Fig. 7-13: The Toolbar “Tab Order”

Symbol	Meaning
⏮	First position; within its child list the object is moved to the first position (zero).
◀	Position – 1; within the child list the object is moved up by one position, e.g. position 3 becomes position 2.
123 456	Sorting horizontally; all child objects of one father object are newly numbered according to its horizontal order in the window beginning with the left. (The common father object must be selected for this function.)
135 246	Sorting vertically; all child objects of one father object are newly numbered according to its vertical order in the window beginning at the top. (The common father object must be selected for this function.)
▶	Position +1; within the child list the object is moved down by one position, e.g. position 3 becomes position 4.
⏭	Last position; within its child list the object is moved to the last position.

Fig. 7-14: Symbols of the Toolbar “Tab Order”

7.6 Toolbar: Standard



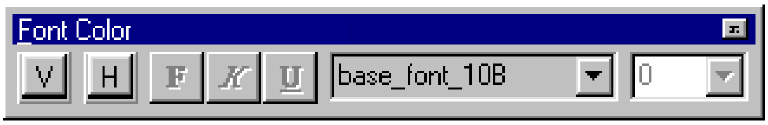
Toolbar_Standard.bmp

Fig. 7-15: The Toolbar “Standard”

Symbol	Meaning
🗋	Create a new resource file.
📂	Open an existing resource file.
💾	Store resource file. Have you opened a window in the dialog editor of the workbench, then only store the actual opened dialog window with the menu point store of the menu file.
🗐	Store all. To store all opened files at the same time.
✂	Cut out. Key combination: CTRL+X or SHIFT+DEL
⧉	Copy. Key combination: CTRL+C or CTRL+INS
📋	Insert. Key combination: CTRL+V or SHIFT+INS
✕	Delete.
↶	Undo editing function. To undo a step.
↷	Redo editing function. To repeat an undone step.
🔍	Search. Write the to be searched text into the combobox “Enter search text” of the toolbar and press the search button.

Fig. 7-16: Symbols of the Toolbar “Standard”

7.7 Toolbar: Font Color



Toolbar_Schriftfarbe.bmp

Fig. 7-17: The Toolbar “Font Color”

Symbol	Meaning
V	Change foreground color.
H	Change background color.
	Font creation: Bold
	Font creation: Italic
	Font creation: Underline
base_font_10B	Font-Family: Font Type
	Font-Height Font Size

Fig. 7-18: Symbols of the Toolbar “Font Color”

8 Glossary

Application Window

The main window of every application is a application window.

From this window sub dialog windows can be called up. An additional dialog can be opened from a dialog window. Additional pop up menus and standard messages are available as window objects. These windows function as background objects for any type of desktop object.

Every application requires minimum one application window. It appears immediately after the start of the application and forms the root for the object hierarchy of the application.

Callback-Event

Specific events are appointed to each GRIT-desktop object, which can derive from the user or the system, e.g. the activation of a button object via a mouse click or RETURN. A so called callback function can be consigned to the application for the appearance of each of these callback events, which is performed during the run time, as soon as the event appears.

Callback-Function

A function, which is performed during the run time with the appearance of a callback event. The so called callback functions can be consigned to each desktop object depending on an awaited or wanted callback event. The function can be written in GRIT-4GL or C/C++.

CGW-File

A binary file (resource file) which contains the parameters for the user desktop. This file can be tied in an C++ application.

Dialog Editor

An editor to create GRIT-application desktops in a WYSIWYG principle. The dialog editor is a component of the GRIT-workbench.

Dialog Window

Applications with a graphical user desktop utilize for the input and output of information of temporary blended in windows, the so called dialog window. A classical example for this is the dialog window to open files. Fundamentally a differentiation between two types of dialog windows is made. mode and modeless See modality

GRIT-Object

A desktop object of a GRIT-application desktop, which is set as an instance of a GwCore derived class. GRIT-objects can be created simply in the dialog window (WYSIWYG).

GRIT-Workbench

see Workbench

GwCore

A base class of all GRIT-desktop objects.

IGW-File

A text file, in which the 4GL-functions are declared and defined. After file log in these functions can be called up from a callback function in the gw.ini or with #require.

MDI (Multi Document Interface)

Word processing programs are a good example to explain the functional principles of MDI applications. While the word processing program itself appears in a main window (MDI frame window), each document to be processed is shown in its own window (MDI child window). This allows parallel processing of documents. In addition, the simultaneous view of several documents facilitates the rapid data exchange by means of the clipboard or the drag-and-drop function (see MDI window). (see MDI-Window)

MDI-Window

Three different window types form part of MDI applications:

MDI Frame Window: This window is the main window (frame) of the application. It is the parent window of the MDI client window.

MDI Client Window: This window is the “invisible” instance between the MDI frame window and the MDI child windows. The exchange of data between the main window and the child windows is exclusively processed via the MDI Client window.

MDI Child Window: These windows are children of the MDI client window and, thus, the grandchildren of the MDI frame window. They contain the documents to be managed.

Messages

Messages can be shown in a message window. A message window is a special form of dialog window. This is to inform the user of warnings, messages or notes. As a standard it holds a bit map, a text and contains a row of buttons.

Modality

Modality windows lock input operations on called up application windows. Therewith the dialog window dominates the interaction text so long until it is closed.

Modeless dialog windows are used for the case that a dialog window should remain open for longer period of time. This window type also allows a further processing of the called up application window. Modeless dialog windows therefore exist beside the application window in order to in e.g. visualize a constant changing condition or to save user efforts to having to often call up the dialog window. A typical example is the searching for text fragments in a document.

Popups

Popups are self contained menus which can be shown at any desirable position of the window, e.g. upon a mouse click at the actual position of the mouse arrow. They are very often used to start actions that are linked to the object through which they were activated.

Resource File

see CGW-File

Workbench

The GRIT-workbench, in short named workbench, combines all for the software development process required GRIT-modules in a unified desktop. To this belong different editors, a project manager, a unified properties dialog as well as options dialog. Furthermore the GRIT-workbench contains an integration of compiler and link tools for all GRIT supported platforms, so that you can compile and process all applications directly via the GRIT-workbench.

WYSIWYG

What You See Is What You Get
A creation principle where - according to the looks - the actual working object is identical the end product.

9 List of Figures

10 Index

11 Kundenbetreuungsstellen - Sales & Service Facilities

Deutschland – Germany

vom Ausland:	(x) nach Landeskennziffer weglassen!!
from abroad:	don't dial (x) after country code!

Vertriebsgebiet Mitte ☒ SALES Germany Centre ☒ Service Rexroth Indramat GmbH Bgm.-Dr.-Nebel-Str. 2 97816 Lohr am Main Telefon: +49 (0)9352/40-0 Telefax: +49 (0)9352/40-4885 **Kompetenz- Zentrum Europa**	**SERVICE** **CALL ENTRY CENTER** **MO – FR von / from 7 – 18** **Tel. +49 (0) 9352 40 50 60** service@indramat.de	**SERVICE** **HOTLINE** **MO – FR von / from 17 - 07** + SA / SO **Tel.: +49 (0)172 660 04 06** **oder / or** **Tel.: +49 (0)171 333 88 26**	**SERVICE** **ERSATZTEILE / SPARES** verlängerte Ansprechzeit: - extended office time - ♦ nur an Werktagen - only on working days - ♦ von 15 -18 Uhr - from 15-18 o'clock - **Tel. +49 (0) 93 52/40 42 22**
Vertriebsgebiet Süd ☒ SALES Germany South ☒ Service Rexroth Indramat GmbH Ridlerstraße 75 80339 München Telefon: +49 (0)89/540138-30 Telefax: +49 (0)89/540138-10 indramat.mue@t-online.de	Gebiet Südwest ☒ SALES Germany South-West ☒ Service Mannesmann Rexroth AG Vertrieb Deutschland – VD-BI Geschäftsbereich Rexroth Indramat Regionalzentrum Südwest Ringstrasse 70 / Postfach 1144 70736 Fellbach / 70701 Fellbach Tel.: +49 (0)711/57 61–100 Fax: +49 (0)711/57 61–125	Vertriebsgebiet Ost ☒ SALES Germany East ☒ Service Rexroth Indramat GmbH Beckerstraße 31 09120 Chemnitz Telefon: +49 (0)371/35 55-0 Telefax: +49 (0)371/35 55-333	Vertriebsgebiet Nord ☒ SALES Germany North ☒ Service Mannesmann Rexroth AG Vertriebsniederlassung Region Nord Gesch.ber. Rexroth Indramat Walsroder Str. 93 30853 Langenhagen Telefon: +49 (0) 511/72 66 57-0 Telefax: +49 (0) 511/72 66 57-93
Vertriebsgebiet West ☒ SALES Germany West ☒ Service Mannesmann Rexroth AG Vertrieb Deutschland Regionalzentrum West Borsigstrasse 15 D - 40880 Ratingen Telefon: +49 (0)2102/409-0 Telefax: +49 (0)2102/409-406	Vertriebsgebiet Mitte ☒ SALES Germany Centre □ Service Mannesmann Rexroth AG Gesch.ber. Rexroth Indramat Lilistraße 14-18 63067 Offenbach Telefon: +49 (0) 69/82 00 90-0 Telefax: +49 (0) 69/82 00 90-80	Vertriebsgebiet Ost ☒ SALES Germany East □ Service Mannesmann Rexroth AG GB Rexroth Indramat GmbH Holzhäuser Str. 122 04299 Leipzig Telefon: +49 (0)341/86 77-0 Telefax: +49 (0)341/86 77-219	Vertriebsgebiet Nord ☒ SALES Germany North □ Service Rexroth Indramat GmbH Kieler Straße 212 22525 Hamburg Telefon: +49 (0)40/85 31 57-0 Telefax: +49 (0)40/85 31 57-15

Kundendienstniederlassungen in Deutschland - Service agencies in Germany

Europa – Europe

vom Ausland: (x) nach Landeskennziffer weglassen, 0 nach Landeskennziffer mitwählen (Italien)!
from abroad: don't dial (x) after country code, dial 0 after country code (Italy)!

Austria ☒ SALES ☐ Service
Mannesmann Rexroth Ges.m.b.H.
Gesch.ber. Rexroth Indramat
Hägelingasse 3
A - 1140 Wien
Telefon: +43 (0)1/9852540-400
Telefax: +43 (0)1/9852540-93

Austria ☒ SALES ☒ Service
Mannesmann Rexroth G.m.b.H.
Gesch.ber. Rexroth Indramat
Industriepark 18
A - 4061 Pasching
Telefon: +43 (0)7221/605-0
Telefax: +43 (0)7221/605-21

Belgium ☒ SALES ☒ Service
Mannesmann Rexroth N.V.-S.A.
Gesch.ber. Rexroth Indramat
Industrielaan 8
B-1740 Ternat
Telefon: +32 (0)2/5830719
Telefax: +32 (0)2/5830731
E-mail: indramat@rexroth.be

Denmark ☒ SALES ☒ Service
BEC AS
Zinkvej 6
DK-8900 Randers
Telefon: +45 (0)87/11 90 60
Telefax: +45 (0)87/11 90 61

Czech Rep. ☒ SALES ☐ Service
Mannesmann-Rexroth, spol.s.r.o.
Hviezdoslavova 5
CS - 627 00 Brno
Telefon: +420 (0)5/48 126 358
Telefax: +420 (0)5/48 126 112

England ☒ SALES ☒ Service
Mannesmann Rexroth Ltd.
Rexroth Indramat Division
Broadway Lane, South Cerney
GB - Cirencester, Glos GL7 5UH
Telefon: +44 (0)1285/863000
Telefax: +44 (0)1285/863030

Finland ☒ SALES ☐ Service
Rexroth Mecman Oy
Rexroth Indramat division
Ansatie 6
SF-017 40 Vantaa
Telefon: +358 (0)9/84 91-11
Telefax: +358 (0)9/84 91-13 60

France ☒ SALES ☒ Service
Mannesmann Rexroth S.A.
Division Rexroth Indramat
Parc des Barbanniers
4, Place du Village
F-92632 Gennevilliers Cedex
Telefon: +33 (0)141 47 54 30
Telefax: +33 (0)147 94 69 41
Hotline: +33 (0)6 08 33 43 28

France ☒ SALES ☐ Service
Mannesmann Rexroth S.A.
Division Rexroth Indramat
270, Avenue de Lardenne
F - 31100 Toulouse
Telefon: +33 (0)5 61 49 95 19
Telefax: +33 (0)5 61 31 00 41

France ☒ SALES ☐ Service
Mannesmann Rexroth S.A.
Division Rexroth Indramat
91, Bd. Irène Joliot-Curie
F - 69634 Vénissieux – Cedex
Telefon: +33 (0)4 78 78 53 65
Telefax: +33 (0)4 78 78 53 62

Hungary ☒ SALES ☐ Service
Mannesmann Rexroth Kft.
Angol utca 34
H - 1149 Budapest
Telefon: +36 (1) 364 00 02
Telefax: +36 (1) 383 19 80

Italy ☒ SALES ☒ Service
Mannesmann Rexroth S.p.A.
Divisione Rexroth Indramat
Via G. Di Vittoria, 1
I - 20063 Cernusco S/N.MI
Telefon: +39 02/92 365 270
Telefax: +39 02/700 408 252378

Italy ☒ SALES ☒ Service
Mannesmann Rexroth S.p.A.
Divisione Rexroth Indramat
Via Borgomanero, 11
I - 10145 Torino
Telefon: +39 011/7 50 38 11
Telefax: +39 011/7 71 01 90

Italy ☒ SALES ☐ Service
Mannesmann Rexroth S.p.A.
Divisione Rexroth Indramat
Via del Progresso, 16 (Zona Ind.)
I - 35020 Padova
Telefon: +39 049/8 70 13 70
Telefax: +39 049/8 70 13 77

Italy ☒ SALES ☐ Service
Mannesmann Rexroth S.p.A.
Divisione Rexroth Indramat
Via Mascia, 1
I - 80053 Castellamare di Stabia NA
Telefon: +39 081/8 71 57 00
Telefax: +39 081/8 71 68 85

Italy ☒ SALES ☐ Service
Mannesmann Rexroth S.p.A.
Divisione Rexroth Indramat
Viale Oriani, 38/A
I - 40137 Bologna
Telefon: +39 051/34 14 14
Telefax: +39 051/34 14 22

Netherlands ☒ SALES ☐ Service
Hydraudyne Hydrauliek B.V.
Kruisbroeksestraat 1
(P.O. Box 32)
NL - 5281 RV Boxtel
Telefon: +31 (0)411/65 19 51
Telefax: +31 (0)411/65 14 83
e-mail: indramat@hydraudyne.nl

Netherlands ☐ SALES ☒ Service
Hydrocare B.V.
Kruisbroeksestraat 1
(P.O. Box 32)
NL - 5281 RV Boxtel
Telefon: +31 (0)411/65 19 51
Telefax: +31 (0)411/67 78 14

Norway ☒ SALES ☐ Service
Rexroth Mecman AS
Rexroth Indramat Division
Berghagan 1 or: Box 3007
N -1405 Ski-Langhus N -1402 Ski
Telefon: +47 (0)64 86 41 00
Telefax: +47 (0)64 86 90 62

Poland ☒ SALES ☐ Service
Mannesmann Rexroth Sp.zo.o.
Biuro Poznan
ul. Dabrowskiego 81/85
PL - 60-529 Poznan
Telefon: +48 061/847 67 99
Telefax: +48 061/847 64 02

Rumania ☒ SALES ☐ Service
Mannesmann Rexroth Sp.zo.o.
Str. Drobety nr. 4-10, app. 14
RO - 70258 Bucuresti, Sector 2
Telefon: +40 (0)1/210 48 25
+40 (0)1/210 29 50
Telefax: +40 (0)1/210 29 52

Russia ☒ SALES ☒ Service
Tschudnenko E.B.
Arsenia 22
RUS - 153000 Ivanovo
Rußland
Telefon: +7 093/223 96 33
oder/or +7 093/223 95 48
Telefax: +7 093/223 46 01

Spain ☒ SALES ☒ Service
Mannesmann Rexroth S.A.
Divisiòn Rexroth Indramat
Centro Industrial Santiga
Obradors s/n
08130 Santa Perpetua de Mogoda
Barcelona
Telefon: +34 937 47 94 00
Telefax: +34 937 47 94 01

Spain ☒ SALES ☒ Service
Goimendi S.A.
División Rexroth Indramat
Parque Empresarial Zuatzu
C/ Francisco Montagne no.2
20018 San Sebastian
Telefon: +34 9 43/31 84 56
Telefax: +34 9 43/31 84 49

Sweden ☒ SALES ☒ Service
Rexroth Mecman Svenska AB
Rexroth Indramat Division
Varuvägen 7
S - 125 81 Stockholm
Telefon: +46 (0)8/727 92 00
Telefax: +46 (0)8/647 32 77

Slowenia ☒ SALES ☒ Service
Rexroth Indramat
elektromotorji d.o.o.
Otoki 21
SLO - 64 228 Zelezniki
Telefon: +386 64/61 73 32
Telefax: +386 64/64 71 50

Switzerland-East- ☒ SALES ☒ Service
Mannesmann Rexroth Schweiz AG
Gesch.ber. Rexroth Indramat
Gewerbestraße 3
CH - 8500 Frauenfeld
Telefon: +41 (0)52/720 21 00
Telefax: +41 (0)52/720 21 11

Switzerland-West- ☒ SALES ☐ Service
Mannesmann Rexroth Suisse SA
Département Rexroth Indramat
Rue du village 1
CH - 1020 Renens
Telefon: +41 (0)21/632 84 20
Telefax: +41 (0)21/632 84 21

Turkey ☒ SALES ☒ Service
Mannesmann Rexroth Hidropar A..S.
Fevzi Cakmak Cad No. 3
TR - 34630 Sefaköy Istanbul
Telefon: +90 212/541 60 70
Telefax: +90 212/599 34 07

Europäische Kundendienstniederlassungen (ohne Deutschland) - European Service agencies (without Germany)

Africa, Asia, Australia – incl. Pacific Rim

vom Ausland:	(x) nach Landeskennziffer weglassen!
from abroad:	don’t dial (x) after country code!

Australia ☒ SALES ☒ Service AIMS - Australian Industrial Machinery Services Pty. Ltd. Unit 3/45 Horne ST Campbellfield , VIC 3061 AUS - Melbourne Telefon: +61 (0)3/93 59 02 28 Telefax: +61 (0)3/93 59 02 86	Australia ☒ SALES □ Service Mannesmann Rexroth Pty. Ltd. No. 7, Endeavour Way Braeside Victoria, 31 95 AUS – Melbourne Telefon: +61 (0)3/95 80 39 33 Telefax: +61 (0)3/95 80 17 33 Email: mel@rexroth.com.au	China ☒ SALES ☒ Service Rexroth International Trade (Shanghai) Co., Ldt. Wai Gaoqiao Free Trade Zone No.122, Fu Te Dong Yi Road Shanghai 200131 - P.R.China Telefon: +86 21/58 66 30 30 Telefax: +86 21/58 66 55 23	China ☒ SALES □ Service Mannesmann Rexroth (China) Ldt. 15/F China World Trade Center 1, Jianguomenwai Avenue Beijing 100004, P.R.China Telefon: +86 10/65 05 03 80 Telefax: +86 10/65 05 03 79
China ☒ SALES □ Service Mannesmann Rexroth (China) Ldt. A-5F., 123 Lian Shan Street Sha He Kou District Dalian 116 023, P.R.China Telefon: +86 411/46 78 930 Telefax: +86 411/46 78 932	China ☒ SALES □ Service Mannesmann Rexroth (China) Ldt. Guangzhou Repres. Office Room 1014-1016, Metro Plaza, Tian He District, 183 Tian He Bei Rd Guangzhou 510075, P.R.China Telefon: +86 20/8755-0030 +86 20/8755-0011 Telefax: +86 20/8755-2387	Hongkong ☒ SALES ☒ Service Rexroth (China) Ldt. 1/F., 19 Cheung Shun Street Cheung Sha Wan, Kowloon, Hongkong Telefon: +852 22 62 51 00 Telefax: +852 27 41 33 44	
India ☒ SALES ☒ Service Mannesmann Rexroth (India) Ltd. Rexroth Indramat Division Plot. 96, Phase III Peenya Industrial Area IND - Bangalore - 560058 Telefon: +91 (0)80/8 39 73 74 Telefax: +91 (0)80/8 39 43 45	India ☒ SALES □ Service Mannesmann Rexroth (India) Ltd. Rexroth Indramat Division Plot. A-58, TTC Industrial Area Thane Turbhe Midc Road Mahape Village IND - Navi Mumbai - 400 701 Telefon: +91 (0)22/7 61 46 22 Telefax: +91 (0)22/7 68 15 31	Indonesia ☒ SALES □ Service PT. Rexroth Wijayakusuma Jl. Raya Bekasi Km 21 Pulogadung RI - Jakarta Timur 13920 Telefon: +62 21/4 61 04 87 +62 21/4 61 04 88 Telefax: +62 21/4 60 01 52	Japan □ SALES ☒ Service Rexroth Automation Co., Ltd. Service Center Japan Yutakagaoka 1810, Meito-ku, NAGOYA 465-0035, Japan Telefon: +81 (0)52/777 88 41 +81 (0)52/777 88 53 +81 (0)52/777 88 79 Telefax: +81 (0)52/777 89 01
Japan ☒ SALES ☒ Service Rexroth Automation Co., Ltd. Rexroth Indramat Division 1F, I.R. Building Nakamachidai 4-26-44, Tsuzuki-ku YOKOHAMA 224-0041, Japan Telefon: +81 (0)45/942 72 10 Telefax: +81 (0)45/942 03 41	Korea ☒ SALES ☒ Service Mannesmann Rexroth-Korea Ltd. Rexroth Indramat Division 1500-12 Dadae-Dong- Saha-Ku Pusan, 604-050 Republic of South Korea Telefon: +82 (0)51/2600 741 Telefax: +82 (0)51/2600 747	South Africa ☒ SALES ☒ Service TECTRA Automation (Pty) Ltd. 28 Banfield Road,Industria North RSA - Maraisburg 1700 Telefon: +27 (0)11/673 20 80 Telefax: +27 (0)11/673 72 69	Taiwan ☒ SALES □ Service Rexroth Uchida Co., Ltd. No.1, Tsu Chiang Street Tu Cheng Ind. Estate Taipei Hsien, Taiwan, R.O.C. Telefon: +886 2/2 68 13 47 Telefax: +886 2/2 68 53 88

Kundendienstniederlassungen in Afrika, Asien, Australien, Pazifik - Service agencies in Africa, Asia, Australia, Pacific Rim

Nordamerika – North America

USA ☒ SALES ☒ Service

Mannesmann Rexroth Corporation
Rexroth Indramat Division
5150 Prairie Stone Parkway
USA -Hoffman Estates, IL 60192-3707

Telefon: +1 847/6 45 36 00
Telefax: +1 847/6 45 62 01
service@indramat.com

Competence Centre America

USA ☒ SALES ☒ Service

Mannesmann Rexroth Corporation
Rexroth Indramat Division
Central Region Technical Center
USA - Auburn Hills, MI 48326

Telefon: +1 248/3 93 33 30
Telefax: +1 248/3 93 29 06

USA ☒ SALES ☒ Service

Mannesmann Rexroth Corporation
Rexroth Indramat Division
Southeastern Technical Center
3625 Swiftwater Park Drive
USA - Suwanee
Georgia 30174

Telefon: +1 770/9 32 32 00
+1 770/9 32 19 03

USA ☒ SALES □ Service

Mannesmann Rexroth Corporation
Rexroth Indramat Division
Northeastern Technical Center
99 Rainbow Road
USA - East Granby,
Connecticut 06026

Telefon: +1 860/8 44 83 77
+1 860/8 44 85 95

USA ☒ SALES □ Service

Mannesmann Rexroth Corporation
Rexroth Indramat Division
Charlotte Regional Sales Office
14001 South Lakes Drive
USA - Charlotte,
North Carolina 28273

Telefon: +1 704/5 83 97 62
+1 704/5 83 14 86

USA Service HOTLINE

+1-800-860-1055

- 7 days / 24hrs -

Canada ☒ SALES ☒ Service

Basic Technologies Corporation
Burlington Division
3426 Mainway Drive
Burlington, Ontario
Canada L7M 1A8

Telefon: +1 905/335 55 11
Telefax: +1 905/335-41 84

Südamerika – South America

Argentina ☒ SALES □ Service

Mannesmann Rexroth S.A.I.C.
Division Rexroth Indramat
Acassusso 48 41/7
RA - 1605 Munro (Buenos Aires)

Telefon: +54 (0)11/4756 01 40
Telefax: +54 (0)11/4762 6862
e-mail:mannesmann@impsat1.com.ar

Argentina ☒ SALES ☒ Service

NAKASE
Servicio Tecnico CNC
Calle 49, No. 5764/66
RA - 1653 Villa Balester
Prov. - Buenos Aires

Telefon: +54 (0) 11/4768 36 43
Telefax: +54 (0) 11/4768 24 13
e-mail: nakase@usa.net
nakase@infovia.com.ar

Brazil ☒ SALES ☒ Service

Mannesmann Rexroth
Automação Ltda.
Divisão Rexroth Indramat
Rua Georg Rexroth, 609
Vila Padre Anchieta
BR - 09951-270 Diadema-SP
[Caixa Postal 377]
[BR-09901-970 Diadema-SP]

Telefon: +55 (0)11/745 90 60
+55 (0)11/745 90 70
Telefax: +55 (0)11/745 90 50
e-mail: awittwer@rexroth.com.br

Brazil ☒ SALES ☒ Service

Mannesmann Rexroth
Automação Ltda.
Divisão Rexroth Indramat
R. Dr.Humberto Pinheiro Vieira, 100
Distrito Industrial
BR - 89220-390 Joinville - SC
[Caixa Postal 1273]

Tel./Fax: +55 (0)47/473 58 33
Mobil: +55 (0)47 974 66 45
e-mail: prochnow@zaz.com.br

Mexico ☒ SALES ☒ Service

Mannesmann Rexroth Mexico S.A. de C.V.
Calle Neptuno 72
Unidad Ind. Vallejo
MEX - 07700 Mexico, D.F.

Telefon: +52 5 754 17 11
+52 5 754 36 84
+52 5 754 12 60
Telefax: +52 5 754 50 73
+52 5 752 59 43

Kundendienstniederlassungen in Nord-/ Südamerika - Service agencies in North- & South- America

12 Revisions to this Document

Despite careful creation and proofreading of this document, we cannot guarantee that it is absolutely free of mistakes. It can also be possible that the most recent modifications and/or supplements of the product described here could not be included in the document. If you notice any incorrect or missing specifications in this description, or if you have any suggestions about improving this publication, do not hesitate to tell us about it on this form. Fax a copy of this form to the address below - and you will help us to keep this document up to date.

Thank you very much for your cooperation.

.... ✂ ....

Rexroth Indramat GmbH
+49 (0) 93 52/40-44 65

From:
Company: ____________________
Departm.: ____________________
Name: ____________________

Document information:

Title: ____

Type code: ____

Where did you notice faults? (Chapter, Page, Fig., Table)

① Chapter:______ Page:_______ Fig.: _______ Table: _____

② Chapter:______ Page:_______ Fig.: _______ Table: _____

③ Chapter:______ Page:_______ Fig.: _______ Table: _____

What is the problem? (please explain in detail what is wrong or what is missing)

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

Notes

287959

Printed in Germany

Rexroth
Indramat