

AppWizard

Wizard for creating ready-to-use emWin applications

User Guide & Reference Manual

Document: UM03003
Software Version: 1.44
Revision: 0
Date: March 12, 2024



A product of SEGGER Microcontroller GmbH

www.segger.com

Disclaimer

The information written in this document is assumed to be accurate without guarantee. The information in this manual is subject to change for functional or performance improvements without notice. SEGGER Microcontroller GmbH (SEGGER) assumes no responsibility for any errors or omissions in this document. SEGGER disclaims any warranties or conditions, express, implied or statutory for the fitness of the product for a particular purpose. It is your sole responsibility to evaluate the fitness of the product for any specific use.

Copyright notice

You may not extract portions of this manual or modify the PDF file in any way without the prior written permission of SEGGER. The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such a license.

© 2024 SEGGER Microcontroller GmbH, Monheim am Rhein / Germany

Trademarks

Names mentioned in this manual may be trademarks of their respective companies.

Brand and product names are trademarks or registered trademarks of their respective holders.

Contact address

SEGGER Microcontroller GmbH

Ecolab-Allee 5
D-40789 Monheim am Rhein

Germany

Tel. +49 2173-99312-0
Fax. +49 2173-99312-28
E-mail: support@segger.com*
Internet: www.segger.com

*By sending us an email your (personal) data will automatically be processed. For further information please refer to our privacy policy which is available at <https://www.segger.com/legal/privacy-policy/>.

Manual versions

This manual describes the current software version. If you find an error in the manual or a problem in the software, please inform us and we will try to assist you as soon as possible. Contact us for further information on topics or functions that are not yet documented.

Print date: March 12, 2024

Software	Revision	Date	By	Description
1.44	0	240312	JE	Chapter <i>Interactions</i> updated. <ul style="list-style-type: none"> • Job SETCOLOR added to GAUGE and MULTIEDIT. • Job SETBKCOLOR added to EDIT, GAUGE, MULTIEDIT and WINDOW. Chapter <i>User Code</i> updated. <ul style="list-style-type: none"> • New function APPW_CalcVar() added. • New functions APPW_SetStreamedBitmap() and APPW_SetStreamed-BitmapEx() added.
1.42	0	231219	JE	New chapter <i>Options / Preferences</i> added. <ul style="list-style-type: none"> • Option 'Use static memory devices' added in order to improve performance on systems with a lot of RAM. • Option 'Run script' added. • Option 'Stay alive loop' added. • Option 'Enable simulation' added.
1.40	1	231030	JE	Chapter <i>Interactions</i> updated. <ul style="list-style-type: none"> • Tools for large projects added.
1.40	0	230705	JE	Chapter <i>Objects</i> updated. <ul style="list-style-type: none"> • New Checkbox object added. • New Radio object added.
1.38	0	230512	JE	Chapter <i>Resource management</i> updated. <ul style="list-style-type: none"> • Font management completely reworked. • Option for merging fonts added. • Option for height- and baseline modification added. • Option for font renaming added. • Detailed example and explanation how to merge fonts added.
1.36a	0	230228	JE	Chapter <i>Objects</i> updated. <ul style="list-style-type: none"> • Decimal mode reworked.
1.36	0	230113	JE	Chapter <i>Directory structure</i> updated. <ul style="list-style-type: none"> • New folder "Keyboard" added. • New folder "Movie" added. Chapter <i>Interactions</i> updated. <ul style="list-style-type: none"> • New signal STARTED added. • New signal STOPPED added. Chapter <i>Objects</i> updated. <ul style="list-style-type: none"> • New Movie object added. • New Movie property added. • Bitmap property reworked. • Colors property reworked. Chapter <i>Resource management</i> updated. <ul style="list-style-type: none"> • New Movie resources added. Chapter <i>User interface</i> updated. <ul style="list-style-type: none"> • "Show missing characters" added to project options. • "Minimum time per frame" added to project options. • "Enable scroller support" added to project options. • Movie button added to quick access buttons. Chapter <i>Variables</i> updated. <ul style="list-style-type: none"> • "Operands" reworked.
1.34	0	221121	FO	Chapter <i>Board support packages (BSPs)</i> updated. <ul style="list-style-type: none"> • Sub-section "Creating custom BSPs" updated. Chapter <i>Objects</i> updated. <ul style="list-style-type: none"> • CCW mode property added to Gauge.
1.32	0	220909	FO	Chapter <i>Objects</i> updated. <ul style="list-style-type: none"> • Sub-section "Object scrollers" added.
1.30	0	220804	JE	Chapter <i>Interactions</i> updated. <ul style="list-style-type: none"> • New job ROTATEDISPLAY added. • New job SETRANGE added.
1.28	1	220601	JE FO	Chapter <i>User interface</i> updated. <ul style="list-style-type: none"> • Section "Importing and exporting of content" added.
1.28	0	220428	FO	Chapter <i>User interface</i> updated. <ul style="list-style-type: none"> • Quick access button section enhanced.

Software	Revision	Date	By	Description
				Chapter <i>Objects</i> updated. <ul style="list-style-type: none"> • New Wheel object added. • The following object property descriptions were moved to the objects they are used in: Auto repeat, Blend colors, Disable animation, Error correction level, Fade mode, Gradient, Keyboard layout, JPEG/GIF/BMP, Line width, Maximum length, Offset, Password mode, Persistent mode, Pixel-size, Rotate marker, Rounded value/ends, Snap position, Start/end angle, Version • Added "Untouchable" property to the objects Image, Text and Box. Chapter <i>Interactions</i> updated. <ul style="list-style-type: none"> • Jobs SETPERIOD, MOVETO, SETALPHA, SETSCALE, SETANGLE, INVALIDATE added. Chapter <i>Command line usage</i> added.
1.26	0	220113	JE	Chapter <i>Objects</i> updated. <ul style="list-style-type: none"> • New Dropdown object added. • New Listbox object added. • New Listview object added. • New property 'Content' added. Chapter <i>Interactions</i> updated. <ul style="list-style-type: none"> • New job ADDITEM added. • New job DELITEM added. • New job INSITEM added. • New job SETITEM added. Chapter <i>Resource management</i> updated. <ul style="list-style-type: none"> • Introduction updated. • Stock fonts updated. • Explanation for use of alpha channel images added. Chapter <i>User interface</i> updated. <ul style="list-style-type: none"> • Workspace directory added to preferences. Chapter <i>Animations</i> updated. <ul style="list-style-type: none"> • More information about autostart option added.
1.24	0	211116	FO	Chapter <i>Drawings</i> added. <ul style="list-style-type: none"> • New feature allowing custom drawings in an application.
1.22	0	210608	JE	Chapter <i>User interface</i> updated. <ul style="list-style-type: none"> • Sample browser added to file options. Chapter <i>Objects</i> updated. <ul style="list-style-type: none"> • New property 'Motion support' added. Chapter <i>Interactions</i> <ul style="list-style-type: none"> • New job SETSTART added. • New job SETEND added.
1.20	0	210308	FO	Chapter <i>User interface</i> updated. <ul style="list-style-type: none"> • Multibuffering option added to project options. Chapter <i>Objects</i> updated. <ul style="list-style-type: none"> • New property 'Stay on top' added. • New property 'Untouchable' added. • Vertical mode: Default bitmaps are switched automatically. Chapter <i>Interactions</i> <ul style="list-style-type: none"> • New signal PIDPRESSED added. • New signal PIDRELEASED added. • New signal UNPINNED added. • New signal FIXED added. • New job SETBITMAP added. • New job SHIFTWINDOW added. • New job ANIMCREATE added. • New job ANIMSTART added. • New job ANIMSTOP added. • "Index" parameter added to job SETCOLOR. • Jobs ANIMCOORD, ANIMVALUE, ANIMRANGE and CASCADECOORD removed. • Signal ANIMCOORD removed. New chapter <i>Animations</i> added. Chapter <i>User Code</i> updated. <ul style="list-style-type: none"> • Added custom user code section. Chapter <i>Board support packages (BSPs)</i> updated. <ul style="list-style-type: none"> • Added "MultibufAvail" option.
1.14	0	210108	FO	Various screenshots updated. Chapter <i>Resource management</i> updated. <ul style="list-style-type: none"> • Sub-chapter 'Variable management' moved to dedicated 'Variables' chapter. Chapter <i>Objects</i> updated. <ul style="list-style-type: none"> • Hexadecimal mode added to Text object.

Software	Revision	Date	By	Description
				<ul style="list-style-type: none"> • New Color property added to Image object. Chapter <i>Interactions</i> updated. <ul style="list-style-type: none"> • Job SETFOCUS added to Button, Multiedit, Rotary, Slider and Switch • New job CALC added. • New job MODALMESSAGE added. Chapter <i>Variables</i> added. Chapter <i>AppWizard SPY</i> updated. <ul style="list-style-type: none"> • Requirements added.
1.12	0	201106	FO	Chapter <i>Directory structure</i> updated. <ul style="list-style-type: none"> • SPY directory added. Chapter <i>User interface</i> updated. <ul style="list-style-type: none"> • AppWizard Spy window added. • Menu bar section enhanced. • Object IDs can be edited from within the hierarchic tree view. • By activating the option in the preferences dialog, object IDs can now be shown in the editor window. Chapter <i>Objects</i> updated. <ul style="list-style-type: none"> • Object focus section added. • Focus options property added. • Opaque property added. • Radius property added to Box and Button objects. • Object example screenshots updated. New chapter <i>AppWizard SPY</i> added. Chapter <i>Glossary</i> updated.
1.10	0	200824	FO	Chapter <i>Objects</i> updated. <ul style="list-style-type: none"> • Multiedit object added. • Timer object added. • Password mode property added. Chapter <i>Interactions</i> <ul style="list-style-type: none"> • New signal TIMER added. • New job START added. • New job STOP added.
1.08	0	200805	FO SC	Fixed several typos. Chapter <i>Objects</i> updated. <ul style="list-style-type: none"> • New object Progbar added. • Added 'Frame color' property to Text object. Chapter <i>Interactions</i> updated. <ul style="list-style-type: none"> • New signal ENTER_PRESSED added. • New job SETX0 added. • New job SETY0 added. • New job SETX1 added. • New job SETY1 added. Chapter <i>User Code</i> updated. <ul style="list-style-type: none"> • Function APPW_GetText() added. • Function APPW_GetValue() added. • Function APPW_SetText() added. • Function APPW_SetValue() added.
1.06	2	200626	FO	Chapter <i>Interactions</i> updated. <ul style="list-style-type: none"> • Added section about conditions.
1.06	1	200605	FO	Chapter <i>Objects</i> updated. <ul style="list-style-type: none"> • Added KEYBOARD_ARA layout for Arabic. • Added additional information about Keyboard object.
1.06	0	200602	FO	Chapter <i>Getting started</i> updated. <ul style="list-style-type: none"> • AppWizard version section added. Chapter <i>Objects</i> updated. <ul style="list-style-type: none"> • Added Keyboard object. • Space property added. • Keyboard layout property added. • Text rotation property added. • Text wrapping property added. • Object properties reworked. Chapter <i>Interactions</i> updated. <ul style="list-style-type: none"> • Job ENABLEPID added. • Job SETFOCUS added. Chapter <i>User Code</i> updated. <ul style="list-style-type: none"> • Function APPW_SetCustCallback() added.
1.04	0	200408	FO	Chapter <i>User interface</i> updated. <ul style="list-style-type: none"> • Interaction window section enhanced. • Screenshots for positioning logic updated. Chapter <i>Objects</i> updated.

Software	Revision	Date	By	Description
				<ul style="list-style-type: none"> • Added Gauge object. • Added QRCode object. • Error correction level property added. • Pixelsize property added. • Version property added. • Line width property added. • Rounded value/ends property added. • Start/end angle property added. Chapter <i>Interactions</i> updated. <ul style="list-style-type: none"> • Signal <code>TEXT_CHANGED</code> added. • Job <code>SETLANG</code> added. • Names of signals and slots have been shortened.
1.02	2	200323	FO	Chapter <i>Board support packages (BSPs)</i> updated. <ul style="list-style-type: none"> • Section 'Importing a custom BSP' updated.
1.02	1	200318	FO	Chapter <i>User interface</i> updated. <ul style="list-style-type: none"> • Screenshots updated. • Added information about Thai support.
1.02	0	200313	FO	Chapter <i>User Code</i> added. <ul style="list-style-type: none"> • Sub-chapter <i>Slot routines</i> merged. • Sub-chapter <i>Screen callback routines</i> added. • Sub-chapter <i>Fonts</i> added. • Sub-chapter <i>Variables</i> merged.
1.00	1	200306	FO	Chapter <i>Interactions</i> updated. <ul style="list-style-type: none"> • 'Slot routines' section updated. • 'Custom user code' section added. Chapter <i>Board support packages (BSPs)</i> updated. <ul style="list-style-type: none"> • Examples updated.
1.00	0	200226	JE FO	Initial release. Chapter <i>Board support packages (BSPs)</i> updated. <ul style="list-style-type: none"> • Section 'Preconfigured BSPs included in the shipment' updated. • Section 'Creating custom BSPs' updated. • Section 'Importing a custom BSP' added.
0.90	1	200221	FO	Chapter <i>Getting Started</i> updated. <ul style="list-style-type: none"> • Added note about the AppWizard Quick Start Guide. Chapter <i>Interactions</i> updated. <ul style="list-style-type: none"> • 'Slot routines' section updated. • Added job-specific parameters to each job.
0.90	0	200102	FO	Initial beta version.

About this document

Assumptions

This document assumes that you already have a solid knowledge of the following:

- The software tools used for building your application (assembler, linker, C compiler).
- The C programming language.
- The target processor.
- DOS command line.

If you feel that your knowledge of C is not sufficient, we recommend *C: A Reference Manual* by Harbison and Steele (ISBN 0--13--089592X). This book provides a complete description of the C language, the run-time libraries, and a style of C programming that emphasizes correctness, portability, and maintainability.

How to use this manual

This manual explains all the functions and macros that the product offers. It assumes you have a working knowledge of the C language. Knowledge of assembly programming is not required.

Typographic conventions for syntax

This manual uses the following typographic conventions:

Style	Used for
Body	Body text.
Parameter	Parameters in API functions.
Sample	Sample code in program examples.
Sample comment	Comments in program examples.
User Input	Text entered at the keyboard by a user in a session transcript.
Secret Input	Text entered at the keyboard by a user, but not echoed (e.g. password entry), in a session transcript.
Reference	Reference to chapters, sections, tables and figures.
Emphasis	Very important sections.
<i>SEgger home page</i>	A hyperlink to an external document or web site.

Table of contents

1	Introduction	16
1.1	What is the AppWizard?	17
1.2	Features	18
1.3	Requirements	20
1.3.1	Host system	20
1.3.2	Target system	20
1.3.3	Development environment	20
1.3.4	Additional software libraries	20
2	Installation	21
2.1	Microsoft Windows	22
3	Getting started	23
3.1	Starting the tool	24
3.2	AppWizard version	24
3.3	Creating a new project	25
3.4	Opening existing projects	27
4	Directory structure	28
4.1	Root folder	29
4.1.1	/Source	29
4.1.1.1	/Source/Config	29
4.1.1.2	/Source/Generated	29
4.1.2	/Resource	31
4.1.2.1	/Resource/Font	31
4.1.2.2	/Resource/Image	31
4.1.2.3	/Resource/Keyboard	31
4.1.2.4	/Resource/Movie	31
4.1.2.5	/Resource/Text	31
5	User interface	32
5.1	Menu bar	33
5.1.1	File / Open example	33
5.1.2	Edit / Options	33
5.1.3	Project / Edit options	33
5.2	Editor window	34
5.3	Property window	36
5.3.1	Id, position and size	36
5.3.2	Positioning logic	36

5.3.3	Positioning details	38
5.3.4	Object dependent details	39
5.4	Hierarchic tree view	40
5.5	Play window	41
5.6	Interaction window	42
5.7	Quick access buttons	43
5.7.1	Text resource window	44
5.7.1.1	Export and import texts in the CSV format	44
5.7.2	Font resource window	46
5.7.3	Image resource window	48
5.7.4	Variable resource window	49
5.7.5	Movie resource window	50
5.7.6	Drawings window	51
5.7.7	Lists window	52
5.7.7.1	Importing and exporting of content	53
5.7.8	Tables window	54
5.8	Starting the simulation project	55
5.9	AppWizard SPY window	56
6	Resource management	57
6.1	Stock resources	58
6.2	Text management	59
6.3	Font management	60
6.3.1	Available information	60
6.3.2	Available operations	61
6.3.2.1	'Create new...'	61
6.3.2.2	'Add existing...'	61
6.3.2.3	'Font codepoint range...'	61
6.3.2.4	'Delete from project'	63
6.3.2.5	'Cleanup'	63
6.3.2.6	'Modify font height...'	63
6.3.2.7	'Merge...'	63
6.3.2.8	'Default codepoint range...'	64
6.3.3	Example how to use merged fonts	65
6.3.3.1	Setting up the project to be able to use Thai and Arabic	65
6.3.3.2	Selecting the fonts to be merged together	65
6.3.3.3	Generating the font files to be merged	65
6.3.3.4	Make the new fonts 'mergeable'	67
6.3.3.5	Include only the codepoints used in the project	68
6.3.3.6	Show result on the screen	68
6.4	Image management	69
6.5	Movie management	71
6.5.0.1	Pitfalls with large files	73
7	Options / Preferences	74
7.1	Options	75
7.1.1	Enable simulation	76
7.1.2	Stay alive loop	76
7.1.3	Use of static memory devices	76
7.1.3.1	Requirements	76
7.1.3.2	How it works	76
7.1.3.3	Performance	77
7.1.4	Run script	77
7.2	Preferences	78
8	Objects	80
8.1	Introduction	81
8.2	Object properties	84

8.2.1	Alignment	86
8.2.2	Bitmap	87
8.2.3	Border size	89
8.2.4	Colors	90
8.2.5	Content	92
8.2.6	Cursor inversion	95
8.2.7	Decimal mode	96
8.2.8	Focus options	97
8.2.9	Font	98
8.2.10	Frame radius	99
8.2.11	Frame size	100
8.2.12	Hexadecimal mode	101
8.2.13	Horizontal mode	102
8.2.14	ID	103
8.2.15	Initial value	104
8.2.16	Invert direction	105
8.2.17	Motion partner	106
8.2.18	Motion support	107
8.2.19	Movie	108
8.2.20	Opaque mode	109
8.2.21	Overwrite mode	110
8.2.22	Period	111
8.2.23	Position and size	112
8.2.24	Radius	113
8.2.25	Range	114
8.2.26	Space	115
8.2.27	Space between items	116
8.2.28	Span of values	117
8.2.29	Stay on top	118
8.2.30	Text	119
8.2.31	Text rotation	120
8.2.32	Text wrapping	121
8.2.33	Tiling	122
8.2.34	Untouchable	123
8.2.35	Vertical mode	124
8.3	Object focus	125
8.4	Object scrollers	126
8.5	Box	129
8.6	Button	131
8.7	Checkbox	132
8.8	Dropdown	133
8.9	Edit	134
8.10	Gauge	136
8.11	Image	138
8.12	Keyboard	140
8.13	Listbox	142
8.14	Listview	143
8.15	Movie	146
8.16	Multiedit	148
8.17	Progbar	150
8.18	QRCode	151
8.19	Radio	152
8.20	Rotary	153
8.21	Screen	154
8.22	Slider	155
8.23	Switch	156
8.24	Text	157
8.25	Timer	158
8.26	Wheel	159
8.27	Window	162

9	Interactions	163
9.1	Introduction	164
9.2	Tools for large projects	166
9.2.1	Using the filter	166
9.2.2	Deactivating interactions	166
9.3	List of signals	167
9.3.1	ANIMEND	168
9.3.2	ANIMSTART	169
9.3.3	CLICKED	170
9.3.4	CREATE	171
9.3.5	DELETE	172
9.3.6	ENTER_PRESSED	173
9.3.7	FIXED	174
9.3.8	GOT_FOCUS	175
9.3.9	INITDIALOG	176
9.3.10	LOST_FOCUS	177
9.3.11	MOTION	178
9.3.12	MOTION_STOPPED	179
9.3.13	PIDPRESSED	180
9.3.14	PIDRELEASED	181
9.3.15	ROTATED	182
9.3.16	STARTED	183
9.3.17	STOPPED	184
9.3.18	RELEASED	185
9.3.19	TEXT_CHANGED	186
9.3.20	TIMER	187
9.3.21	UNPINNED	188
9.3.22	VALUE_CHANGED	189
9.4	List of jobs	190
9.4.1	ADDVALUE	192
9.4.2	ADDITEM	193
9.4.3	ANIMCREATE	194
9.4.4	ANIMSTART	195
9.4.5	ANIMSTOP	196
9.4.6	CALC	197
9.4.7	CLEAR	198
9.4.8	CLOSESCREEN	199
9.4.9	DELITEM	200
9.4.10	ENABLEPID	201
9.4.11	INSITEM	202
9.4.12	INVALIDATE	203
9.4.13	MODALMESSAGE	204
9.4.14	MOVETO	205
9.4.15	ROTATEDISPLAY	206
9.4.16	SET	207
9.4.17	SETALPHA	208
9.4.18	SETANGLE	209
9.4.19	SETBITMAP	210
9.4.20	SETBKCOLOR	211
9.4.21	SETCOLOR	212
9.4.22	SETCOORD	213
9.4.23	SETENABLE	214
9.4.24	SETEND	215
9.4.25	SETFOCUS	216
9.4.26	SETITEM	217
9.4.27	SETLANG	219
9.4.28	SETPERIOD	220
9.4.29	SETRANGE	221
9.4.30	SETSCALE	222

9.4.31	SETSIZE	223
9.4.32	SETSTART	224
9.4.33	SETTEXT	225
9.4.34	SETVALUE	226
9.4.35	SETVIS	227
9.4.36	SETX0	228
9.4.37	SETY0	228
9.4.38	SETX1	228
9.4.39	SETY1	228
9.4.40	SHIFTSCREEN	229
9.4.41	SHIFTWINDOW	230
9.4.42	SHOWSCREEN	231
9.4.43	START	232
9.4.44	STOP	233
9.4.45	TOGGLE	234
9.4.46	NULL	235
9.5	Conditions	236
9.5.1	Introduction	236
9.5.2	Terms and operands	236
10	Variables	239
10.1	Variable management	240
10.2	Calculations	241
10.2.1	Terms and operands	241
10.2.1.1	Operands	241
10.2.1.2	Operators	242
10.2.1.3	Creating a term	243
10.2.1.4	Calculating a variable	243
10.3	Manipulating variables from user code	244
11	Animations	245
11.1	Pre-defining animation IDs	246
11.2	Edit animations	247
11.2.1	Animation properties	247
11.2.2	Start and end time of animation items	247
11.2.3	Animation values	248
11.2.4	Animation ease	248
11.3	Running animations	249
12	Drawings	250
12.1	Creating a drawing object	251
12.2	Defining a drawing	252
12.3	Displaying a drawing	254
12.4	Available drawing functions	255
13	User Code	257
13.1	Slot routines	258
13.1.1	APPW_ACTION_ITEM	259
13.1.2	Custom user code	260
13.2	Screen callback routines	261
13.3	General AppWizard API	262
13.3.1	APPW_CalcVar()	263
13.3.2	APPW_DoJob()	264
13.3.3	APPW_Exec()	265
13.3.4	APPW_GetText()	266
13.3.5	APPW_GetValue()	267
13.3.6	APPW_SetCustCallback()	268
13.3.7	APPW_SetPos()	269

13.3.8	APPW_SetStreamedBitmap()	270
13.3.9	APPW_SetStreamedBitmapEx()	271
13.3.10	APPW_SetText()	272
13.3.11	APPW_SetValue()	273
13.4	Fonts	274
13.4.1	How to use fonts	274
13.4.2	Font API	275
13.4.2.1	APPW_GetFont()	276
13.5	Variables	277
13.5.1	How to use variables	277
13.5.2	Variables API	277
13.5.2.1	APPW_GetVarData()	278
13.5.2.2	APPW_SetVarData()	279
14	Board support packages (BSPs)	280
14.1	Preconfigured BSPs included in the shipment	281
14.1.1	Example	281
14.1.1.1	Step 1: Select BSP	281
14.1.1.2	Step 2: Generate code	281
14.1.1.3	Step 3: Run SEGGER Embedded Studio Project	282
14.1.1.4	Step 4: Compile and run on target	282
14.2	Creating custom BSPs	283
14.2.1	Example	283
14.2.1.1	Step 1: Create a project with AppWizard	283
14.2.1.2	Step 2: Create some elements	283
14.2.1.3	Step 3: Export & Save	283
14.2.1.4	Step 4: Copy evaluation software package into project folder	284
14.2.1.5	Step 5: Exchange libraries	284
14.2.1.6	Step 6: Add file access routines	285
14.2.1.7	Step 7: Add library to project	285
14.2.1.8	Step 8: Add file access routines to the project	286
14.2.1.9	Step 9: Adjust include files	286
14.2.1.10	Step 10: Adjust data alignment	286
14.2.1.11	Step 11: Add support for run-time display rotation (optional)	286
14.2.1.12	Step 12: Add application to project	289
14.2.1.13	Step 13: Compile and run on target	290
14.3	Importing a custom BSP	291
14.3.1	Step 1: Create BSP folder	291
14.3.2	Step 2: Copy project into BSP folder	291
14.3.3	Step 3: Add an image	291
14.3.4	Step 4: Add information file	291
14.3.5	Step 5: Import the BSP into AppWizard	292
14.4	Using the emWin source code	293
14.4.1	Step 1: Remove the pre-compiled static libraries	293
14.4.2	Step 2: Add the source code to the project directory	293
14.4.3	Step 3: Add the source code to the project	293
14.4.4	Step 4: Set include paths	294
15	AppWizard SPY	295
15.1	Requirements	296
15.2	Opening the SPY dialog	297
15.3	Building a project	298
15.4	Running a project	299
15.5	Recording	301
15.6	Playing a recording	303
16	Command line usage	304
16.1	Command format	305

16.2	Command line options	305
17	Glossary	306

Chapter 1

Introduction

This introduction gives some information about this document. It also gives an overview of the AppWizard's features and its requirements.

1.1 What is the AppWizard?

The AppWizard is a tool for creating complete and ready-to-use emWin applications consisting of a number of screens.

Each screen consists of its own graphical control elements like buttons, sliders, images, text, child windows and so on. Applications are generated as a bundle of C files. Those C files are included automatically by the BSPs shipped with the AppWizard or can be used by a custom defined project.

Resources can be compiled and linked with the application code or stored externally on an SD card. Using resources at runtime from SD card is supported by the AppWizard without any additional configuration or code.

1.2 Features

Interactions

Interactions define the application behavior in case of user input. Several methods, animations or swiping can be used to switch between the application screens. To be able to extend the applications behavior user defined code can be invoked on interactions which can be edited within the AppWizard.

Conditions

A condition can be added to an interaction to determine precisely, under which circumstances an action should be executed. Conditions allow to implement a complex program logic into the application using the AppWizard.

Positioning

Positioning of objects can be done by specifying absolute coordinates or relative to already existing elements. Zooming can be used to be able to place small elements.

Resources

Resources like fonts, text and images are managed completely by the AppWizard. That means the application designer gets completely rid of resource management. Resources can be part of the created application code or generated as binary files to be stored on external media. The behavior 'intern' or 'extern' can be specified for each resource separately. That makes it possible to have frequently used resources directly in the addressable ROM area and rarely used resource components on external media. The content of the resource folder is automatically managed by the AppWizard.

Variables and calculations

The user may also add variables to the project which can be manipulated from outside of the application. Variables are mostly used for interactions. For example, an interaction can be set for a variable that will be triggered after its value has changed. An interaction can also change the value of a variable.

Any form of calculation can be done using variables, just like in a C program. This provides even more opportunities for creating AppWizard applications.

Animations

Animations enable the user to animate objects in the application. Since most values within an application can be animated, such as object position and size, variables and object values, there are almost no limits to animations.

Drawings

Drawings allow to utilize a wide array of emWin's 2D graphic drawing functions and add custom drawings to any widget object in the project. The drawing functions include e.g. standard geometric shapes such as rectangles, circles, arcs, and much more. A widget can draw the defined drawing before its standard appearance is drawn (pre-draw) or afterwards (post-draw).

Multiple languages

Multiple languages can be defined in the integrated multilingual text management system. Text can also be part of the application code or located on external media.

Font management

To be able to display the text with the right font the AppWizard contains its own font management system which is used to create emWin-fonts. The range of included codepoints can be specified for each font separately. It can be specified by custom defined pattern files, custom defined ranges or automatically by the range of characters resulting from

the application defined text. Fonts can also be located on an SD card or as part of the application code.

Integrated play mode

The internal play mode can be used for a quick check of the application's behavior without the need of compiling. Display size and color management can be changed on demand within the AppWizard.

BSPs (Board Support Packages)

The AppWizard comes with a set of BSPs which may also include precompiled libraries of emFile and embOS. In case of using a BSP the possibility of changing the color format and display size are restricted. Those ready-to-use and preconfigured BSPs make it possible to write and execute applications without any knowledge of writing applications in C code. All BSPs automatically include the generated application code, which means nothing needs to be changed or configured to be able to run the application on the target. Of course it is also possible to use custom defined BSPs with the AppWizard.

Simulation

Projects generated by the AppWizard also contain a simulation project for Microsoft Visual Studio. The difference between the integrated play mode and the simulation is that application defined code is not compiled and executed by the play mode. The simulation on the other hand also runs application defined code.

AppWizard SPY

The emWin SPY tool is fully integrated into AppWizard as AppWizard SPY. This tool allows the monitoring of memory usage in the application, as well as properties of the widgets that are present. The tool also has a logging feature and a recording feature so that certain debug cases can be rerun.

1.3 Requirements

1.3.1 Host system

The first version is available for Windows systems only, requiring Windows 7 or newer.

The recommended screen resolution is at least full HD (1920 * 1080).

1.3.2 Target system

To be able to use applications generated by the AppWizard we recommend that the target at least fulfills the following requirements:

- At least 256 KBytes of flash. *1
- At least 130 KBytes of RAM. *2
- At least a 32 bit CPU running at 100 MHz or more. *3

At the end RAM and ROM requirement depends on the application built with the AppWizard.

Note

*1 256 KBytes of flash memory are required for emWin and the AppWizard library. Additional flash memory or external storage is required for resources like fonts, images and text.

*2 130 KBytes are required for emWin including 100 KByte of working RAM. This does not include memory required for a framebuffer.

*3 Ideal would be a device with a hardware accelerator such as D/AVE 2D by Renesas or Chrom-ART Accelerator by ST.

1.3.3 Development environment

The AppWizard can be used with any IDE and any ANSI C compiler complying with at least one of the following international standards:

- ISO/IEC/ANSI 9899:1990 (C90) with support for C++ style comments (//)
- ISO/IEC 9899:1999 (C99)
- ISO/IEC 14882:1998 (C++)

1.3.4 Additional software libraries

No additional software library is required to be able to use the AppWizard. The AppWizard optionally supports resource (fonts, images and text) management from external storage. If external storage should be used for resource management a file system for reading operations is required. Any file system can be used.

Chapter 2

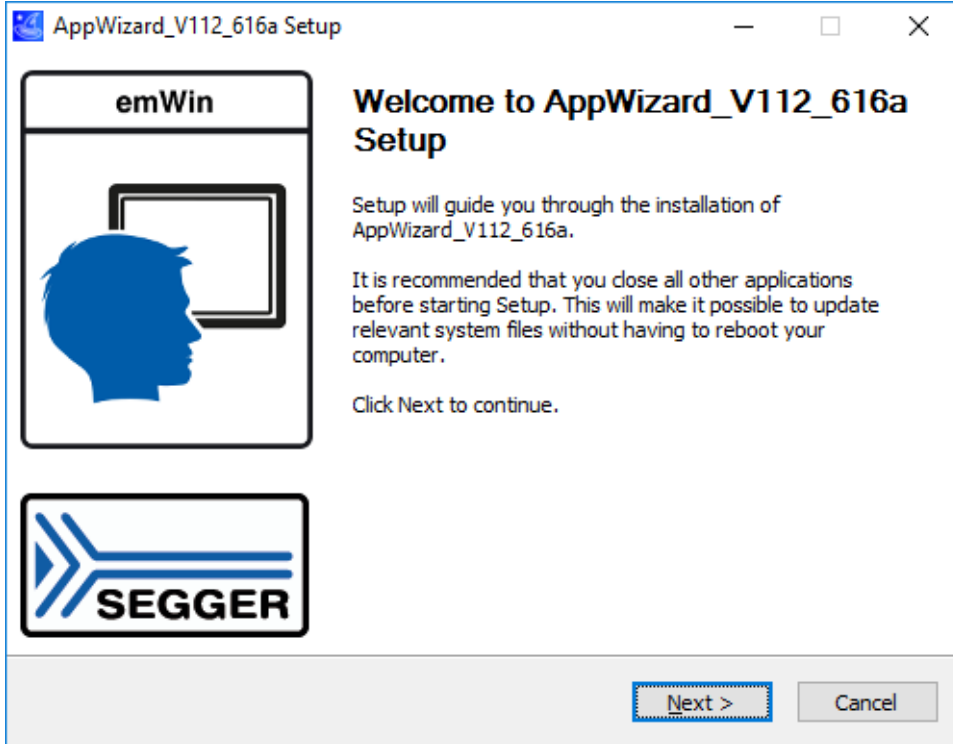
Installation

The following chapter describes how to install the AppWizard.

2.1 Microsoft Windows

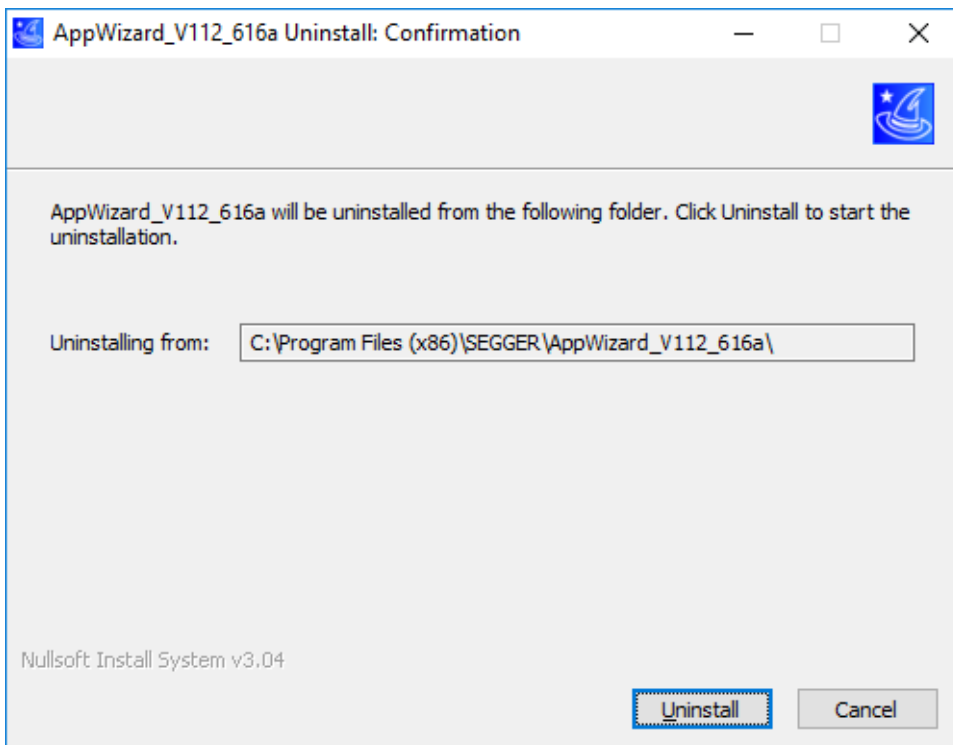
Installing the AppWizard

To install the AppWizard, simply run the setup wizard which will guide you through the installation. It comes with all required components without use of downloading and installing further tools.



Uninstalling the AppWizard

To uninstall the AppWizard, simply run the uninstaller which is located in the program directory.



Chapter 3

Getting started

The following chapter will provide an overview on how to get started with the AppWizard right after the installation has finished.

Note

The shipment also includes a **Quick Start Guide** to the AppWizard which provides step-by-step guides for creating example projects or performing simple actions (e.g. adding objects to the screen).

The guide is located in the `docs` directory and named **AN03003_AppWizard_QuickStartGuide.pdf**.

3.1 Starting the tool

The AppWizard application (AppWizard.exe) can be started from the Windows Start menu or the installation directory.

3.2 AppWizard version

The current version of the AppWizard can be read in the top bar of the program. The version number contains the AppWizard version and the emWin version that is used.

The image shows a small rectangular window with a blue title bar. The text inside the window reads "AppWizard V1.12_6.16a".

In the above example, AppWizard version 1.12 is used in conjunction with emWin version 6.16a.

APP_Version.h

The current version of AppWizard is defined in the file `APP_Version.h`. The example below equals V1.04.

```
#define APPW_VERSION    10400
```

GUI_Version.h

The current emWin version is defined in `GUI_Version.h`. The example below equals V6.12.

```
#define GUI_VERSION    61200
```

Note

The used emWin libraries/code must match the corresponding emWin and AppWizard versions.

3.3 Creating a new project

The following section will guide you through the entire process of creating and running a project with the AppWizard.

1 Create a new project

The initial step is to create a new project. Right after opening the AppWizard, the user has the option to either create a new project or open an existing one.

When creating a new project, the user can choose the project path, a name for the project, specify the target's display size and pick a color format. Alternatively a BSP can be selected, which already includes the respective display size and color format. The user also has the option to enable extern storage mode by ticking the checkbox next to the SD card image. Other options are to enable support of Thai script or bi-directional text.

The screenshot shows the 'Create new project' dialog box with the following settings:

- Project:** Project path: C:\Work\AppWizard\NewProject1 (Browse button); Project name: NewProject1 (Browse button)
- BSP:** Selected BSP: None (Select BSP button)
- Color scheme and display options:** Display size x: 480; Display size y: 272; Color format: 32 Bit, GUICC_M8888I; Enable Multibuffering:
- Text:** Show text from SD-card: ; Show missing characters: ; Enable bi-directional text: ; Enable Thai support:
- Focus:** Enable focus support: ; Select focus color: [Color Picker]; Set focus radius: 0; Set focus width: 1
- Animations:** Minimum time per frame: 40
- Scroller:** Enable scroller support:

When generating a project, the AppWizard also generates a simulation project in the folder \Simulation located in the project directory.

2 Build up a structure

After the project has been created, the user can start to build their application by dragging objects onto the screen, adding interactions to the objects, or adding their resources to the project like bitmaps.

The first thing to add to an empty application is a screen object. This object serves as a parent object for all other objects to be added. Window objects may be added to the screen object to divide the screen into different sections, enhancing the application structure.

Objects like buttons can be placed into the screens or windows and the object's properties can be edited in the 'Properties' window to the right.

A more detailed explanation on how the user interfaces work can be found in the chapter *User interface* on page 32. To learn more about objects, see the chapter *Objects* on page 80.

3 Run and test the application

As the user is building their application, they can run their application during this entire process of building. This makes it very easy to test the application. The application can be run by entering play mode which is done by clicking the play button in the top right corner of the editor window. More information about this can be found in *Play window* on page 41.

4 Export and save the project

The option "File → Save" (<CTRL>+S) simply saves the project file. If the user wants to save their application as C files, they are able to save and export their project by clicking "File → Export & Save" (<CTRL>+<SHIFT>+E). By doing this, the AppWizard generates C files from this project.

5 Run the simulation project

Once the project has been exported, the AppWizard generated C sources with runnable emWin code. The source files are located in the \Source directory. To run the generated code, the simulation project can be used which the AppWizard generated after the creation of the project. The exported source files are automatically linked to the simulation project, which means it is ready to be run.

6 Compile and run on target

The chapter *Board support packages (BSPs)* on page 280 explains how to run a project on a hardware target.

3.4 Opening existing projects

The user may open existing projects either on start-up of the AppWizard by clicking the button "Open existing project" or by using the command "File → Open" (<CTRL>+O).

The only files applicable for opening are AppWizard project files that have a `.AppWizard` extension. When opening an existing project, the project settings may be changed by selecting a different BSP, if needed.

Chapter 4

Directory structure

This chapter gives an overview on how the structure of an AppWizard project looks like.

4.1 Root folder

The root folder contains the project file `<PROJECT_NAME>.AppWizard`. It also contains the following sub-folders:

Folder	Description
<code>/Source</code>	Root directory for the generated application source code.
<code>/Source/CustomCode</code>	Directory for application source code which the user is allowed to extend by custom code.
<code>/Source/Generated</code>	Directory for fixed application code which should not be edited.
<code>/Resource</code>	Root directory for resource files.
<code>/Resource/Font</code>	Fonts created or referred by the project including generated C files.
<code>/Resource/Image</code>	Images used by the project including generated C and DTA files.
<code>/Resource/Keyboard</code>	Keyboard layouts used by the project including generated C files.
<code>/Resource/Movie</code>	Movies used by the project including generated C files.
<code>/Resource/Text</code>	Text defined in the multilingual text editor.
<code>/Target</code>	Selected board support package for target hardware.
<code>/Simulation</code>	Simulation project.
<code>/Spy</code>	AppWizard SPY related files.

4.1.1 /Source

The folder `/Source` contains the file with the application entry point. The file is named `<PROJECT_NAME>.c`.

4.1.1.1 /Source/Config

The sub-folder `/Config` contains the following files intended to be changed/enhanced by the user:

File	Description
<code>Application.h</code>	Header file to be used by user defined code.
<code><SCREEN_ID>_Slots.c</code>	Interaction slots to be used to invoke user defined code on interactions.

When opening a project, the AppWizard reads the user defined slot code. It can be edited within the AppWizard.

4.1.1.2 /Source/Generated

The sub-folder `/Generated` contains the following files not intended to be modified by the user.

File	Description
<code>APPWConf.c</code>	Configuration file (text and driver initialization).
<code>Application.h</code>	Header file to be used by user defined code.
<code><SCREEN_ID>.c</code>	Screen definition(s).
<code>Resource.c</code>	Resource and screen information.

File	Description
Resource.h	Prototypes of resource and screen information elements.

4.1.2 /Resource

The folder `/Resource` is the root directory for text, font and image resources:

4.1.2.1 /Resource/Font

The sub-folder `/Font` contains all font files referenced by the project:

File	Description
<code><FONTNAME>.xbf</code>	Binary file of font in XBF format.
<code><FONTNAME>.c</code>	Simple C arrays of XBF font files which are not managed on external memory.

4.1.2.2 /Resource/Image

The sub-folder `Image` contains all image files referenced by the project:

File	Description
<code><IMAGENAME>.<SUFFIX></code>	Image file referenced by the project.
<code><IMAGENAME>.dta</code>	Streamed image.
<code><IMAGENAME>.c</code>	Simple C arrays of streamed image files which are not managed on external memory.

4.1.2.3 /Resource/Keyboard

The sub-folder `Keyboard` contains all keyboard layout files referenced by the project:

File	Description
<code><KEYBOARD_LAYOUT>.skbd</code>	Streamed keyboard layout.
<code><KEYBOARD_LAYOUT>.c</code>	Simple C arrays of streamed keyboard layout.

4.1.2.4 /Resource/Movie

The sub-folder `Movie` contains all movie files referenced by the project:

File	Description
<code><MOVIE>.emf</code>	emWin movie file.
<code><MOVIE>.avi</code>	Movie file in 'Audio Video Interleave' format.
<code><MOVIE>.c</code>	Simple C arrays of movie files which are not managed on external memory.

4.1.2.5 /Resource/Text

The sub-folder `Text` contains all text defined in the project:

File	Description
<code>APPW_Language_<n>.txt</code>	Text file(s), one for each language.
<code>APPW_Language_<n>.c</code>	Simple C arrays of text file(s) if not managed on external memory.

Chapter 5

User interface

The following chapter will give an overview on the user interface of the AppWizard. The user interface of the AppWizard consists of a menu bar and a couple of windows.

The following windows exist:

- 'Editor' window (center/top)
- 'Interactions' window (center/bottom)
- 'Add objects' window (left/top)
- 'Hierarchic tree view' window (left/bottom)
- 'Properties' window (right)
- Quick access buttons for text, fonts, images and variables at the lower left edge

5.1 Menu bar

It consists of the following items:

- File (New project, Open, Save, Save As, Export & Save, Rename, Close project, Open example, Import BSP, Exit, Recent files)
- Edit (Undo, Redo, Cut, Copy, Paste, Delete, Select all, Preferences)
- Project (Edit options, Play F5, Start Simulation F6, Start Spy)
- Resource (Edit Text, Edit Fonts, Edit Images, Edit Variables)
- Help (About, Open User Guide, Open Quick Start Guide, Open Release Notes)

5.1.1 File / Open example

The screenshot shows a window titled 'Examples' with a tree view on the left and a preview area on the right. The tree view lists folders like 'Application' and 'HowTo' with sub-items such as 'Analog', 'ClimateControl', 'Future', 'LightsControl', 'Minimalism', 'Speedometer', 'WeatherForecast', 'HowTo_AddDialog', 'HowTo_CreateFlipbook', 'HowTo_DrawPieChart', 'HowTo_RotateAndScaleImage', 'HowTo_SetAlphaBitmap', 'HowTo_SetCustomDropdownText', and 'HowTo_SetItemOfListView'. The preview area shows two screens: 'Unit Control' with a rotary knob and three input buttons, and 'Unit Status' with various gauges and switches. Below the preview, there is a detailed description of the screens and 'Open'/'Cancel' buttons.

The first screen:

- A complex animation turns a rotary and toggles some switches in a couple of steps.
- The value of the rotary is passed to some smaller rotaries.
- Conditions are used to filter the right rotaries.
- It also shows how to achieve a screen switch after an animation.

The second screen:

- Shows a rotary, which can be turned by a timer.
- Two toggle buttons can be used to switch on rotation cw or ccw.
- It also shows how snapping with sliders can be achieved.
- Calculations are used to take the slider results and convert them into the desired values shown in text objects.

The example browser allows browsing through the available example projects. On the left side it shows a tree with the available examples and a short description. When selecting an example it shows a screenshot and a more detailed description on the right side. When clicking the 'Ok'-button a dialog for selecting the parent folder for the sample is shown. After selecting the parent folder the example will be extracted and immediately loaded into the AppWizard.

5.1.2 Edit / Options

Details can be found under *Preferences* on page 78.

5.1.3 Project / Edit options

Details can be found under *Options* on page 75.

5.2 Editor window

The editor window shows the currently selected screen by drawing it directly with emWin. That makes sure that "what you see is what you get". Additionally each object has a slightly semi-transparent frame which ensures that also invisible objects give a slight optical feedback.

To be able to place graphical objects a screen has to be created at first. That is done by clicking on the screen icon in the 'Add Object' window left to the editor window. Placing controls is done in the same way. Simply drag an element from the 'Add Object' window onto an existing screen or window object in the editor window.

Independent horizontal and vertical placing

Horizontal and vertical placement of an object can be defined independently. The behavior of each axis can be defined by either a relative position and a size or two relative positions. 'Relative' means relative to its parent or relative to a sibling. That makes it possible to create screens or windows which are self-adjusting when changing the parent's or sibling's placement.

Hierarchical structure

Window elements are used to achieve a hierarchic object structure. They can be placed within a screen or an already existing window. When placing objects on a window the position of those objects can be changed by simply moving or animating the window.

Snapping

Snapping is used when moving objects with the mouse. Edges and center points of existing objects are used for snapping. When aligned with other objects the editor generates optical feedback by highlighting the according object and/or center line.

Selecting objects

Left-clicking selects the first object under the clicked coordinate. A selected object has nine drag points for modifying the coordinates, one on each edge, one on each corner and one in the center point.

With the <CTRL> key pressed multiple objects can be selected. Selected objects are getting joint into a selection group. In that case the drag points are getting placed on to the rectangle surrounding the selection group. Rectangle selection can be done by clicking with the left button in an empty area of the editor window, holding the button pressed and dragging the rectangle with the mouse. When releasing the button the objects within the rectangle will be selected.

Positioning

Objects and groups can be positioned by dragging them with the mouse. The drag points are used to modify the geometry of an object. The property window on the right hand side can also be used to modify the size, coordinates and relations of objects.

Concatenating object positions

To concatenate object coordinates, one of the edge drag points of an object has to be connected to the edge of another object using the right mouse button. This will result in when moving the object the other object was connected to, both objects will be moved synchronous on the axis of the drag point.

A concatenated object position can be cleared by selecting any of the nine positioning options. These options are explained under *Positioning logic* on page 36.

Copy/Paste

Single objects, groups or complete screens can be copied and pasted by either using the keyboard or the menu bar. IDs of copied objects are extended with the suffix `'_Copy'`. The AppWizard makes sure that the generated IDs are unique within the current screen.

Zooming and panning

The content of the editor window can be easily zoomed by using the '+' or '-' button, the '+' or '-' key or the mouse wheel in combination with the <CTRL> key.

The zoom level can be reset by pressing the '1:1' button.



The content of the editor window can be moved by panning, which is done by pressing the <SPACE> bar and moving the mouse while pressing the left mouse button.

Play mode

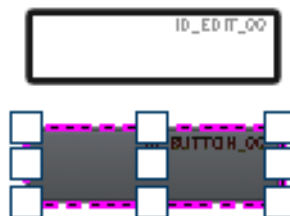
The play button in the upper right corner of the editor window opens the play window, which allows a quick check of the current application.

More information about this window can be found in the chapter *Play window* on page 41.

Object IDs

As mentioned earlier in this chapter, when the option is activated in the Preferences dialog, the object IDs can be shown in the editor window.

The ID is displayed in the upper right corner of an object. The IDs of objects are only shown when any of them is selected and is not shown at all in Play mode.



5.3 Property window

The window on the right shows the object specific properties. It consists of four areas (top to bottom):

- Id, position and size
- Positioning logic
- Coordinate and size modification
- Object specific area

5.3.1 Id, position and size

The top area shows the selected object's Id, which can be edited. Below that it shows the coordinates and size of the object.

Placing details can be modified in the 'Positioning details' area below.

Id	ID_TEXT_00			
Position:	64	201	Size:	100 32

5.3.2 Positioning logic

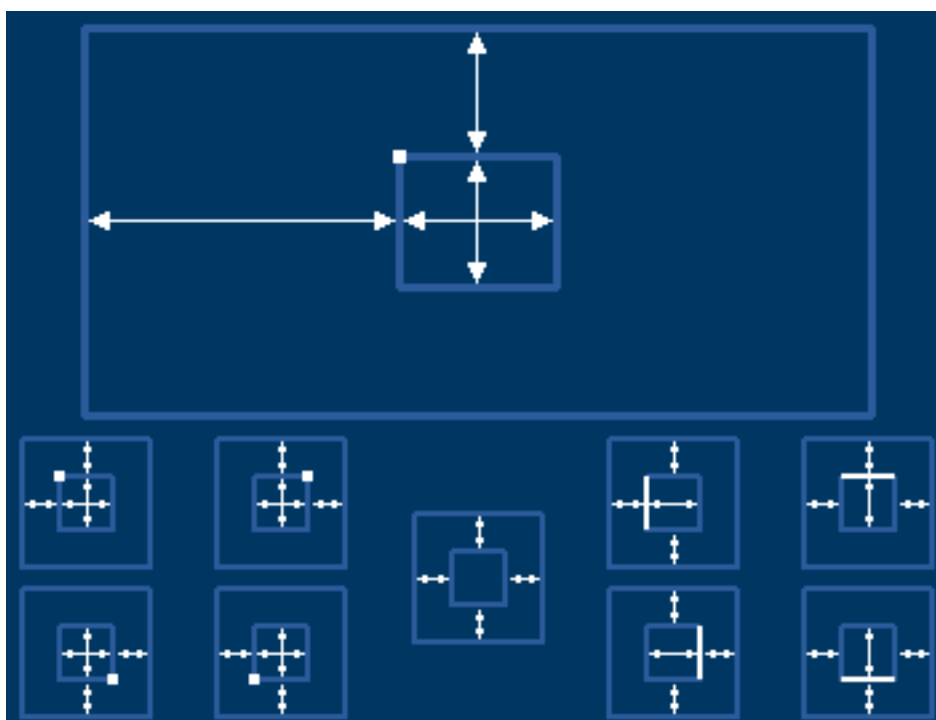
The rectangle of a simple emWin window is defined by its upper left position and its X- and Y-size. To be more flexible with this, the AppWizard supports more options.

One option for example is specifying the coordinates of one of the edges and the objects X- and Y-size. That is similar to a normal emWin window except the option of using any edge and not only the top/left coordinates.

Each coordinate can be relative to an existing edge of the parent or any other sibling. For example, the top coordinate can be relative to the parent, the Y-size fixed and right and left coordinates relative to the parent.










The Y-position of the next object can then be relative to the object above and so on. This mechanism makes it possible to generate screens which are self-adjusting when the parent's size or orientation changes.

To remove a concatenated positioning logic, one of the nine options for positioning logic has to be clicked.



The top of the area shows the positioning logic of the selected object. Dimension lines are used to show coordinate and size definitions. In case of coordinates relative to existing siblings it shows the Id of the according sibling.

There are nine positioning options to choose from:

Positioning option	Description
	Top and left coordinate relative to parent. Width and height defined by given value.
	Top and right coordinate relative to parent. Width and height defined by given value.
	Bottom and right coordinate relative to parent. Width and height defined by given value.
	Bottom and left coordinate relative to parent. Width and height defined by given value.
	Top, left, bottom and right coordinate relative to parent.
	Top, left and bottom coordinate relative to parent. Width defined by given value.
	Top, left and right coordinate relative to parent. Height defined by given value.
	Top, right and bottom coordinate relative to parent. Width defined by given value.
	Left, bottom and right coordinate relative to parent. Height defined by given value.

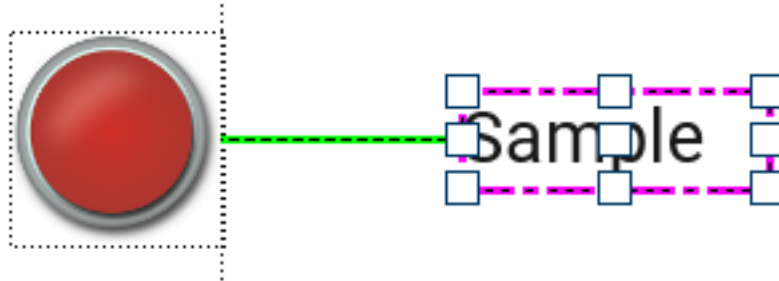
Note

The positioning logic can be changed at anytime.

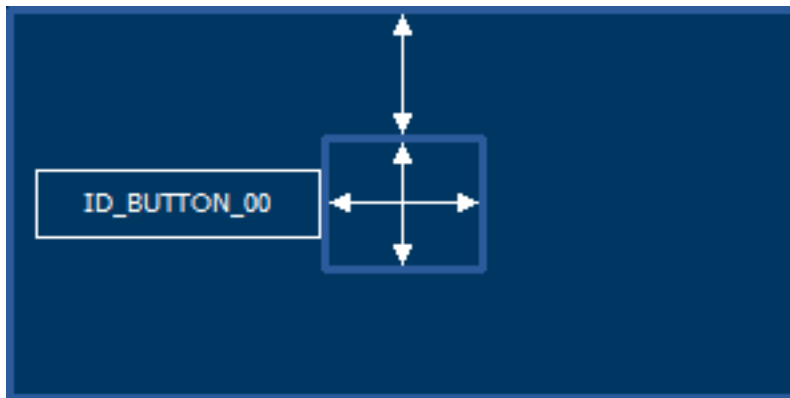
Example

In this example, the Text object's X-position shall be relative to the X-position of the Button object. To do that, one of the Text object's contact points on the X-axis has to be right-clicked. After clicking, a line appears that has to be moved to the Button's X-axis contact point.

When the line appears in a green color, the operation is valid and will be applied when releasing the right mouse button.



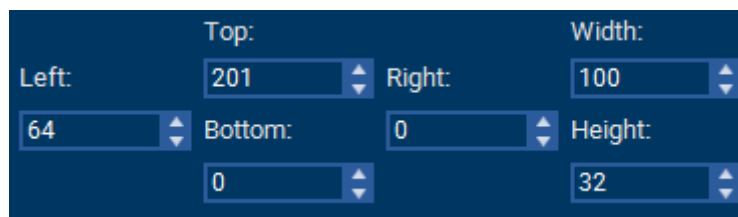
When selecting the Text object, the positioning logic property shows that its X-axis is dependent on the Button object.



To remove this positioning property, the user simply has to select one of the nine positioning options that are explained above.

5.3.3 Positioning details

This section allows setting up top, left, bottom and right coordinates and X/Y size by spin boxes, depending on the selected positioning option.



5.3.4 Object dependent details

Each object has its own properties that can be edited, they are located below the 'positioning details' section in the 'object dependent details' section.

Depending on which object is selected, its properties are shown. To see a list of all existing object properties, see the chapter *Object properties* on page 84.

Editing properties

Each property is shown with a text and an arrow button to the left.



To set or define a property, the arrow button should be clicked.



It opens a configuration area to be able to specify the property details. An existing property can be closed with the arrow button.



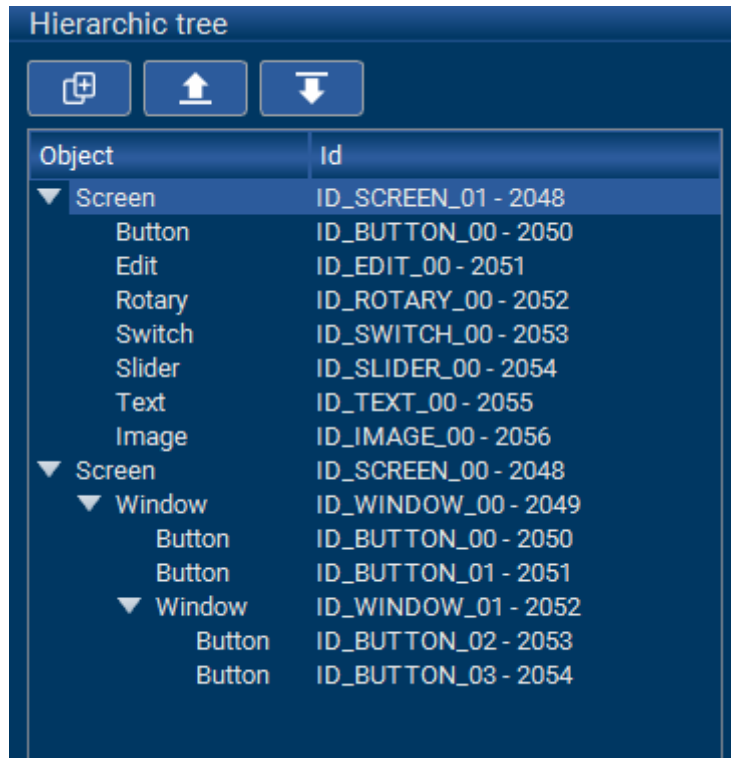
To delete an existing property, the **X** button on the right side has to be clicked.

Clicking the arrow button of an existing property opens and closes the according property definition area. The preferences dialog (Edit → Preferences) allows the option to open all existing property areas per default.

Properties like text, fonts or images open the according resource management and selection dialog.

5.4 Hierarchic tree view

The hierarchic tree view gives a quick overview about the currently existing objects. It allows changing the relative position of siblings per drag and drop. Selecting an object within the tree view also selects the object in the editor window.



Duplicating objects



When clicking the duplicate object button, a copy of the selected object is inserted into the same level of the hierarchic tree.

Moving objects



Clicking the 'Move up' button will move the currently selected object upwards.



Clicking the 'Move down' button will move the currently selected object downwards.

Note

The 'Move up' and 'Move down' buttons can only move an object within their level of the hierarchic tree. This means an object can not be moved to another parent. To move an object to another level/parent, it has to be cut out and pasted to the new location by right-clicking it.

Editing object IDs

The ID of a selected object can be edited when the `ENTER` key is pressed.

5.5 Play window

The play window shows the user a 'running' version of the current project application.

It can be opened by clicking on the play button in the upper right corner of the editor window and closed by pressing the escape key. It may also be opened and closed by pressing the <F5> key.



When opening the play window, a modal dialog with the resulting interactive application will be shown.

Limitations

There is one limitation to the play mode, as the AppWizard does not compile any C code, the play mode does not include any code by the user added to interactions.

5.6 Interaction window

The interaction window shows a list of all interactions associated with the selected screen. Each interaction has its own emitter, signal, job and receiver.

	Emitter	Signal	Job	Receiver	Comment
X	ID_SCREEN_Gratin	INITIAL_DIALOG	SETVAL...	ID_LISTVIEW_Selection	
X	ID_LISTVIEW_Selection	VALUE_CHANGED	SETVAL...	ID_VAR_LISTVIEW_VALUE	Save current index of LISTVIEW in a variable.
X	ID_LISTVIEW_Selection	CLICKED	SETVAL...	ID_VAR_LISTVIEW_OLDVAL_Gratin	Save old LISTVIEW value in variable.
X	ID_LISTVIEW_Selection	VALUE_CHANGED	CALC	ID_VAR_LISTVIEW_SEL_Y0_Gratin	Calculate new selection box position and animate it.
X	ID_LISTVIEW_Selection	VALUE_CHANGED	ANIMCR...		
X	ID_LISTVIEW_Selection	VALUE_CHANGED	ANIMST...		
X	ID_BUTTON_Back	CLICKED	SHIFTS...		
X	ID_LISTVIEW_Selection	RELEASED	SETVAL...	ID_VAR_LAST_SCREEN	Remember which screen we come from (Gratin = 2).
X	ID_LISTVIEW_Selection	RELEASED	SHIFTS...		Move to operation screen if selected element was clicked.
X	ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row1	Set to unselected color first.
X	ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row2	Set to unselected color first.
X	ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row3	Set to unselected color first.
X	ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row4	Set to unselected color first.
X	ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row5	Set to unselected color first.
X	ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row1	Set color to white if selected.
X	ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row2	Set color to white if selected.
X	ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row3	Set color to white if selected.
X	ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row4	Set color to white if selected.
X	ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row5	Set color to white if selected.

Creating a new interaction

Creating a new interaction is done by pressing the + button at the end of the list. To learn more details on how to create new interactions, see the Introduction section of the Interaction chapter.

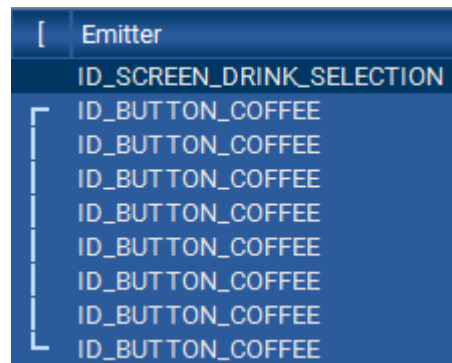
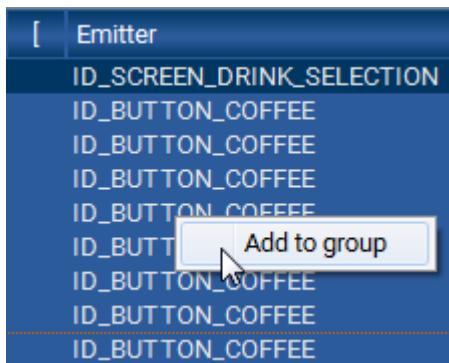
After specifying the receiver a dialog occurs which allows it to specify job dependent data and/or user defined code of the interaction slot. Clicking the pen opens a dialog for editing those parameters.

Removing an interaction is done by clicking the X button in the first column.

Grouping interactions

Interactions may also be grouped together. This makes sense if there is a large number of interactions present in the application and a more overseeable structure is desired. Grouped interactions are marked via the line in the “[” column.

To group interactions, select the interactions, right click them and select “Add to group”. As demonstrated below, after a group has been created, all the interactions between the start and end of the line are in that group.



Now, when one item of the group is clicked, all group items are selected. To delete the group, right click and select “Clear group”.

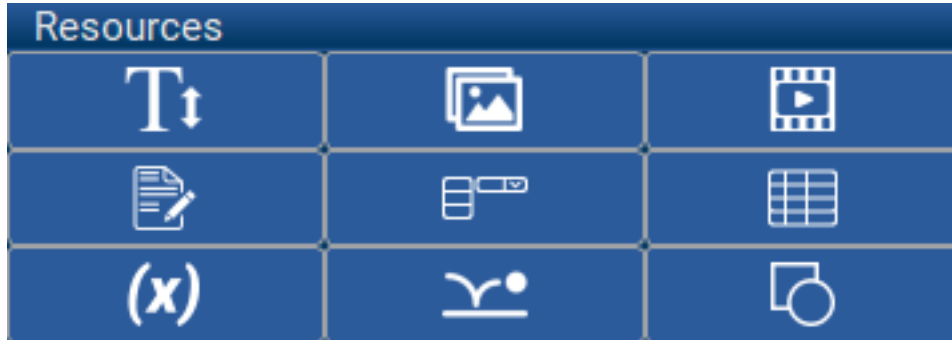
Adding a comment

A comment can be added for each interaction. To do so, simply double click the empty area next to the interaction in the “Comment” column.

5.7 Quick access buttons

In the lower left corner there are six quick access buttons. Clicking on one of the buttons will open the corresponding window that allows management of either texts, fonts, images, variables, animations or drawings.

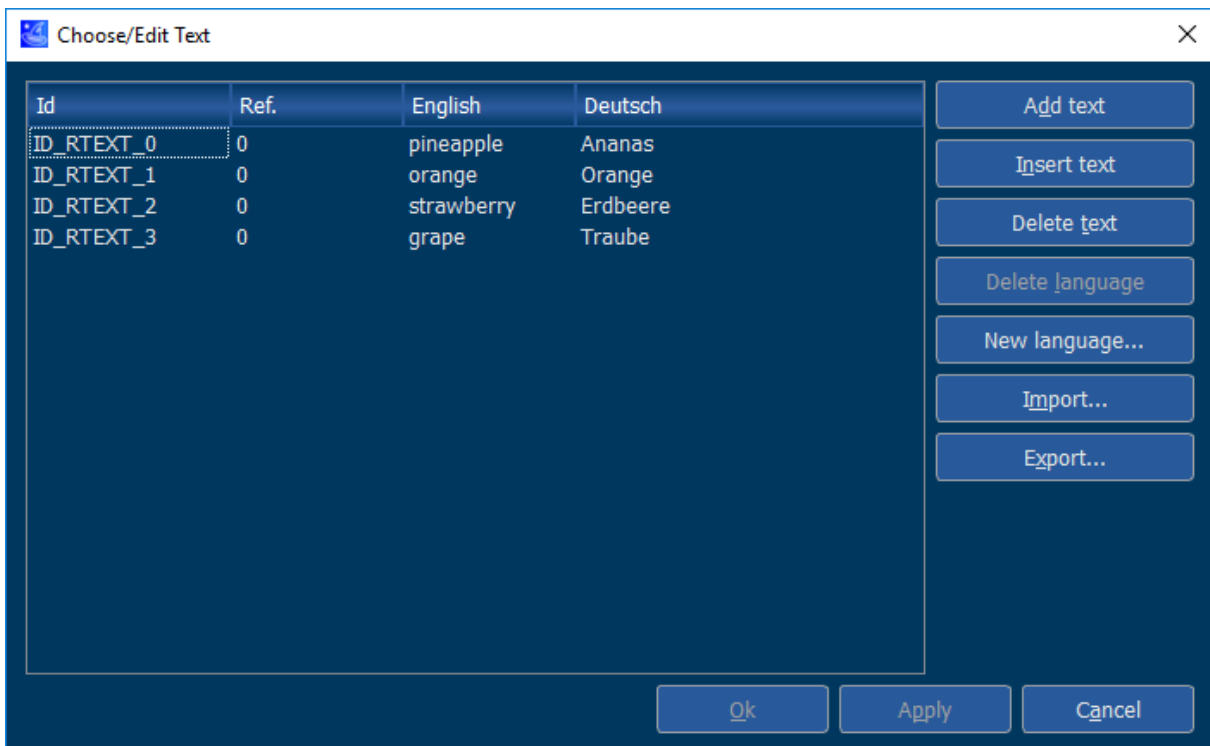
For more information about managing resources, see the chapter *Resource management* on page 57.



5.7.1 Text resource window



The text resource window makes it possible to save texts in multiple languages. The order of the languages may be changed by using drag and drop on the column header.



Each text has its own ID, that can be assigned to objects. The **Ref** column states how often the text is referenced in any objects.

Add texts or languages

Before adding texts, you need to have added at least one language first. New languages/columns are added via the **New language** button.

New texts are added via the **Add text** or **Insert text** buttons. They can be edited by clicking on the corresponding field.

Texts can be deleted via the **Delete text** button, but only when they are not being referenced by any objects.

5.7.1.1 Export and import texts in the CSV format

It is also possible to export the texts to a CSV file or import a CSV file.

CSV file rules

In order for the import of a CSV file to work, the following rules have to be observed:

1. The file must be UTF-8 encoded.
2. The first row has to contain the names of languages.
3. Each following row contains a text item in the given number of languages.
4. Each language string must be enclosed by quotes, e.g. "pineapple".
5. All values are to be separated by commas.
6. After the last value there should be no comma.
7. Optional: A desired ID can be added before the first value in a row. The ID is to be separated with a comma from the values, but it must **not** be enclosed by quotes.

An example CSV file with all the rules applied could look like this:

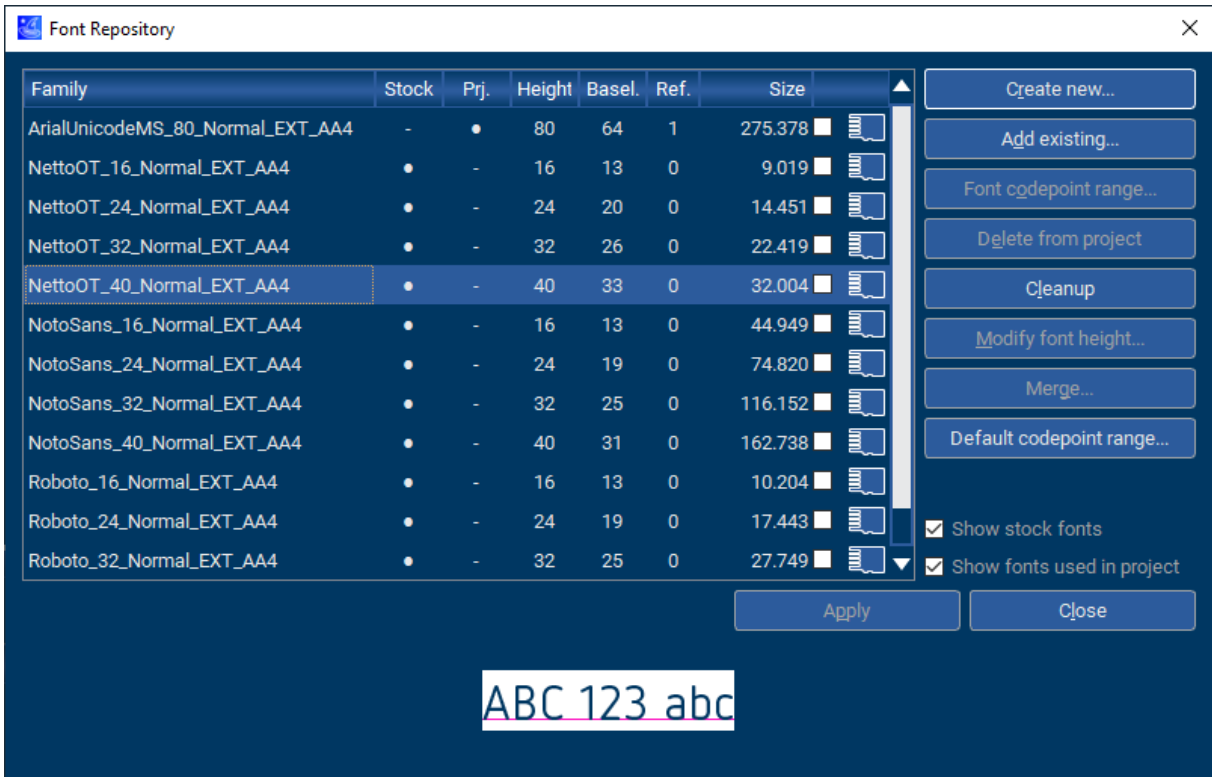
```
"DE" , "EN" , "FR"
```

```
ID_RTEXT_0,"German text 0","English text 0","French text 0"  
ID_RTEXT_1,"German text 1","English text 1","French text 1"  
ID_RTEXT_2,"German text 2","English text 2","French text 2"  
ID_RTEXT_3,"German text 3","English text 3","French text 3"
```

5.7.2 Font resource window



The font resource window allows the user to manage fonts.



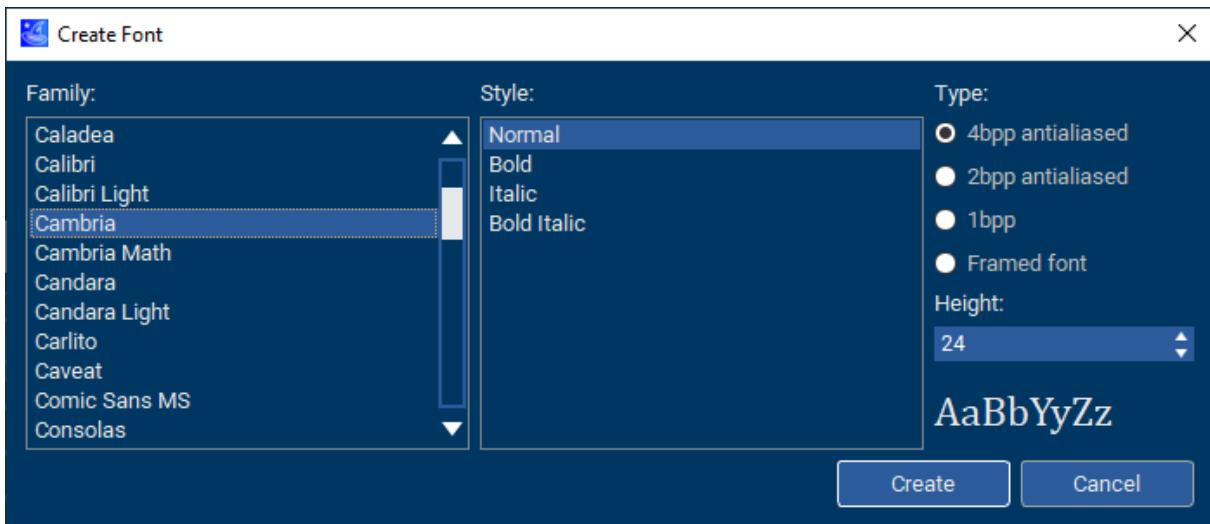
The table shows whether the font is a stock font and/or is used in the project. It also shows how often fonts are referenced in any objects and the height of the font. By ticking the checkbox next to the SD card, the font can be marked as an external resource.

Note

Only XBF files created with the AppWizard can be used!

Create new fonts

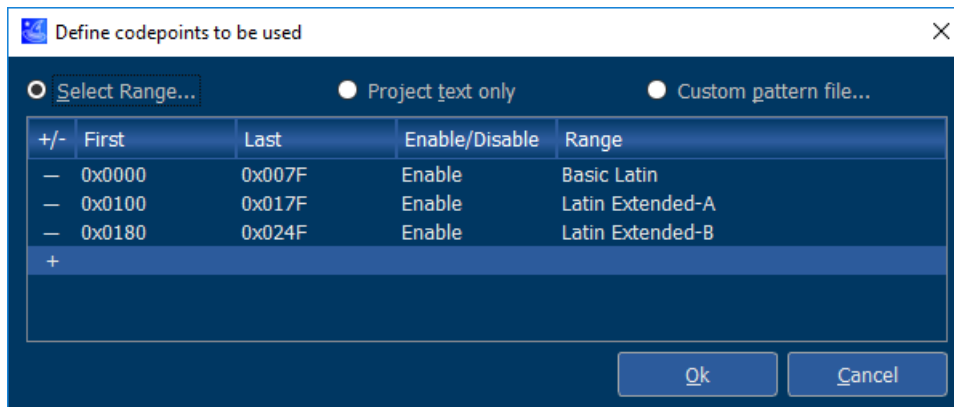
Clicking on **Create new...** allows the user to add a new font from the local installed fonts. When clicking the button, a window similar to that in the FontConverter is opened. The user has to select which font should be added, in which style and height and also select the anti-aliasing level.



To optimize memory footprint, the user may define which characters should be present in the font. This can be done by clicking on **Codepoint range...** and either selecting a range of characters, keeping only the characters that are used in all project's texts or by parsing a pattern file.

By default, a range of characters is used for creating a font. The default range of enabled characters is:

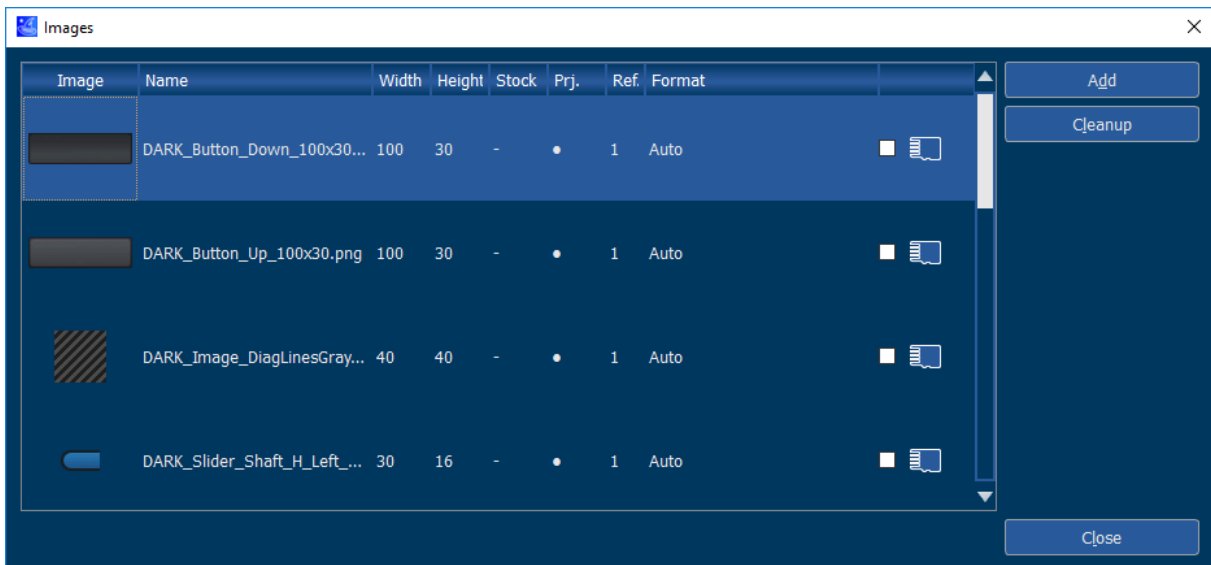
```
0x0000    to    0x007F
0x0100    to    0x017F
0x0180    to    0x024F
```



5.7.3 Image resource window



The image resource window allows the user to manage images.



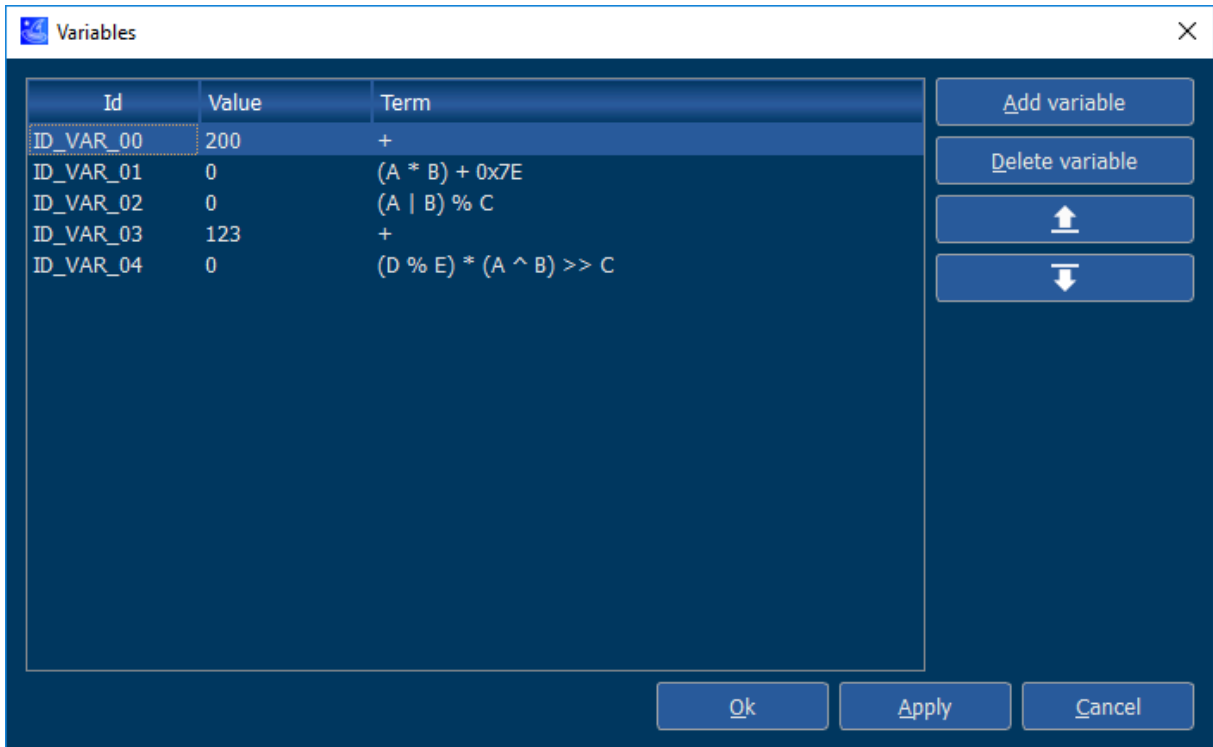
The window gives an overview of all the images used in the project, showing also their dimensions, the bitmap format applied to the image and how often it is referenced in other objects.

As in the other management windows, the user can choose if the image should be marked as an external resource. The **Add** button can be used to add another image from your local disk.

5.7.4 Variable resource window



The variable window lets the user add or remove variables. More explanation on what variables are used for can be read in the chapter *Variables* on page 239.



5.7.5 Movie resource window



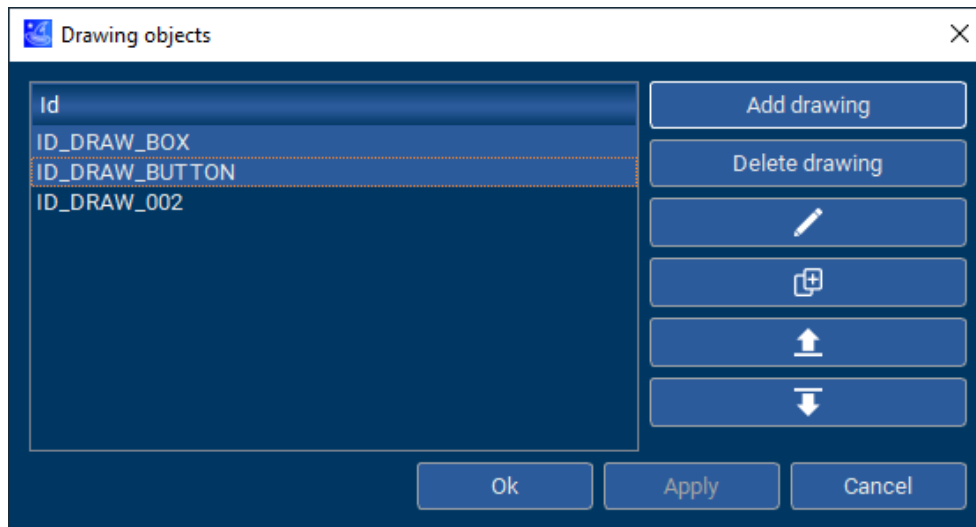
The movie resource window allows the user to manage movies. A detailed explanation about movie management can be found under *Movie management* on page 71.



5.7.6 Drawings window



The drawing window allows the user to define custom drawings that can be used in the application.

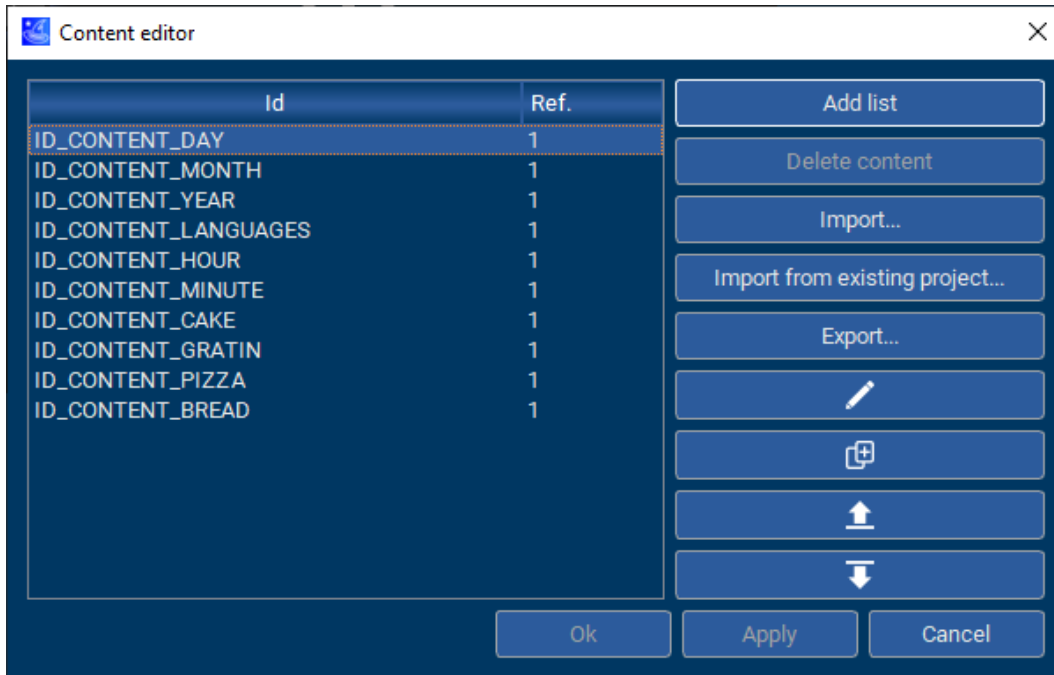


More detail about drawings can be found in the chapter *Drawings* on page 250.

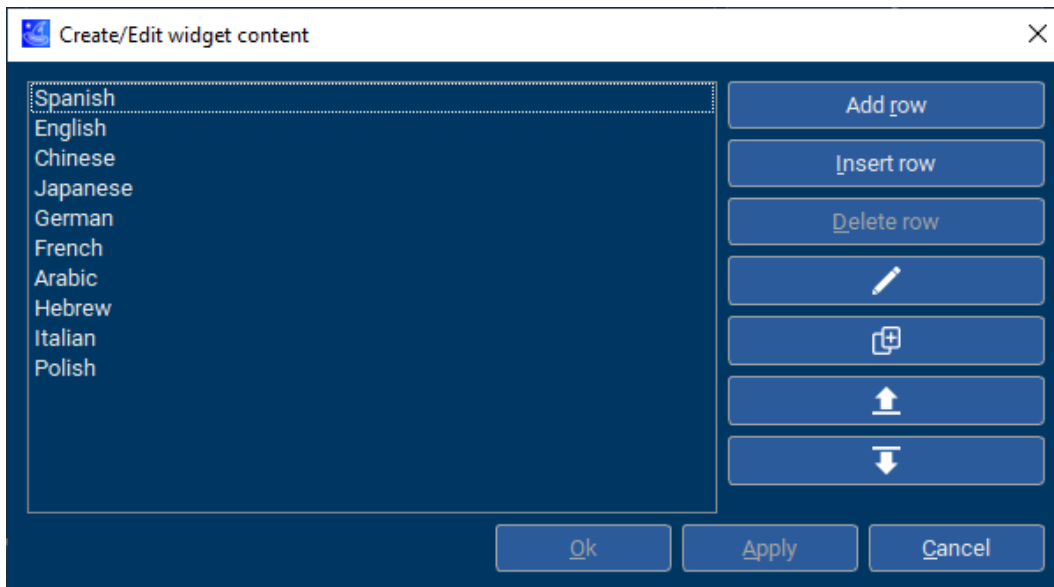
5.7.7 Lists window



The list window allows the user to add content objects in the form of lists. These lists can be used in the application by list-based objects such as Listbox, Dropdown or Wheel.



When clicking the edit button, the list can be edited and rows can be added, deleted or moved.



5.7.7.1 Importing and exporting of content

To import and export `.txt` or `.csv` files as lists or tables, click the corresponding quick access button and click on the (**Import...** or **Export...** button. Note that for importing, multiple files can be selected and imported at once.

TXT file import and export

The TXT format is only available for lists and the format being applied is one line in the file equals one row in the final list.

CSV file import and export

The import and export of `.csv` files is available for lists and tables. Below an example that demonstrates the CSV format being applied:

```
"Header column 0","Header column 1","Header column 2"
"Row 0, Column 0","Row 0, Column 1","Row 0, Column 2"
"Row 1, Column 0","Row 1, Column 1","Row 1, Column 2"
"Row 2, Column 0","Row 2, Column 1","Row 2, Column 2"
```

The following CSV rules are observed:

- Each string is encapsulated by quotes.
- Each value is separated by a comma, except at the end of a line.
- Each line in the text file represents a row in the table.
- The first line in the CSV file is always the header line (this is applied to lists as well).

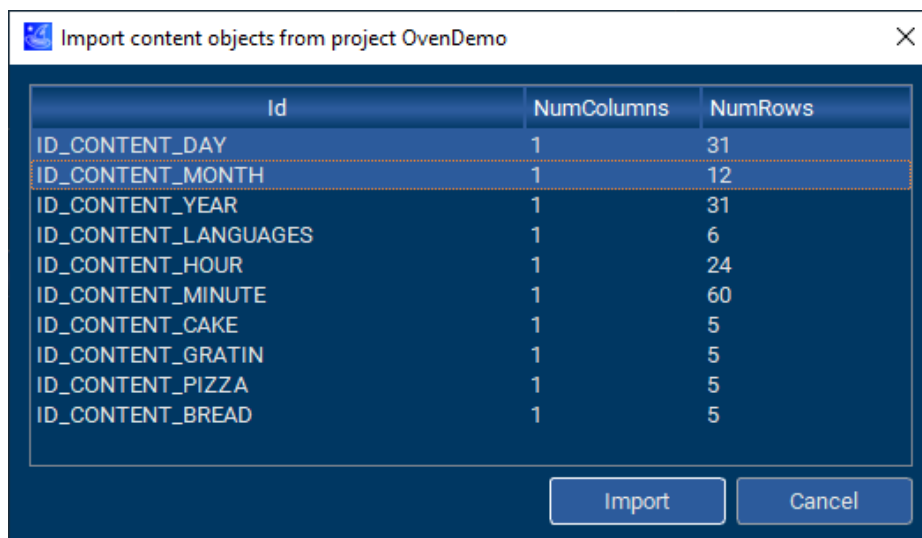
Note

The CSV import and export is intended to only be used for the texts of a table. This means the tables' properties, such as column widths or alignments will not be written to the CSV file. These properties are only imported when importing a table from an existing project (see below).

Importing tables and list from other projects

To import lists or tables from other projects, click the button **Import from existing project...**

If a suitable project file has been selected, the IDs of the existing content objects will be shown. Multiple content objects can be selected for importing.



5.7.8 Tables window



The table window allows the user to add content objects in the form of tables. Tables can be used in the application by table-based objects. As of now, the only table-based object is the Listview.

The table window allows the definition of the actual table of data, but also the column sizes and alignment and the content of the header line of the table.



Col 0	Col 1	Col 2
Item 0/0	Item 1/0	Item 2/0
Item 0/1	Item 1/1	Item 2/1
Item 0/2	Item 1/2	Item 2/2

A more detailed explanation on how the table window works can be found under *Content* on page 92.

5.8 Starting the simulation project

By selecting menu entry `Project` → `Start Simulation` or by pressing `F6` the AppWizard starts automatically the simulation project which is placed within the AppWizard project directory. Depending on the selection made under 'Preferences' either the VS2013 or VS2015 project gets started.

5.9 AppWizard SPY window

The AppWizard SPY window can be opened through the menu entry `Project` → `Start Spy`. Alternatively, the `F7` key can be pressed to open the window.

More details about the features and usage of AppWizard SPY can be read in the chapter *AppWizard SPY* on page 295.

Chapter 6

Resource management

The AppWizard manages all text, fonts, images and movies required for the application. The user gets completely rid of additional resource management like creating font files with the font converter, image files with the bitmap converter, movie files with the movie converter or text data to be used in the project.



Per default resources are compiled and linked into the application. For systems short on ROM, large resources or resource data which should be changeable at runtime, those resources can be managed from SD card with the file system included in the BSP.

Optionally the AppWizard manages the content of the SD card which needs to be available at runtime.

Please refer to the *Creating custom BSPs example* to learn how BSPs with or without a file system can be used.

Furthermore, there are other types of data that can also be seen as kind of 'resources' in a broader sense. These are:

- Tables and lists
- Animations
- Variables
- Drawings

6.1 Stock resources

The AppWizard comes with a bunch of different stock fonts and images that are ready for use for any application.

Note

Note that as with any resources, any stock resources that have been used are saved in the exported project as well!

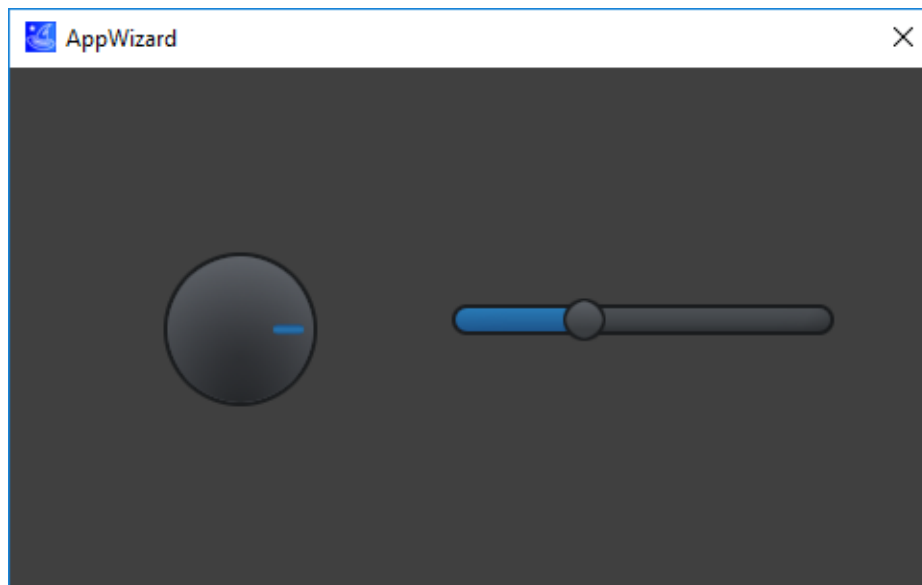
Stock fonts

The AppWizard by default supplies a few fonts, each in different sizes. All stock fonts use 4bpp anti-aliasing.

- NettoOT_16_Normal_EXT_AA4
- NettoOT_24_Normal_EXT_AA4
- NettoOT_32_Normal_EXT_AA4
- NettoOT_40_Normal_EXT_AA4
- NotoSans_16_Normal_EXT_AA4
- NotoSans_24_Normal_EXT_AA4
- NotoSans_32_Normal_EXT_AA4
- NotoSans_40_Normal_EXT_AA4
- Roboto_16_Normal_EXT_AA4
- Roboto_24_Normal_EXT_AA4
- Roboto_32_Normal_EXT_AA4
- Roboto_40_Normal_EXT_AA4

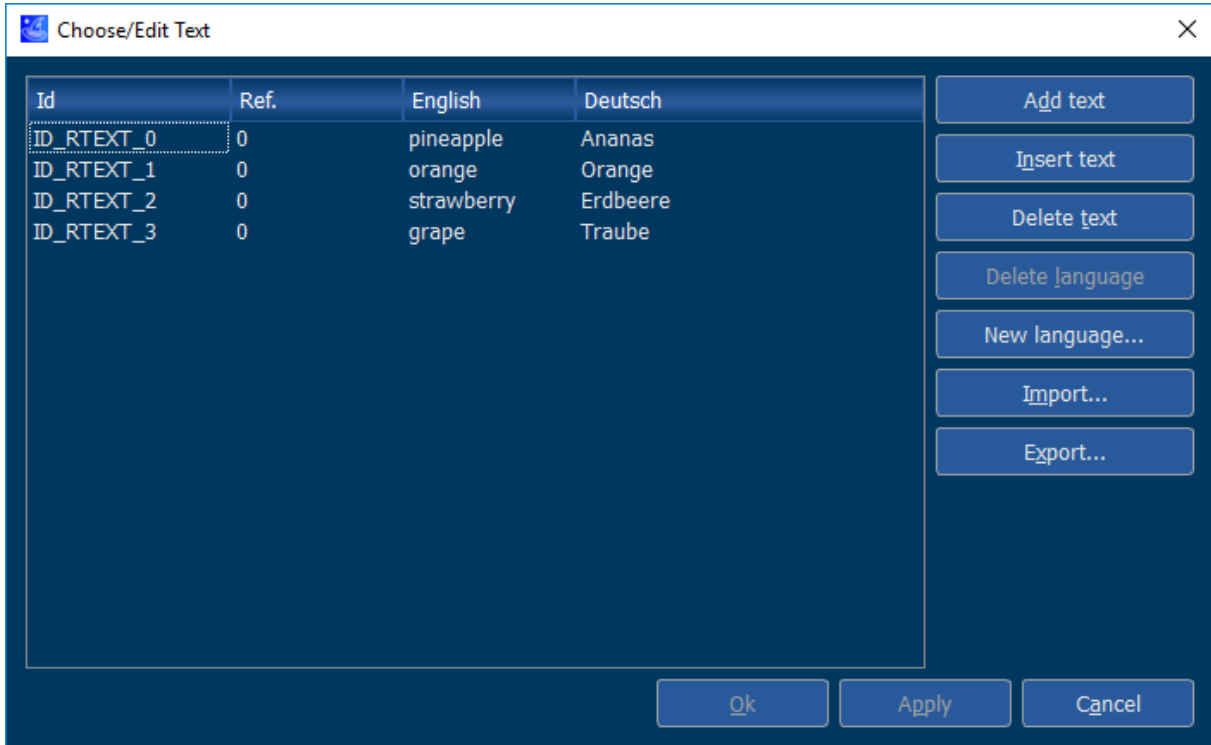
Stock images

For each object which can make use of a bitmap, the AppWizard offers a set of images. For example, bitmaps for a Rotary and its marker, or a thumb and shaft bitmap for a Slider object.



6.2 Text management

A text input dialog allows entering text in multiple languages. Text usage is based on using IDs instead of using strings directly. Text access within the application is realized by using text IDs. In combination with emWin's language module it becomes quite easy to switch between languages. More details about editing text can be found under *Text resource window* on page 44.



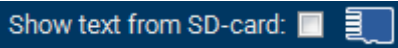
Linebreaks

Linebreaks in text can be achieved with the sequence `\n` in the text strings. A `\n` will be replaced by the AppWizard with a newline.

Managing text from SD card

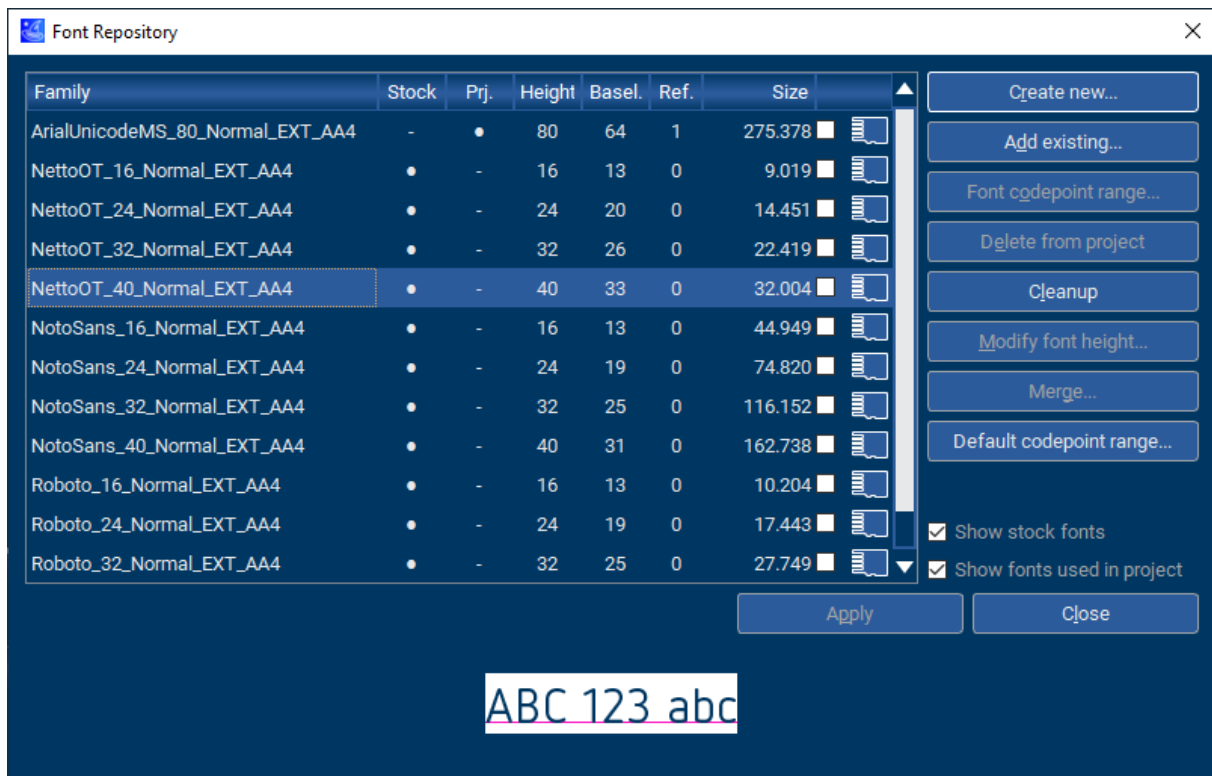
The project property dialog has the option to enable text management directly from SD card. In that case the text is not compiled and linked with the application code.

When exporting the project, the text will then be stored in the specified media path in the directory `<Mediapath>\Resource\Text`.



6.3 Font management

The AppWizard comes with a small set of default fonts in form of XBF font files and the option for creating new fonts. The resource path contains all fonts referenced by the project. A font management dialog shows all available fonts in the project.



6.3.1 Available information

The list of available fonts shows the following columns:

- Font family.
- Stock font, means that the font is located in font stock.
- Project font, means that the font is located in resource folder of the project.
- Height in pixels.
- Baseline in pixels.
- Number of references.
- File size of font file.
- Check box to specify SD card management for that font.

Two checkboxes can be used to determine the content:

- Show stock fonts
- Show project specific fonts

SD card management

When compiling the project all referenced fonts are compiled and linked with the application per default. SD card management excludes the font from compiling and linking with the application. Those fonts are managed from SD card or similar media directly without using addressable ROM for the content of the font. When exporting the project, these fonts will be saved in a directory in the specified media path, that is <Mediapath>\Resource\Fonts.

Font preview

The bottom of the dialog contains a small preview. The characters used for the preview can be defined here: *Font preview string*

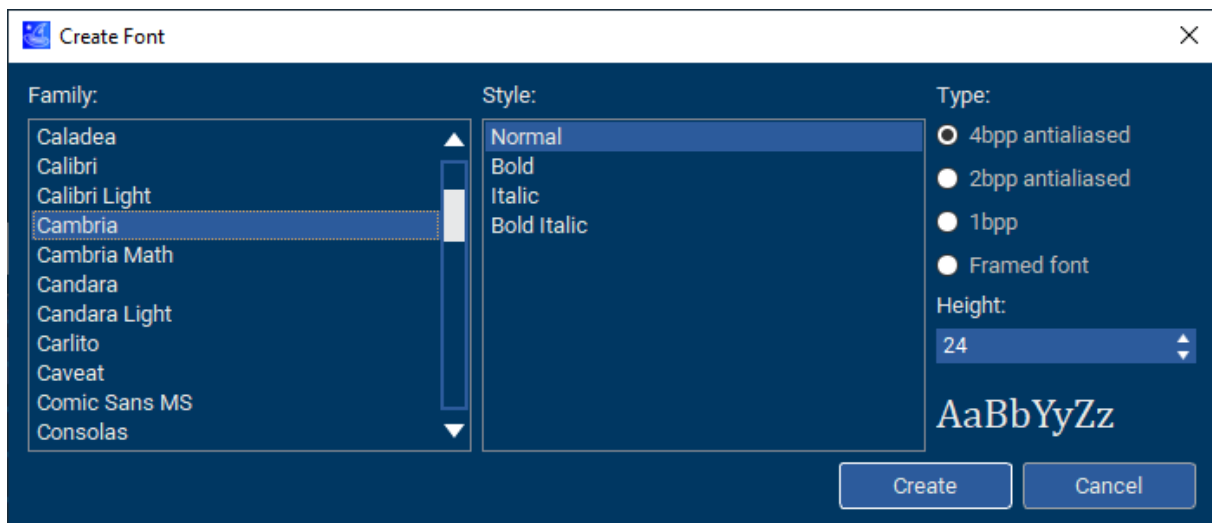
6.3.2 Available operations

The font repository dialog offers the following operations:

- Creating new font files.
- Adding existing files from other projects.
- Modification of the code point range.
- Deleting of (non referenced) fonts.
- Cleanup and remove non referenced fonts.
- Font height modification.
- Merging of font files.
- Modification of default codepoint range.

6.3.2.1 'Create new...'

A new font file will be created with the subsequent shown dialog:



The following types of fonts can be created:

- 4bpp antialiased
- 2bpp antialiased
- 1bpp
- Framed fonts

The resource folder of the project contains all fonts which are used or have been used within the project. That means the folder can contain fonts which are not currently used. Those fonts are shown with zero references and can be deleted by pressing the 'Delete' button if not planned to be used any longer.

Note

Fonts created with the FontConverter can not be used or imported by the AppWizard!

6.3.2.2 'Add existing...'

This option allows importing font files from already existing projects.

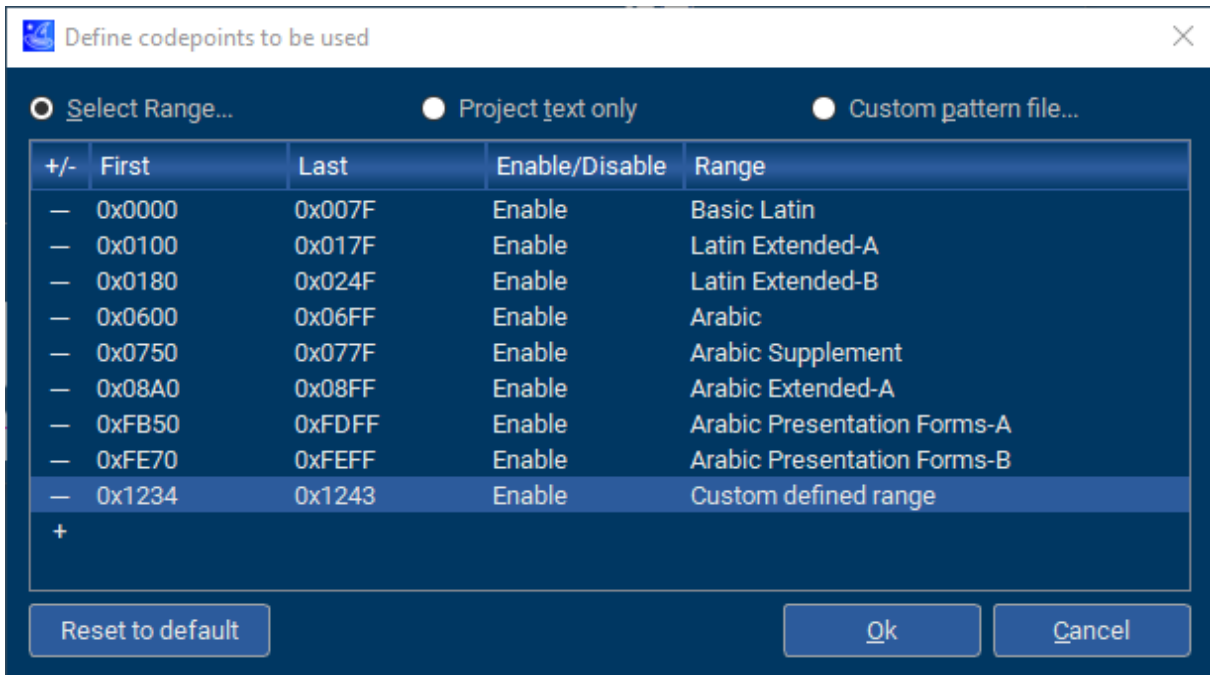
6.3.2.3 'Font codepoint range...'

Each font can have its own range of code points. The font selection dialog has the option for specifying the desired code point range. Clicking the according button opens a dialog for setting up the desired range. The following options exist:

- Setting up a list of code point ranges
- Using all code points required to draw the text defined in the application
- Using a custom pattern file which defines the code points to be used

Working with dedicated code point ranges

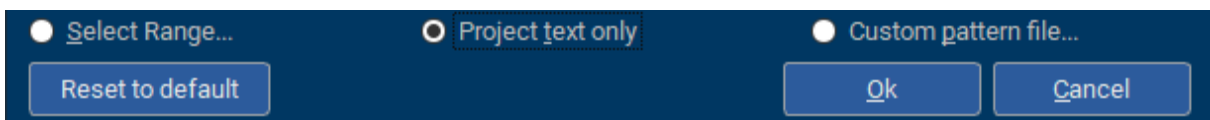
This option offers a convenient way to specify the codepoint ranges to be included in the font file:



Predefined Unicode ranges or user-defined ranges can be assembled.

Working with project text only

This option offers a way for including only those codepoints in the font which are used in the project:



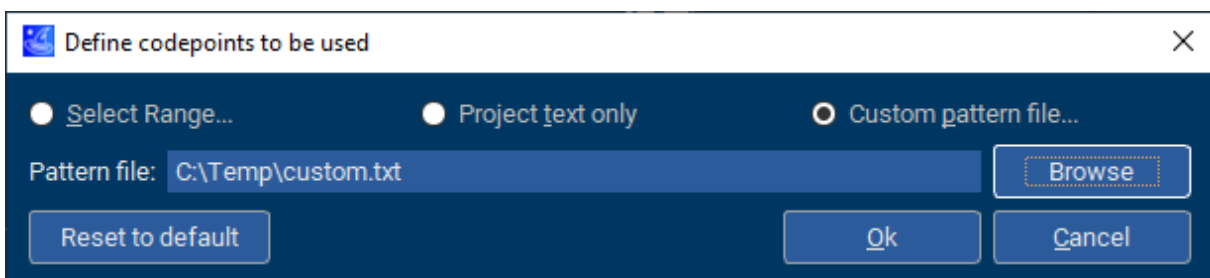
Those fonts will be recreated each time the project text changes.

Note

Fonts created with this option always include digits, decimal point, comma and space.

Working with pattern files

A pattern file is a file which contains all codepoints which should be used in the font. The following dialog is used to specify the pattern file to be used:



After choosing a pattern file it will be copied into the text resource folder of the project.

Note

Pattern files must have UTF-16 LE encoding.

6.3.2.4 'Delete from project'

Fonts, which are located in the projects font resource folder and which have no references can be deleted.

6.3.2.5 'Cleanup'

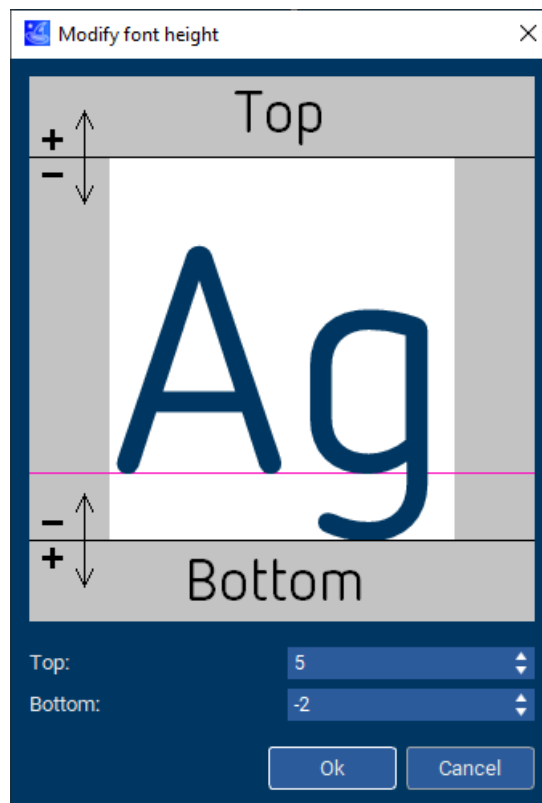
This option automatically removes all font files from the projects font resource folder which have no references.

Note

No undo is possible here.

6.3.2.6 'Modify font height...'

This option allows adding or removing pixels to/from bottom/top of a font file. The following dialog is used:



This option only exists for files which are located in the font resource folder of the project.

6.3.2.7 'Merge...'

Merging of fonts can become very important when creating multilingual applications. Unfortunately, most freely available fonts do not contain all characters needed for multilingual applications. Because of that we added the option for merging font files.

Here are some rules that must be met by the font files that are to be merged:

- All fonts to be merged together have to be of the same type.
- They must have the same height and the same baseline.
- Overlapping code points are not allowed.

To select the fonts which should be merged together simply select them all in the font repo dialog.

Note

Please keep in mind that the rules must be fulfilled by all fonts. Multiple font selection does only work if the rules are fulfilled.

If the height (or baseline) of the fonts to be merged are slightly different 'Modify font height...' can be used to adjust height (and baseline).

6.3.2.8 'Default codepoint range...'

Those codepoints are used when creating a new font. Can be changed later with 'Font codepoint range...'

6.3.3 Example how to use merged fonts

To clarify the procedure for merging fonts in more detail, here is an example that shows how to merge completely different fonts. And how to (optionally) make sure that only those characters are included in the font files that are actually referenced by the project. In this example we explain how to compile one font file that supports **Thai**, **Chinese**, **English** and **Arabic** by merging 4 fonts.

And we make sure, that this font file is automatically adapted when the project text changes to make sure, that **it always only contains the minimum of codepoints required for the project**.

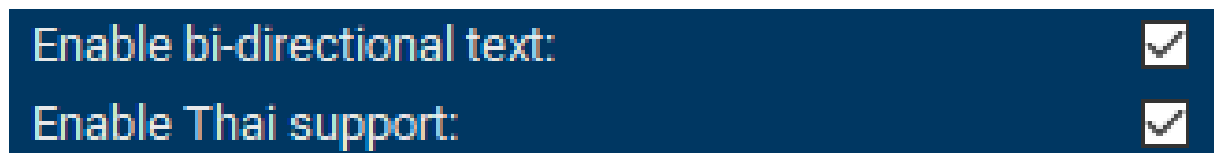
This example (`HowTo_MergeFonts`) is available in the examples under 'File/Open example...':



The following steps show how to create the merged font used by this example.

6.3.3.1 Setting up the project to be able to use Thai and Arabic

To be able to show Thai and Arabic text those features need to be enabled in the project settings. To do so please select 'Project/Edit options' and activate the corresponding check boxes:



6.3.3.2 Selecting the fonts to be merged together

First of all, we need fonts containing all the characters we need:

- **Itim** for Thai.
- **Noto Sans SC** for Chinese.
- **Roboto** for English.
- **Cairo** for Arabic.

To be able to reproduce the example the host system should have installed those fonts.

Note

These fonts can be found freely under fonts.google.com.

6.3.3.3 Generating the font files to be merged

We will create separate font files for the different code point ranges. To do this, we need to specify the ranges to be used for each individual font file.

For **Roboto** (used for English) we need to define the the following ranges:

<input type="radio"/> Select Range... <input checked="" type="radio"/> Project text only <input type="radio"/> Custom pattern file...				
+/-	First	Last	Enable/Disable	Range
–	0x0000	0x007F	Enable	Basic Latin
–	0x0080	0x00FF	Enable	Latin-1 Supplement
–	0x0100	0x017F	Enable	Latin Extended-A
–	0x0180	0x024F	Enable	Latin Extended-B

Ok, the Latin-extensions are not required for English. But on the other hand it makes things easy if for example umlauts or similar codepoints are used.

To create the first file use 'Default codepoint range...' to select the above ranges and then use 'Create new..' for creating a new font file. The height should be in our example 32 pixels. 'Style' should be 'normal' and 'Type' should be '4bpp antialiased'. Important is that the other fonts to be merged should use the same presets. After that we should see a new font file similar to this:

Family	Stock	Prj.	Height	Basel.	Ref.	Size	
Roboto_32_Normal_EXT_AA4_Cust	-	•	32	25	0	54.478	

For **Cairo** (used for Arabic) we need to define the the following ranges:

<input type="radio"/> Select Range... <input checked="" type="radio"/> Project text only <input type="radio"/> Custom pattern file...				
+/-	First	Last	Enable/Disable	Range
–	0x0600	0x06FF	Enable	Arabic
–	0x0750	0x077F	Enable	Arabic Supplement
–	0x08A0	0x08FF	Enable	Arabic Extended-A
–	0xFB50	0xFDFF	Enable	Arabic Presentation Forms-A
–	0xFE70	0xFEFF	Enable	Arabic Presentation Forms-B

Please keep in mind that for drawing Arabic text the representation forms are very important. Without those representation forms emWin is not able to draw Arabic text. To create the font file for the AppWizard setup the above ranges with 'Default codepoint range...' and then create a new font file with same presets as we used for the first file.

For **Noto Sans SC** (used for Chinese) we need the following ranges:

<input type="radio"/> Select Range... <input checked="" type="radio"/> Project text only <input type="radio"/> Custom pattern file...				
+/-	First	Last	Enable/Disable	Range
–	0x4E00	0x9FFF	Enable	CJK Unified Ideographs
–	0xFF00	0xFFEF	Enable	Halfwidth and Fullwidth Forms

Please note that creating the Chinese font file will take a while.

And the last one is **Itim**. Here we need the following range for Thai:

<input type="radio"/> Select Range... <input checked="" type="radio"/> Project text only <input type="radio"/> Custom pattern file...				
+/-	First	Last	Enable/Disable	Range
–	0x0E00	0x0E7F	Enable	Thai
+				

After creating the font file we should have a collection similar to this:

Family	Stock	Prj.	Height	Basel.	Ref.	Size	
Cairo_32_Normal_EXT_AA4	-	•	32	22	0	411.266	■
Itim_32_Normal_EXT_AA4	-	•	32	25	0	10.745	■
NotoSansSC_32_Normal_EXT_AA4	-	•	32	26	0	5.135.640	■
Roboto_32_Normal_EXT_AA4_Cust	-	•	32	25	0	54.478	■

Note

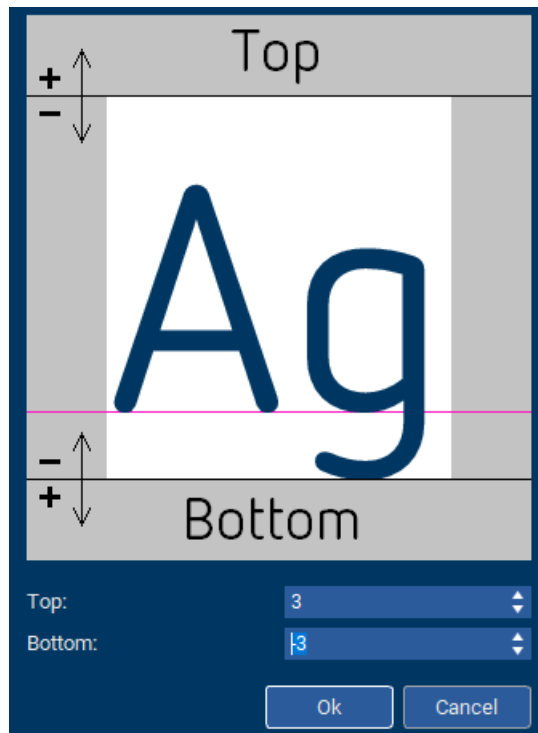
Please keep in mind that the ranges used to create font files are stored within the font files itself.

The button 'Font codepoint range...' can be used to show (and change) those codepoint definitions. Further the codepoint definition dialog contains a button 'Take...'. This button can be used to take the definitions from an existing font file as default for new font files.

6.3.3.4 Make the new fonts 'mergeable'

Please remember the rules for merging explained above. The fonts to be merged have to have the same height and the same baseline. But as shown in the font repository the Cairo font and NotoSansSC have a different baseline than Roboto and Itim. In typography, the baseline is the imaginary line on which the letters in a font appear to rest. The value is the distance from the top of the font.

In our case we leave Itim and Roboto as they are and move the baselines from Cairo 3 pixels down and NotoSansSC one pixel up. To do that select the font to be changed and then use the button 'Modify font height...'. The baseline can be easily moved by adding or removing the same number of pixels from the top and bottom of the font. Adding 3 pixels to the top and removing the same amount from the bottom should do the trick and move the baseline 3 pixels down:

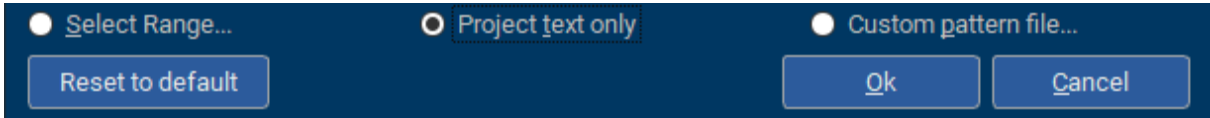


The next step is moving the baseline from NotoSansSC one pixel up. To do so please remove one pixel from the top and add one pixel to the bottom.

After that we should be able to (multi)select all 4 font files simultaneously.

6.3.3.5 Include only the codepoints used in the project

The fonts we have created until here include fixed codepoint ranges. And especially the Chinese font file is quite big. To make sure that the end result contains only the codepoints which are used by the text in the project we have to change the default codepoint range to 'Project text only'. It is important to do that before creating the merged font file:



Now we can multiselect the 4 font files and use the button 'Merge...' to merge them together. Immediately after merging the font file it contains all characters of the merged font files which makes it also quite big:

Family	Stock	Prj.	Height	Basel.	Ref.	Size	
Cairo_32_Normal_EXT_AA4	-	•	32	25	0	411.267	
Itim_32_Normal_EXT_AA4	-	•	32	25	0	10.745	
Merged_32_Normal_AA4	-	•	32	25	0	5.345.349	
NotoSansSC_32_Normal_EXT_AA4	-	•	32	25	0	5.135.641	
Roboto_32_Normal_EXT_AA4_Cust	-	•	32	25	0	54.478	

But each time the text content of the project changes the font file will be (re)assembled, so that it includes only the characters we need for the text in the project.

Note

Recalculating the content of the font can also be triggered in the font repository directly by selecting the merged font, clicking on 'Font codepoint range...' and pressing the 'Ok' button.

If the project contains **Arabic text** and 'Project text only' is used for a font the AppWizard automatically makes sure, that the representation forms required for drawing the text are also included in that font automatically. If at least one single Arabic character is used the representation forms will be included automatically. After recalculating the size of the merged font will be only a fraction of the size before:

Merged_32_Normal_AA4	-	•	32	25	1	418.551	
----------------------	---	---	----	----	---	---------	--

Please keep in mind that the font files, which are only used for merging will not part of the output for the compiler. But each time the project text changes the content of those fonts will be used to reassemble the merged font.

6.3.3.6 Show result on the screen

To see the result on the screen please enter the text 'Hello World!' in all 4 languages.

Note

To get the right strings Google Translate (translate.google.com) can be used. Simply copy and paste the strings into the text editor of the AppWizard.

After entering the text and pressing 'Ok' the AppWizard recalculates the content of those fonts which are created with the codepoint option 'Project text only'. Now you can create 4 text objects and select the new font.

Or simply open the example 'HowTo_MergeFonts' and press <F5>.

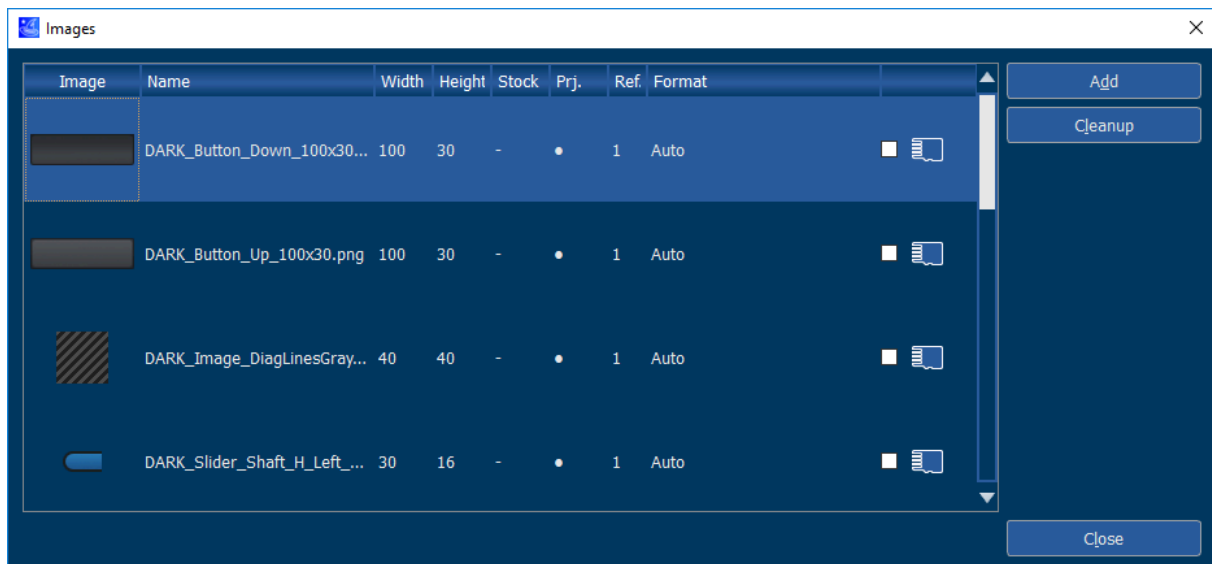
6.4 Image management

Note

Please note the naming rules for image files:

- The name must not start with a digit.
- The only valid characters are letters, numbers and underscores.

The image management dialog shows all images located in the image stock or the resource folder of the project.



The dialog shows the following columns:

- Image preview
- File name
- Width and height in pixels
- Stock image, means image is located in image stock
- Project image, means image is located in resource folder of the project
- Number of references
- Check box to specify SD card management for that image

The following options exist:

- Show stock images
- Show project specific images

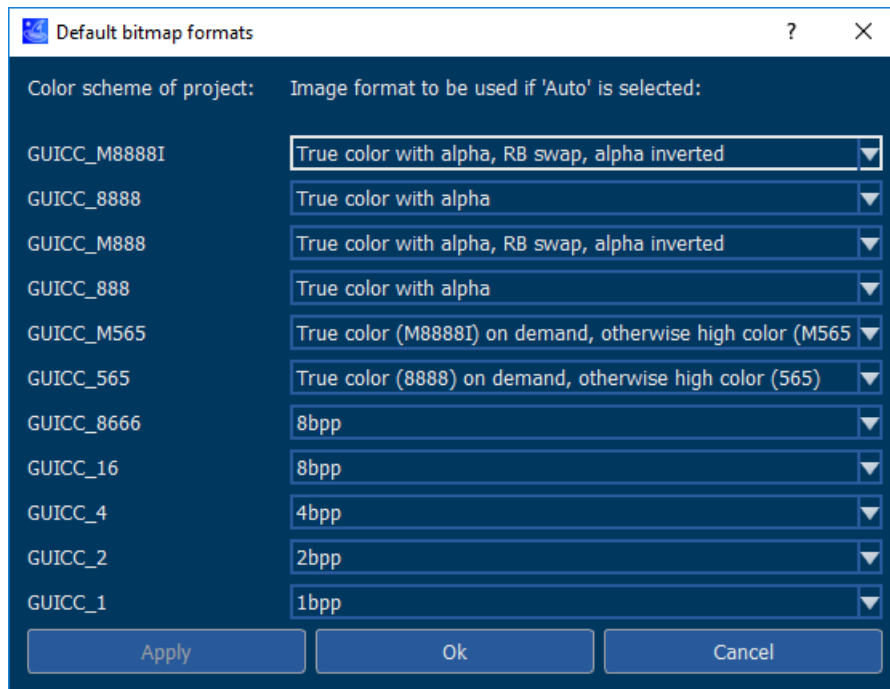
Note

The project folder will contain all images used in the project, this applies to stock images as well.

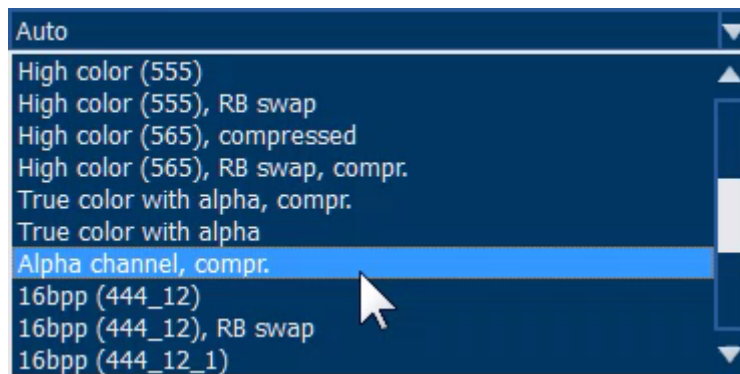
Bitmap format

When adding a new image, by default the format is set to 'Auto'. This will automatically choose the fitting bitmap format depending on which color format has been selected for the project. But the user may also select a specific bitmap format if the hardware requires it.

Additional to that the user has the option to select a completely different bitmap format by setting a format which should be used for the 'Auto' option. This can be done by opening the 'Default bitmap formats' dialog through the 'Preferences' menu.



Alpha channel bitmaps



This bitmap format allows bitmaps to be drawn in the current text color. With this format, only the alpha values of the individual pixels are stored. The RLE compression ensures an optimal use of the memory. The output is then in the text color set for the corresponding object.

Deleting images

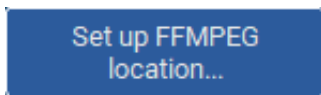
Images can be deleted from the project by clicking the **Delete from project** button, after selecting the image that should be deleted. Images can only be deleted if they haven't been referenced, that means the reference count shows zero.

6.5 Movie management

The movie resource window allows the user to add or remove movie files.

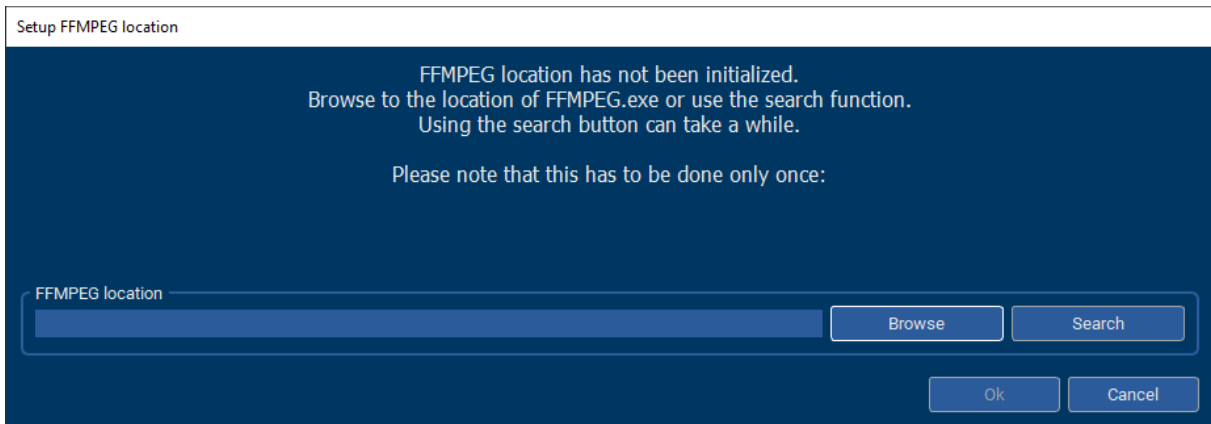


It is important to know that the movie support of AppWizard requires an external tool to convert movies to an emWin compatible format. That tool is FFmpeg which is free available under www.ffmpeg.org. If this tool is not yet set up, the 'Create new...' button is not yet visible.



Instead of that the button for setting up the FFmpeg location is shown.

If FFmpeg is not already available on the host PC it needs to be installed first. After that (or if FFmpeg is already available) the button can be used to open the dialog for setting up the location of FFmpeg.exe:



Search

This button starts an automatic search for the executable. Please note that the search can require a while. After an executable has been found that file can be used or the search can be continued.

Browse

If the location is known the browse button can be used to specify the right location.

Creating a movie file for AppWizard

After FFmpeg location has been set up the 'Create new...' button can be used to create a new movie file. It opens the following dialog:

File to be converted

Used to select the source file to be converted. Any video format supported by FFmpeg can be used here. No modifications will be done at this file.

Video quality

The quality parameter defines the `-q` parameter for FFmpeg. The lower the number, the better the quality of the generated movie and the larger the file size.

Frame rate

Defines the desired number of frames per second of the movie to be generated.

File format to be generated

AVI - **A**udio **V**ideo **I**nterleave format, a multimedia container format. The converted file will contain a motion JPEG movie.

EMF - **e**mWin **M**ovie **F**ormat, a more easy movie format only supported by emWin and the emWin movie player.

Resolution

The sizes in X and Y axis can be specified independently. Please note that the aspect ratio fits to the format of the source file which will be scaled to the given parameters.

Presets

Can be used to save the parameters for later operations.

Start conversion

Pressing the 'Ok' button starts the process of file conversion. A new movie file will be created in the movie resource folder of the project.

File name of generated file

The file name will be generated automatically:


```
<Baseline of source file>_<X-size>x<Y-size>.<Extension(emf or avi)>
```

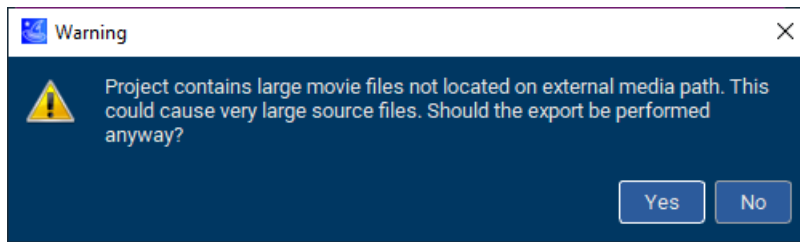
Please note that the process of file conversion can take a while.

6.5.0.1 Pitfalls with large files

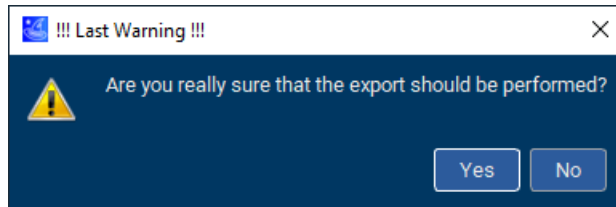
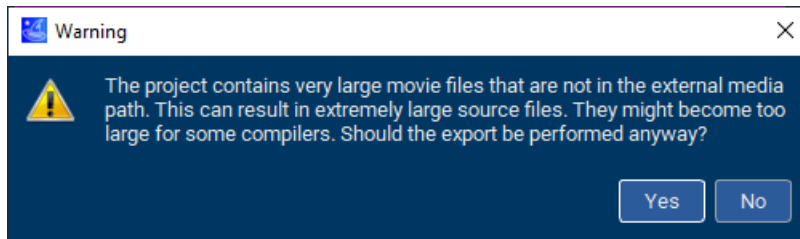


The column having the SD-card symbol defines how the movie should be used.

If it is deactivated the movie will be compiled into the code section. Please be aware that using large movie files without any external media location could cause problems. The resulting C files in that case can become very large. And maybe too large for some compilers. To inform the user that quite large files will be generated a warning will be generated if one of the internal movies exceeds the size of 512 KByte:



If one of the internal used movie files exceeds the size of 4 MByte the warning is slightly stronger:



This warning is required because each byte of the movie file generates 6 bytes of source code to be compiled. In case of a 4 MByte movie the result would be a more than 24 MByte large C file which is quite large for some compilers.

We recommend using an SD-card or similar media to store large movie files.



It can be done simply by activating the SD-card check box..

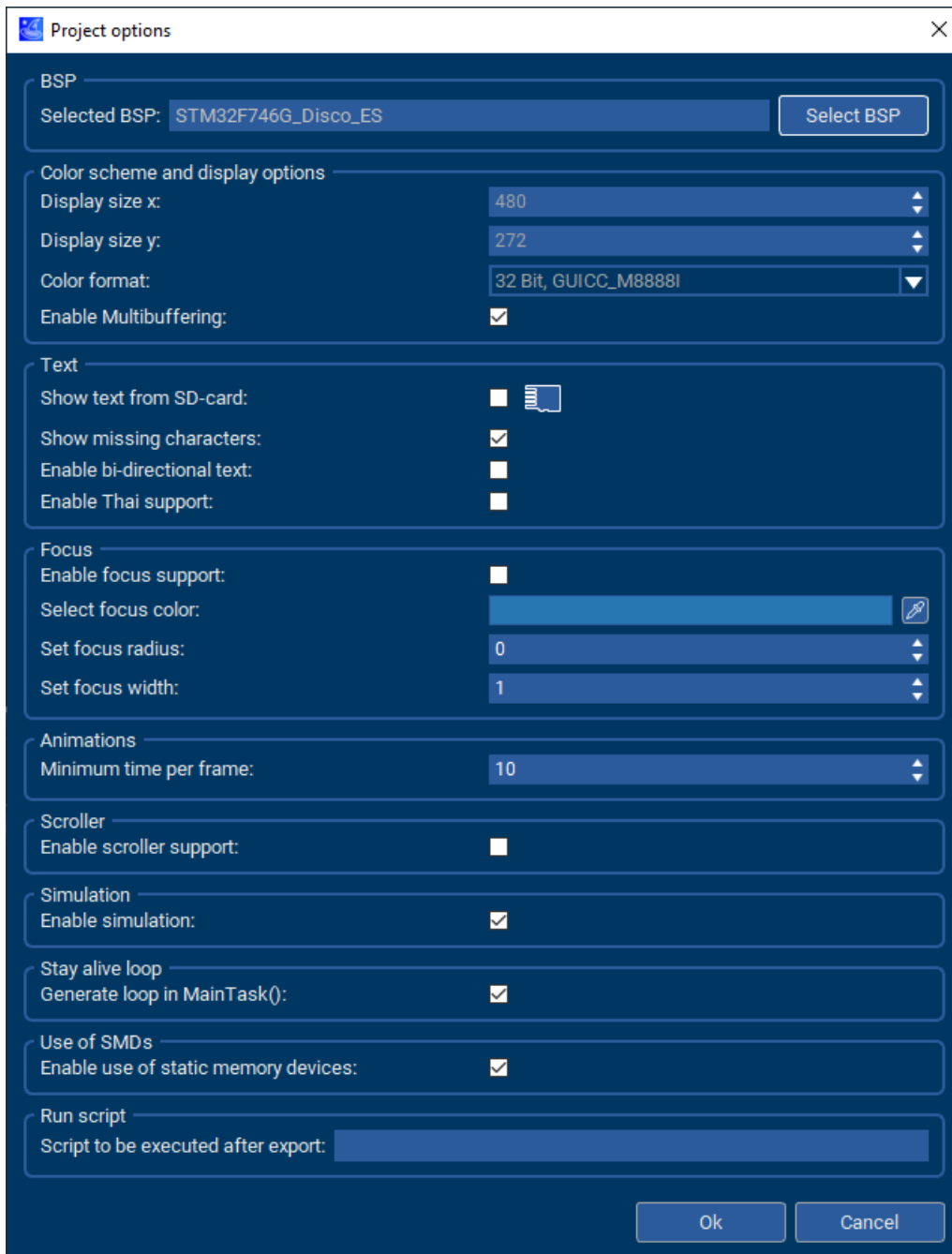
If it is activated the movie will be exported to the external media path when exporting the project. After setting up the external media path under 'Edit/Preferences/Media path' all external media files will be generated automatically in that path. The movies can be found then in the folder /Resource/Movie.

Chapter 7

Options / Preferences

7.1 Options

The options dialog is available under `Project / Edit options`.



The project options dialog has the following options:

BSP

- **Selected BSP:** Desired BSP for target hardware.

Color scheme and display options

- **Display size x:** Horizontal display size.
- **Display size y:** Vertical display size.
- **Color format:** Desired color format.
- **Enable Multibuffering:** Option whether or not multi buffering should be enabled in the project. This option is only available, if the selected BSP supports multi buffering.

Text

- **Show text from SD-card:** Option to outsource the texts to external media.

- **Show missing characters:** Shows placeholders for missing characters.
- **Enable bi-directional text:** Enables support of bi-directional texts.
- **Enable Thai support:** Enables support of Thai script.

Focus

- **Enable focus support:** Enables the ability for objects to receive focus.
- **Select focus color:** Color used for drawing the focus rectangle.
- **Set focus radius:** Radius used for the corners of the focus rectangle.
- **Set focus width:** Line width of the focus rectangle.

In case of using a BSP display size and color format are fixed and come from the BSP.

Focus rectangles are explained later on in the chapter *Object focus* on page 125.

Animations

- **Minimum time per frame:** Minimum time of one animation slice in ms.

Scroller

- **Enable scroller support:** Activates automatic use of scroller objects.

7.1.1 Enable simulation

Per default the AppWizard generates a simulation project for Visual Studio.

Under certain circumstances it can be useful to deactivate this option. This can be the case if the project structure does not allow Visual Studio and/or Windows specific c-files in this location or if the simulation is simply not required.

Please note that if no simulation project is generated the emWin SPY can not be used.

7.1.2 Stay alive loop

If this option is enabled the AppWizard will generate an endless loop in the function `MainTask()` found in `APPW_MainTask.c`.

In some cases it may be necessary to disable this option. For example, if there is already another endless loop in the application which handles other parts of the application.

Please note that if this option is disabled the user has to take care of calling `APPW_Exec()` periodically to keep the AppWizard application alive.

7.1.3 Use of static memory devices

This is an option to accelerate the process of drawing.

7.1.3.1 Requirements

It makes sense if a hardware acceleration function for drawing 32 bpp memory devices is available. Further it requires a large amount of RAM, because each window gets its own memory device with a color depth of 32 bpp.

7.1.3.2 How it works

If a window / object needs to be (re)drawn, emWin first creates a static memory device with the content of the window. After that the memory device is drawn instead.

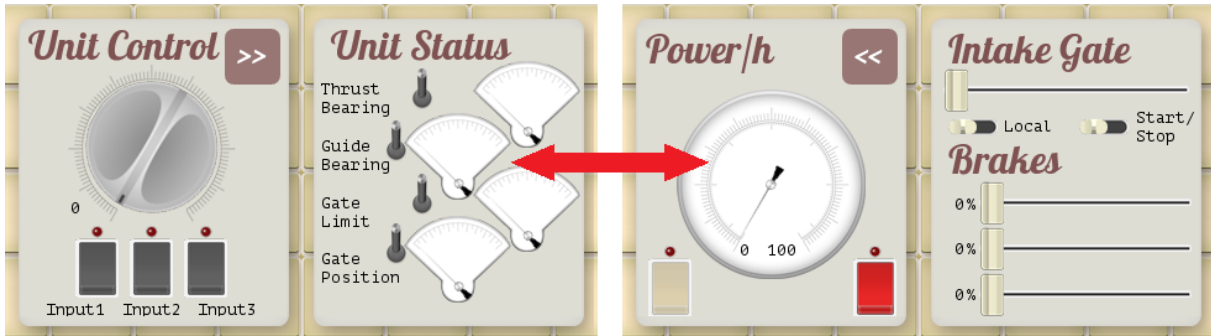
When a screen or a window is swiped by the touch screen, the memory device does not need to be rebuild again. Only the already existing memory device will be drawn at the new position.

This makes the most sense if the project contains hardware acceleration for drawing 32 bpp memory devices. If the system becomes short of RAM unused memory devices will be deleted automatically.

7.1.3.3 Performance

Depending on the content of a screen, using static memory devices can improve the performance significantly. The more complicated the drawing of the objects is, the more performance gain can be achieved.

Here are the example screens of the performance test:



We measured the following performance gain for swiping operations when enabling static memory devices:

	CPU-Load	Frames per second
Static memory devices disabled	57%	8
Static memory devices enabled	24%	16

We used an iMXRT1060 for this test. It shows clearly that we halved the CPU-load by doubling the framerate with this option.

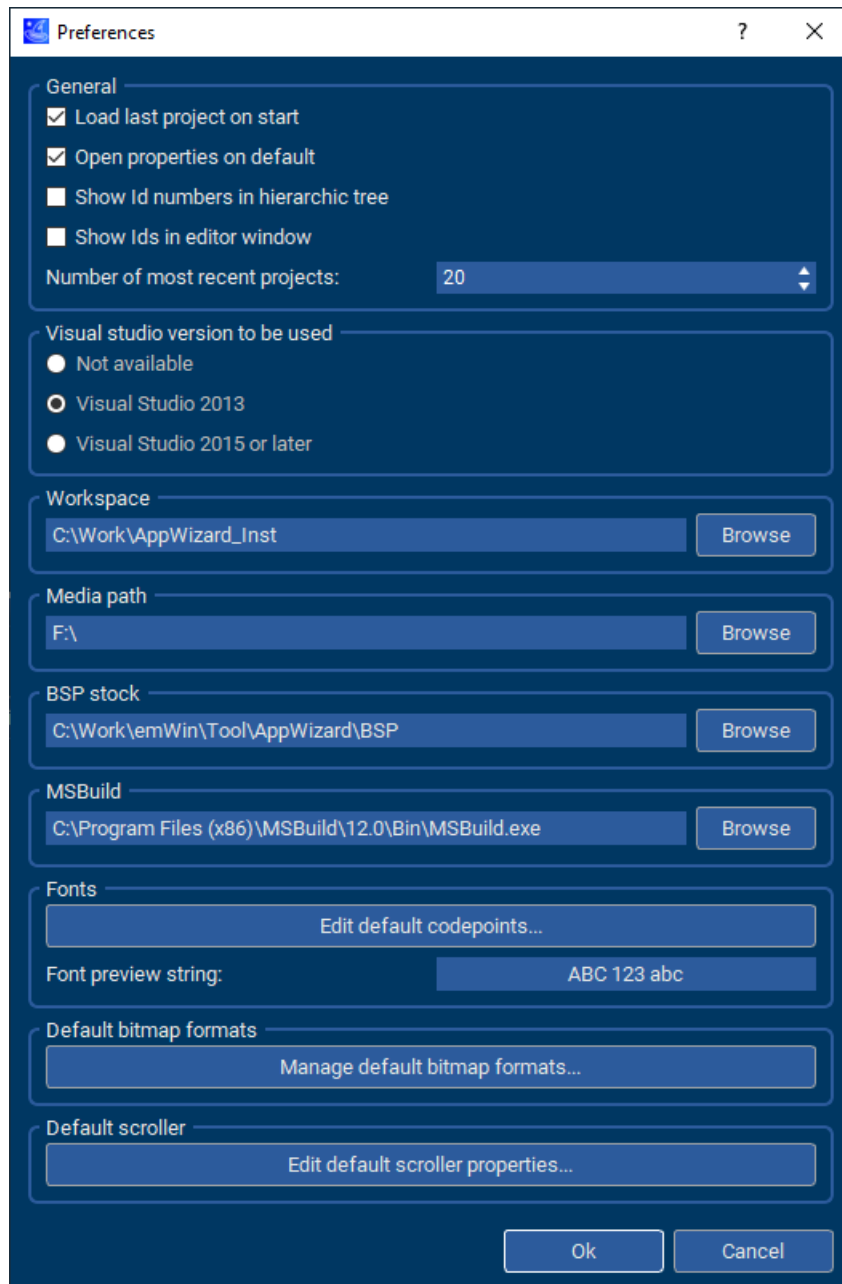
7.1.4 Run script

A path can be set which points to a script which will be executed after the Export and save operation of the AppWizard.

Such a script file can be used to automatically compile the generated output or copy the output to a specific location.

7.2 Preferences

The preferences dialog is available under `Edit / Preferences`.



The preferences dialog has the following options:

- **Load last project on start:** Enables loading the last used project after starting the application.
- **Open properties on default:** Opens all object properties by default.
- **Show Id numbers in hierarchic tree:** Shows the Window Manager Id next to the AppWizard Id.
- **Show Ids in editor window:** Displays the object's ID in the editor window.
- **Number of most recent projects:** Defines the maximum number of projects in the most recent file list.
- **Simulation project to be used:** Select which Visual Studio simulation project should be used when pressing F6. If you are using Visual Studio 2010-2013 select the first option. If using a later version select the second option.
- **Workspace:** Path to be used to open examples.
- **Media path:** Path to external media.
- **BSP stock:** Sets a path, where custom BSPs are located.
- **MSBuild:** MSBuild to be used for AppWizard SPY.

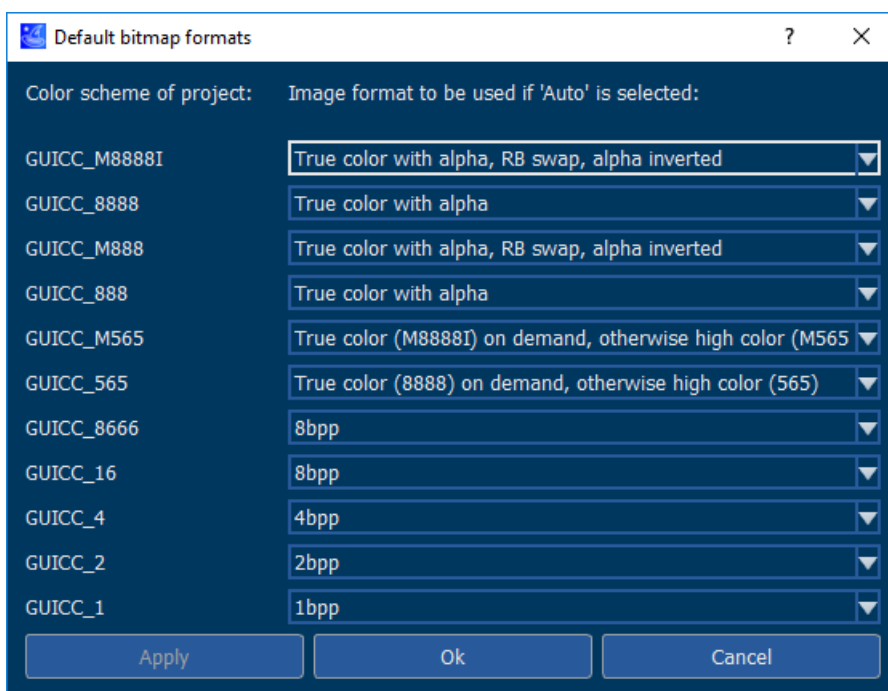
- **Edit default codepoints...:** Editing the default range of code points to be used for new fonts.
- **Font preview string:** This string is used to generate the font preview string.
- **Default bitmap formats:** Set the default bitmap format to be used when bitmaps use the 'auto' format.
- **Default scroller:** Set the default scroller format to be used when bitmaps use the 'auto' format.

Default codepoints

The user can define which characters should be included in newly created fonts. A more detailed description and explanation on the codepoint dialog can be found under *Font resource window* on page 46.

Default bitmap formats

The user has the option to select a default bitmap format which will be used by the 'Auto' format option when a bitmap is added to the project.



Chapter 8

Objects





















8.1 Introduction






The objects the AppWizard supports are similar to the widgets in emWin. The following table gives an overview about the currently available objects in the AppWizard.

Note

Some of the objects are based on emWin widgets. For more information refer to the document **UM03001 emWin User Guide & Reference Manual**.

Name	Symbol	Description
Box		Box object that can be colored or filled by a gradient and optional drawn with rounded corners.
Button		Clickable button object, can also be used as toggle button.
Checkbox		Checkbox with up to 3 states.
Dropdown		Allows the user to choose one value from a 'drop down' list.
Edit		Edit field for user input. It also supports a decimal mode for entering numbers.
Gauge		Radial progress bar.
Image		Object that displays an image. Supports direct drawing of GIF, BMP, JPEG and Bitmap files.
Keyboard		Screen keyboard for entering text or numbers. Various pre-defined layouts can be used.

Name	Symbol	Description
Listbox		Showing and selecting text from simple text lists.
Listview		Table view containing multiple columns of text and a header line.
Movie		Movie object for playing movies.
Multiedit		Edit field for multiple lines of user input.
Progbar		Progressbar to display the progression of a process.
QRCode		Displays a QR code.
Radio		Radio object.
Rotary		Circular object that can be rotated.
Screen		A screen serves as a parent for all other objects.
Slider		Movable slider.

Name	Symbol	Description
Switch		Toggleable switch with two states and an optional fade mode.
Text		An object displaying text. A decimal- and a hexadecimal mode for showing numbers is also available.
Timer		Timer object.
Window		Similar to screen object, serves as a parent object for other objects. Supports an opaque mode with an optional background color.
Wheel		Swipeable rotating wheel of items with texts and/or bitmaps.

8.2 Object properties

Every object has its own properties than can be edited. The following section lists all common properties that are used in multiple objects. Object-specific properties are explained in the corresponding object section.

This table lists the common properties and provides links to its corresponding chapter with more explanation.

Property	Description
Alignment	Alignment of an object.
Bitmap	Bitmap to be shown in an object.
Border size	Size of border for an object.
Color and background color	Foreground and background colors for an object.
Content	List- and table content for text based objects.
Cursor inversion	When disabling cursor inversion, the color for the cursor isn't the inverted background color.
Decimal mode	Makes an object only eligible for decimal digits.
Focus options	Disables the focus for an object or hides it.
Font	Font for the object.
Frame radius	Radius of frame around an object.
Frame size	Pixel-size of frame around an object.
Hexadecimal mode	Makes an object only eligible for hexadecimal digits.
Horizontal mode	Changes an object to be horizontal.
Motion partner	Motion settings for swiping between screens.
Motion support	Settings for swiping content.
Movie	Movie to be shown.
ID	ID for the object.
Initial value	Initial value for an object.
Invert direction	Inverts direction of an object.
Opaque mode	Removes the transparency flag of an object.
Overwrite mode	Overwrite mode for text cursors of an object.
Period	Movement period.
Position and size	Position and size of an object.
Radius	Radius of an object.
Range	Range of position values of an object.
Space	Spacing.
Space between items	Additional spacing between items.
Span of values	Set range of values of an object.
Stay on top	Makes sure a screen is displayed on top of all other screens.
Text	Text to be shown.
Text color	Color for text.
Text rotation	Rotation mode for text.
Text wrapping	Enables text wrapping.
Tiling	Tiling mode for bitmaps.

Property	Description
<code>Untouchable</code>	Screen will not be able to receive touch input.
<code>Vertical mode</code>	Changes an object to be vertical.

8.2.1 Alignment

Description






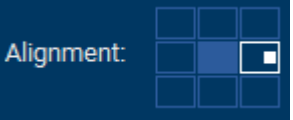


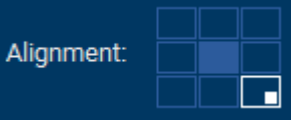
The alignment property allows to choose a combination of a horizontal alignment and a vertical alignments. This property can be set for bitmaps and texts.

Available objects

This property can be set for the following objects:

- *Button* object
- *Dropdown* object
- *Edit* object
- *Gauge* object
- *Multiedit* object
- *Listbox* object
- *Listview* object
- *Text* object

Usage

Combined with	Horizontal left	Horizontal center	Horizontal right
Vertical top	Alignment: 	Alignment: 	Alignment: 
Vertical center	Alignment: 	Alignment: 	Alignment: 
Vertical bottom	Alignment: 	Alignment: 	Alignment: 

You may also add an x and y offset to the object.

Offset x:

Offset y:

8.2.2 Bitmap

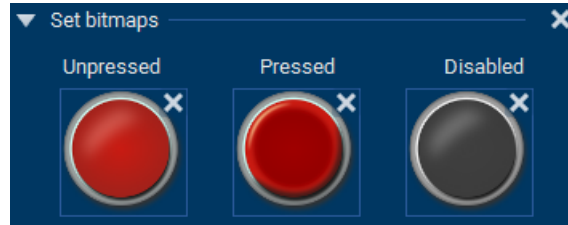
Description

The bitmap property allows to set a bitmap to a specific purpose or state of an object.

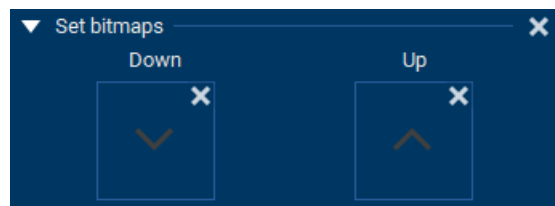
Available objects

This property can be set for the following objects:

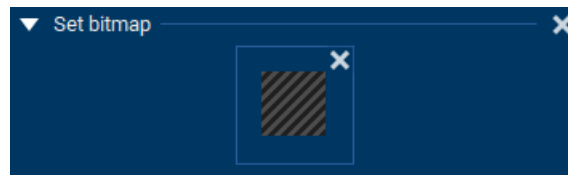
- *Button* object



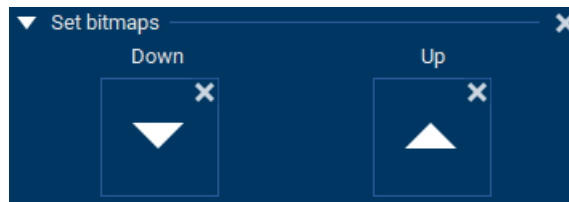
- *Dropdown* object



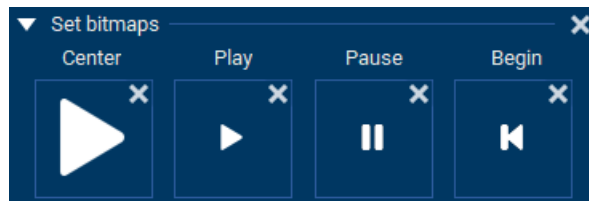
- *Image* object



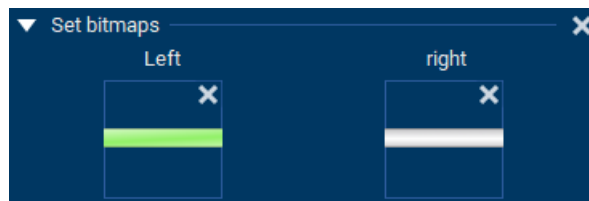
- *Listview* object



- *Movie* object

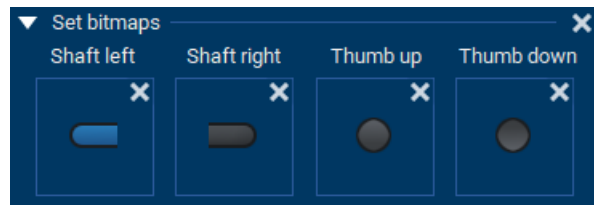


- *Progbar* object



- *Slider* object

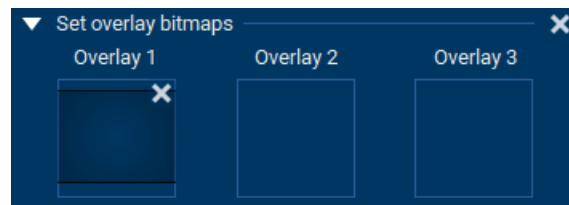
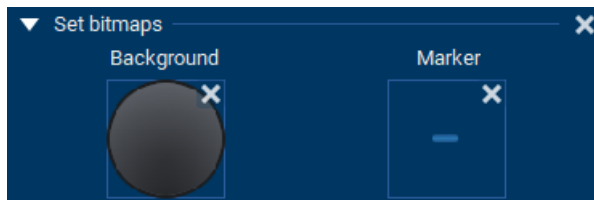
- *Switch* object



- *Rotary* object



- *Wheel* object



8.2.3 Border size

Description

This property sets the border size of an object in pixels. The border is the spacing between the frame and the text.



Available objects

This property can be set for the following objects:

- *Edit* object
- *Multiedit* object
- *Wheel* object

8.2.4 Colors

Description

Colors can be set for several individual object properties:

- Text,
- Images (alpha images only),
- Background,
- Frame,
- Gridlines,
- ...and more.



Available objects

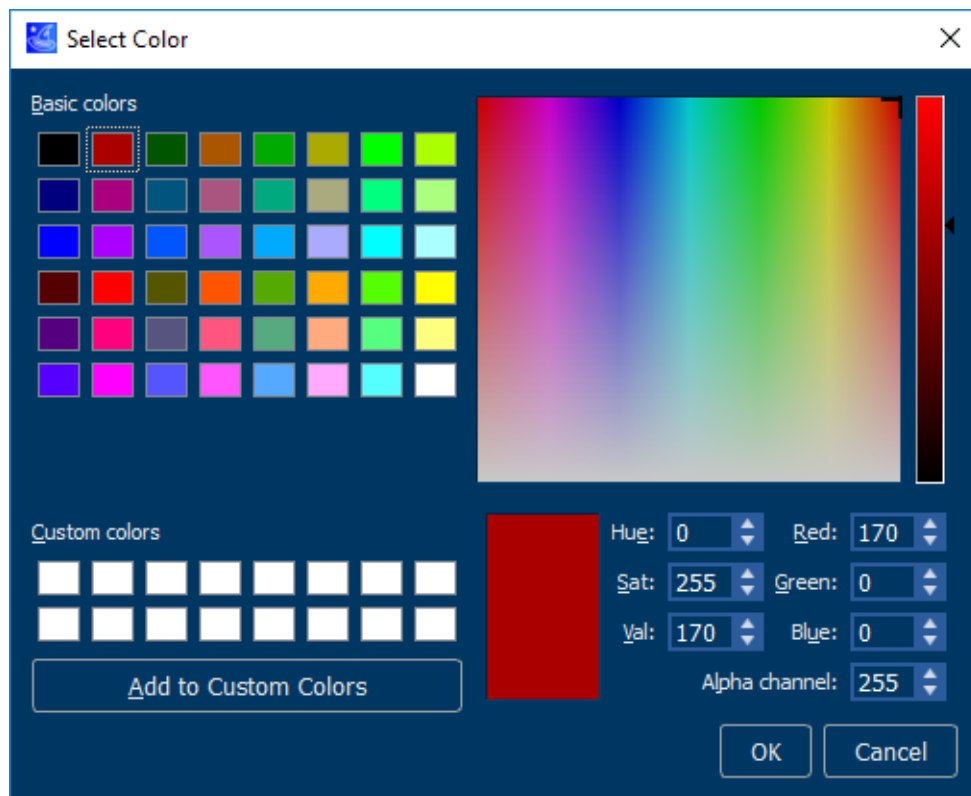
Color properties can be set for the following objects:

- *Box* object
- *Button* object
- *Dropdown* object
- *Edit* object
- *Gauge* object
- *Image* object
- *Keyboard* object
- *Listbox* object
- *Listview* object
- *Movie* object
- *Multiedit* object
- *Progbar* object
- *Wheel* object

Usage

When selecting the color, a dialog is opened. This dialog allows to set a specific color by setting RGB and HSV values, as well as the alpha value.

You may also save custom colors.



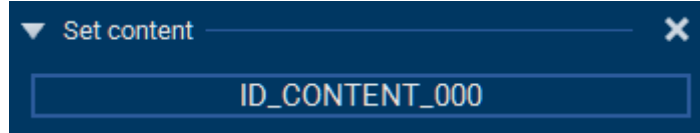
Related topics

- Text color

8.2.5 Content

Description

The term 'Content' means lists and tables for text based objects. A 'list' means a simple array of strings. A 'table' consists of multiple columns and a header line. Column size, text alignment and content of the header line is also part of a table definition.



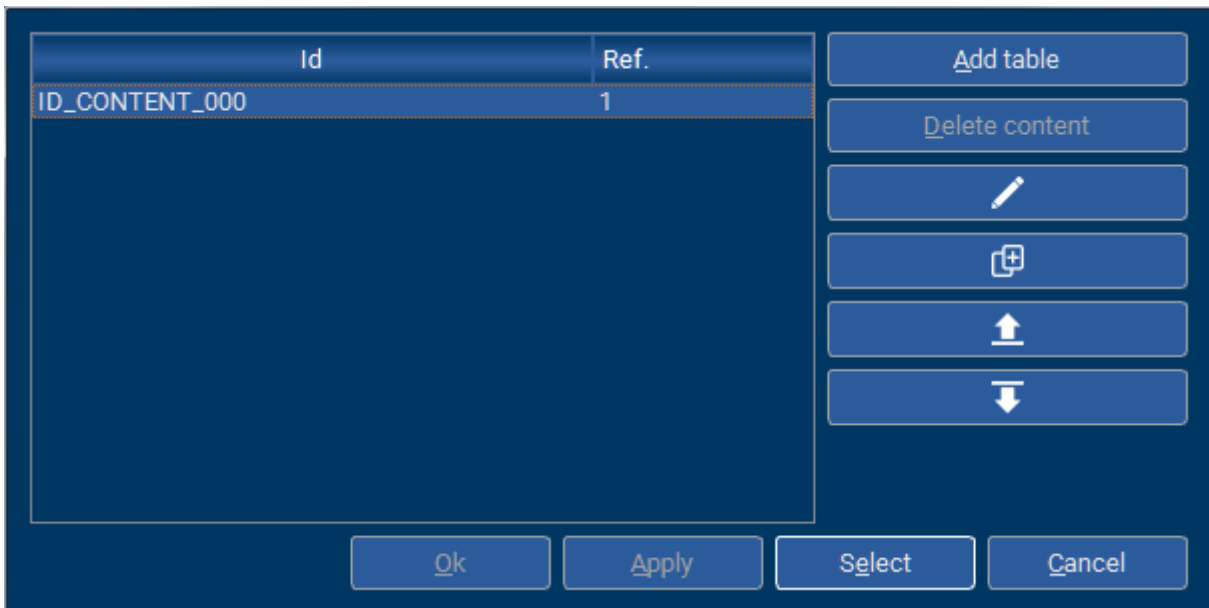
Available objects

This property can be set for the following objects:

- *Dropdown* object
- *Listbox* object
- *Listview* object
- *Wheel* object

Usage

When opening the content selection, a dialog with a list of the currently existing content is shown. It allows selecting existing or creating new content.



In dependence of the underlying object either tables or lists can be selected/created. For example LISTBOX- and DROPDOWN-objects require lists and the LISTVIEW-object requires tables. When clicking the edit pen or the 'Add table/list' button the content definition dialog opens:



This dialog is used to edit existing content and/or to create new content. Lists consist of simple text arrays. Tables consist of several columns. For each column the width and the alignment can be defined as well as the content of the header.

Using text from text resources

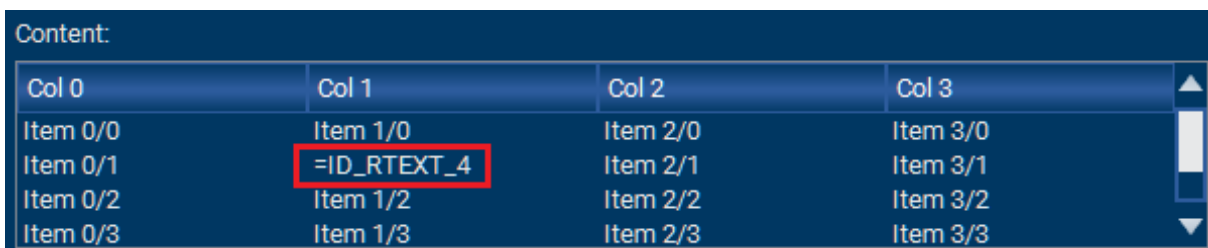
The content objects also support using text from the text resources. This can be achieved by the following mechanism: Simply enter an equal sign followed directly by the name of the desired text resource.

Example

Assuming the text resources contains an item named `ID_RTEXT_4`:

Id	Obj. Ref.	d
ID_RTEXT_4	2	Example

To use it in a table, in a list or in the header line of a table, use an equal sign with the resource name:



Importing and exporting content

A detailed description on how lists and tables can be imported and exported can be found under *Importing and exporting of content* on page 53.

Setting table- or list content by custom code

After creating an object containing a list or a table an `APPW_MSG_GET_CONTENT` message is send to the screen callback routine. The element 'hWinSrc' of the message structure is

(ob)used for passing the object id of the object requesting the content. To pass content to the requesting object simply set `Data.p` to an `APPW_CONTENT` structure. Here is a code excerpt which shows how to realize that:

```
static const char * ID_CONTENT_Left_Col_0[] = {
    "Left Item 0",
    "Left Item 1",
    "Left Item 2",
    "Left Item 3",
    "Left Item 4",
    "Left Item 5",
    "Left Item 6",
    "Left Item 7",
    "Left Item 8",
    "Left Item 9",
};

static const char ** ID_CONTENT_Left_Text[] = {
    ID_CONTENT_Left_Col_0,
};

const APPW_CONTENT ID_CONTENT_Left_Data = {
    ID_CONTENT_Left_Text,
    GUI_COUNTOF(ID_CONTENT_Left_Col_0)
};

static const char * ID_CONTENT_Right_Col_0[] = {
    "Right Item 0",
    "Right Item 1",
    "Right Item 2",
    "Right Item 3",
    "Right Item 4",
    "Right Item 5",
    "Right Item 6",
    "Right Item 7",
    "Right Item 8",
    "Right Item 9",
};

static const char ** ID_CONTENT_Right_Text[] = {
    ID_CONTENT_Right_Col_0,
};

const APPW_CONTENT ID_CONTENT_Right_Data = {
    ID_CONTENT_Right_Text,
    GUI_COUNTOF(ID_CONTENT_Right_Col_0)
};

void cbID_SCREEN_00(WM_MESSAGE * pMsg) {
    switch (pMsg->MsgId) {
        case APPW_MSG_GET_CONTENT:
            switch (pMsg->hWinSrc) {
                case ID_DROPDOWN_00:
                    pMsg->Data.p = &ID_CONTENT_Left_Data;
                    break;
                case ID_DROPDOWN_01:
                    pMsg->Data.p = &ID_CONTENT_Right_Data;
                    break;
            }
            break;
    }
}
```

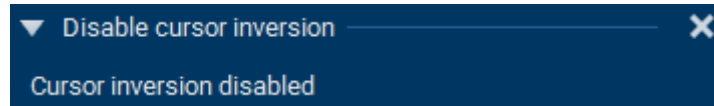
The sample library also contains a sample which shows how to set custom dropdown text.

8.2.6 Cursor inversion

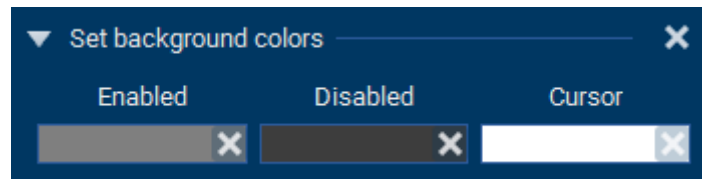
Description

Cursor inversion defines the drawing behavior of the cursor in an Edit or Multiedit. If cursor inversion is enabled, the cursor will be drawn by inverting the background and text color at its position.

If cursor inversion is disabled, the user can pick a custom cursor color instead.



The user can set the color of the cursor with the Background color property under 'Cursor'.



Available objects

This property can be set for the following objects:

- *Edit* object
- *Multiedit* object

8.2.7 Decimal mode

Description

Using the decimal mode makes it possible to show digits instead of characters. The mask determines how exactly the digits should be shown. It determines if a sign should be shown in any case, if leading zeroes should be shown, the number of digits and the location of an optional decimal point. Also, when using decimal mode, a range property is added to the object to limit the numbers that can be entered.

Available objects

This property can be set for the following objects:

- *Edit* object
- *Text* object

Additional information

The mask being set for an Edit or Text object behaves slightly different.

The Edit object accepts either a single '0' or a single '#' as mask. A '0' shows leading zeros depending on the range set for this object.

The Text object shows as many leading zeros as '0' are being used as mask. The number of '#' used indicates the maximum number of digits.

Examples (Text object)

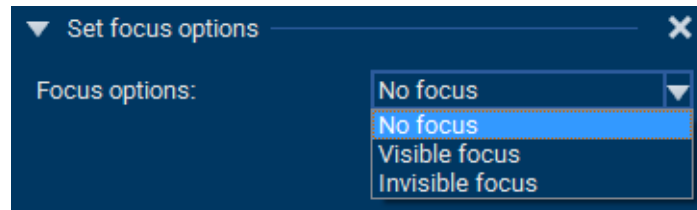
The following table shows some examples with mask, value to be shown and drawing result:

Mask	Value	Result
+00.00	987	+09.87
+00.00	0	00.00
+00.00	-100	-01.00
+00	-1	-01
+00	1	+01
00	-1	-1
00	1	01

8.2.8 Focus options

Description

Focus options for an individual object. The focus rectangle can be shown or hidden or the ability to receive input focus can be disabled altogether.



Available objects

This property can be set for the following objects:

- *Button* object
- *Dropdown* object
- *Edit* object
- *Listbox* object
- *Listview* object
- *Multiedit* object
- *Rotary* object
- *Slider* object
- *Switch* object

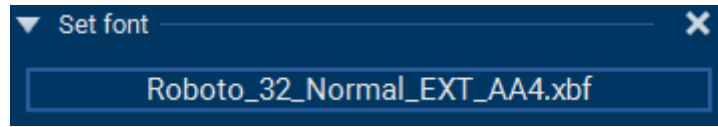
Additional information

Focus support has to be enabled in the project options. More information about input focus can be read under *Object focus* on page 125.

8.2.9 Font

Description

The font property allows to set a font to an object.



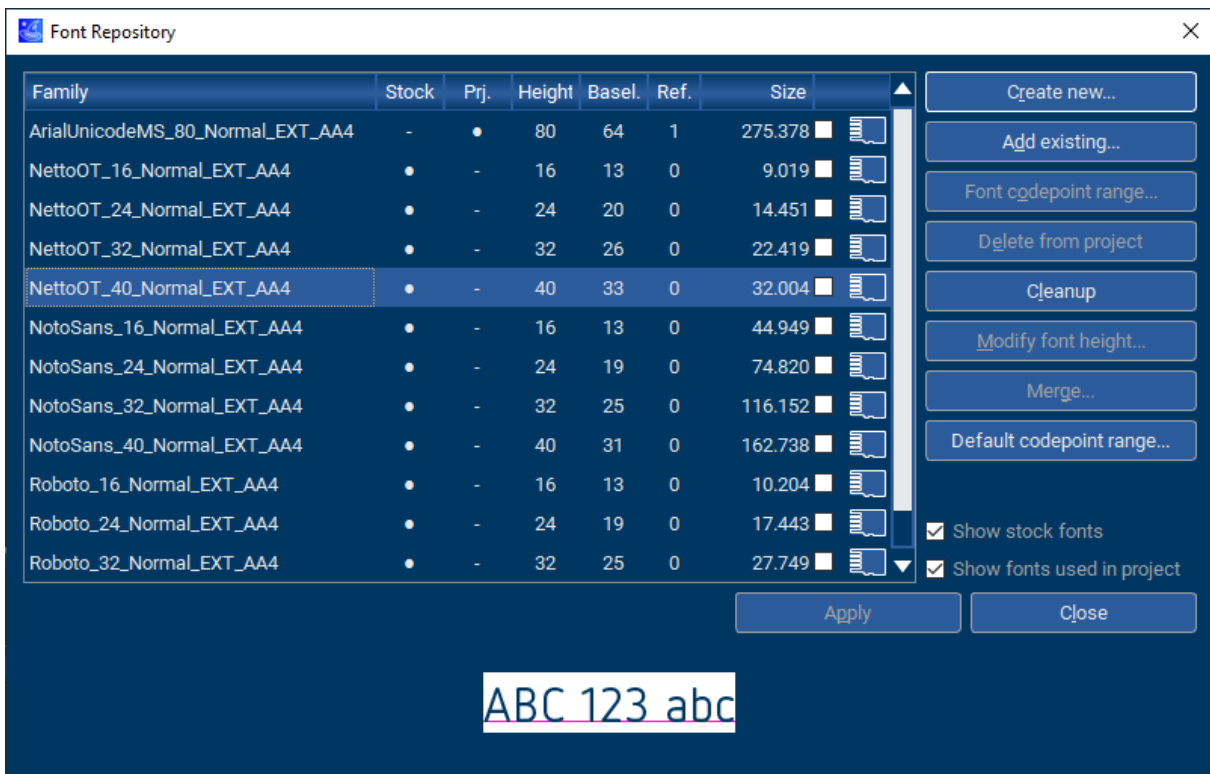
Available objects

This property can be set for the following objects:

- *Button* object
- *Dropdown* object
- *Edit* object
- *Keyboard* object
- *Multiedit* object
- *Switch* object
- *Text* object
- *Wheel* object

Usage

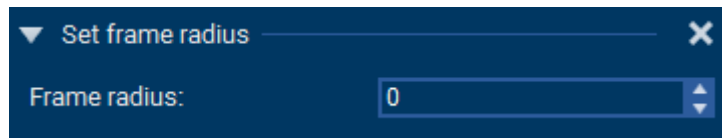
Choosing a font will open a dialog showing all fonts available in the project.



8.2.10 Frame radius

Description

This property sets the radius of the frame drawn around a given object.



Available objects

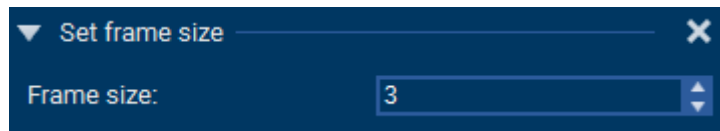
This property can be set for the following objects:

- *Edit* object
- *Listview* object
- *Multiedit* object
- *Wheel* object

8.2.11 Frame size

Description

This property sets the width of an object's frame in pixels.



Available objects

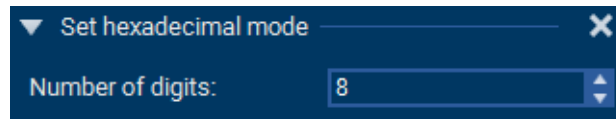
This property can be set for the following objects:

- *Dropdown* object
- *Edit* object
- *Listbox* object
- *Listview* object
- *Multiedit* object
- *Progbar* object
- *Wheel* object

8.2.12 Hexadecimal mode

Description

Hexadecimal mode makes an object only eligible for hexadecimal digits. The number of shown digits has to be set via the property.



Available objects

This property can be set for the following objects:

- *Text* object

8.2.13 Horizontal mode

Description

The horizontal mode property changes the orientation of an object to be horizontal. By default it is vertical.



Available objects

This property can be set for the following objects:

- *Wheel* object

8.2.14 ID

Description

Every object has an ID that can be set in order to identify that object.



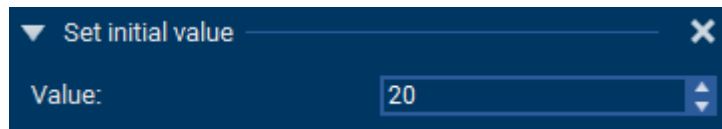
Available objects

This property can be set for all objects.

8.2.15 Initial value

Description

This property sets the initial value of an object.



Available objects

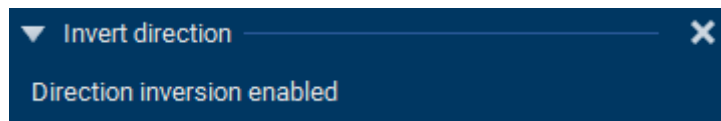
This property can be set for the following objects:

- *Gauge* object
- *Progbar* object
- *Rotary* object
- *Slider* object

8.2.16 Invert direction

Description

This property inverts the direction of an object, meaning its lowest value and initial position will be on the right instead of the left.



Available objects

This property can be set for the following objects:

- *Slider* object
- *Progbar* object

8.2.17 Motion partner

Description

Horizontal and vertical motion allow swiping between different screens or windows.

Available objects

This property can be set for the following objects:

- *Screen* object
- *Window* object

Horizontal motion properties

Property	Description
Left partner	Screen/window that should be located left from the screen/window.
Mode left	Mode that should be applied to the left partner. Either 'disclose' or 'replace'.
Right partner	Screen/window that should be located right from the screen/window.
Mode right	Mode that should be applied to the left partner. Either 'disclose' or 'replace'.
Period	Period to be used until motion stops.

Vertical motion properties

Property	Description
Upper partner	Screen/window that should be located above the screen/window.
Mode up	Mode that should be applied to the upper partner. Either 'disclose' or 'replace'.
Lower partner	Screen/window that should be located below the screen/window.
Mode down	Mode that should be applied to the lower partner. Either 'disclose' or 'replace'.
Period	Period to be used until motion stops.

Disclose mode

In 'disclose mode' the window that the user is swiping to will be disclosed. This means only the window that is swiped away moves, the other window does not.

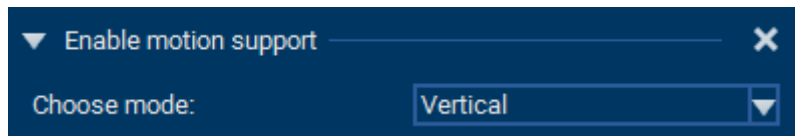
Replace mode

In 'replace mode' the window that the user is swiping to replaces the old window as the user is swiping.

8.2.18 Motion support

Description

Horizontal and vertical motion allow swiping the content of the widget.



Available objects

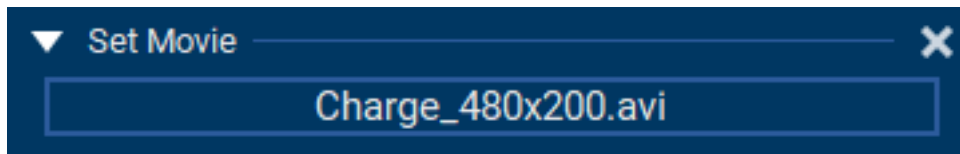
This property can be set for the following objects:

- *Multiedit* object

8.2.19 Movie

Description

The movie property allows to set a movie to be used in a movie object.



Available objects

This property can be set for the following objects:

- *Movie* object

8.2.20 Opaque mode

Description

This property removes the transparency flag of an object. For example, when a transparent button is pressed, not only the button is redrawn, but also the window that is behind the button.

When an opaque button is pressed, only the button itself is redrawn, not the window that is behind the button.



Note

This property should **only** be used, if the entire area of the object is drawn. For example, if an opaque button has rounded corners, the corners of the button will not be redrawn and corrupted pixels will appear on the screen.

Available objects

This property can be set for the following objects:

- *Button* object
- *Image* object
- *Window* object

8.2.21 Overwrite mode

Description

This property sets the mode of a possible cursor to overwrite. The default mode is insert.

Available objects

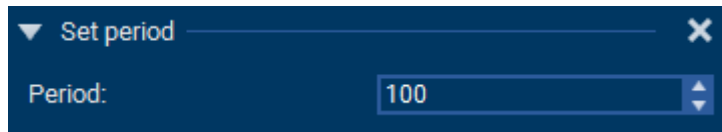
This property can be set for the following objects:

- *Edit* object
- *Multiedit* object

8.2.22 Period

Description

This property sets a time period in ms how long the related operation should take until it is finished.



Available objects

This property can be set for the following objects:

- *Keyboard* object
- *Multiedit* object
- *Progbar* object
- *Rotary* object
- *Switch* object
- *Timer* object
- *Wheel* object

8.2.23 Position and size

Description

Every object has its position and its size.

Position: Size:

Available objects

These properties can be set for every object.

8.2.24 Radius

Description

This property sets the radius of an object or a specific part of an object. For the Rotary object, it depends on this radius where the marker bitmap will be positioned.



Available objects

This property can be set for the following objects:

- *Box* object
- *Button* object
- *Gauge* object
- *Keyboard* object
- *Progbar* object
- *Rotary* object

Rounded corners

For the Box and Button objects, this property defines the radius of rounded corners.



8.2.25 Range

Description

This property allows to define a range for an object.



Available objects

This property can be set for the following objects:

- *Edit* object (in decimal mode)
- *Progbar* object
- *Rotary* object
- *Slider* object
- *Text* object (in decimal mode)

8.2.26 Space

Description

This property defines the spacing. For instance, spacing in X- and Y-axis can be set between each key on a Keyboard object.



Available objects

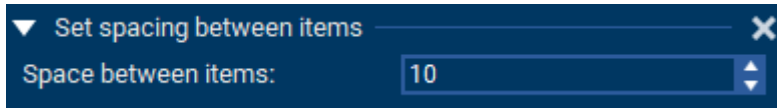
This property can be set for the following objects:

- *Edit* object
- *Keyboard* object

8.2.27 Space between items

Description

This property defines an additional spacing between the items. For instance, additional spacing in Y-axis can be set between each item on a Listbox object.



Available objects

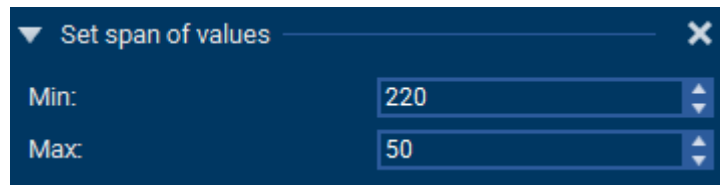
This property can be set for the following objects:

- *Listbox* object

8.2.28 Span of values

Description

This property defines the range of numbers an object should return.



▼ Set span of values ×

Min:	220
Max:	50

Available objects

This property can be set for the following objects:

- *Gauge* object
- *Rotary* object

8.2.29 Stay on top

Description

This property allows a screen to be shown on top of all other screens. In order for the screen to be visible, Persistent mode also has to be enabled.

If multiple screens are marked to stay on top, their order in the hierarchic tree will determine in what order they are shown.



Available objects

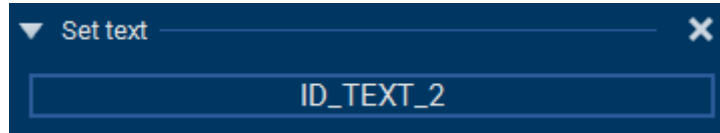
This property can be set for the following objects:

- *Screen* object

8.2.30 Text

Description

The text property allows to select a text to be shown from the text window. Only a text from the text window can be selected. For more information about the text window and how to add texts, refer to *Text resource window* on page 44.



Available objects

This property can be set for the following objects:

- *Button* object
- *Edit* object
- *Multiedit* object
- *QRCode* object
- *Switch* object
- *Text* object

8.2.31 Text rotation

Description

Rotates the text in an object.



Available objects

This property can be set for the following objects:

- *Text* object

Available rotation modes

Rotation	Description
CCW	Counter-clockwise rotation (210°).
CW	Clockwise rotation (90°).
180°	180° rotation.

8.2.32 Text wrapping

Description

Text wrapping for text in an object.



Available objects

This property can be set for the following objects:

- *Multiedit* object
- *Text* object

Comparison

No text wrapping	Text wrapping
A diagram showing a dashed pink rectangular frame. Inside the frame, the text "Lorem ipsum de" is displayed on a single line, extending to the right edge of the frame. There are small white squares at the corners and midpoints of the frame's edges.	A diagram showing a dashed pink rectangular frame. Inside the frame, the text "Lorem ipsum" is on the first line and "dolor sit amet" is on the second line, wrapping to fit within the frame's width. There are small white squares at the corners and midpoints of the frame's edges.

8.2.33 Tiling

Description

Tiling mode will fill the entirety of the Image object with the selected image.

It can also be used for Progbar objects to fill the entire Progbar with a narrow bitmap (usually 1 px wide).

Available objects

This property can be set for the following objects:

- *Image* object
- *Progbar* object

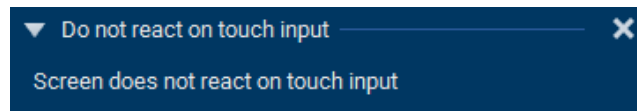
Example

See chapter *Image* on page 138 for an example.

8.2.34 Untouchable

Description

An untouchable screen is not able to receive touch input.



Available objects

This property can be set for the following objects:

- *Screen* object
- *Box* object
- *Image* object
- *Text* object
- *Movie* object

8.2.35 Vertical mode

Description

The vertical mode property changes the orientation of an object to be vertical. By default it is horizontal.

Changing a Progbar or Slider to horizontal or vertical mode will automatically change the default bitmaps accordingly.



Available objects

This property can be set for the following objects:

- *Progbar* object
- *Slider* object

8.3 Object focus

Since V1.12, objects are able to receive input focus. The ability for objects to receive focus must be enabled in the project options, as described earlier in the chapter *User interface* on page 32.

Enabling focus support

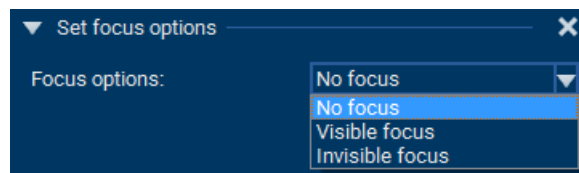
The user can define show the focus rectangle should be drawn by setting the line width, line color and corner radius in the project options.



How it works

If an object has the input focus, all key inputs are sent to the object. Using the tab key sets the focus to the next object, according to the hierarchic object tree.

Optionally, the ability of an object to receive focus or visibility of it can be modified for individual objects through the use of the Focus options property.



The objects that are able to receive focus are listed under the signal *GOT_FOCUS* on page 175.

Example

Below is an example Edit object that has the focus:



8.4 Object scrollers

Since V1.32, objects are able to show scroll bars by the use of emWin's SCROLLER widget. The scrollers can either be used only as a scrolling indicator or as an active scroll bar that can be moved by touch (see "Interactive" property below).

Note

For more detailed guide about SCROLLER properties, please have a look at the guide **UM03001 emWin User Guide & Reference Manual**.

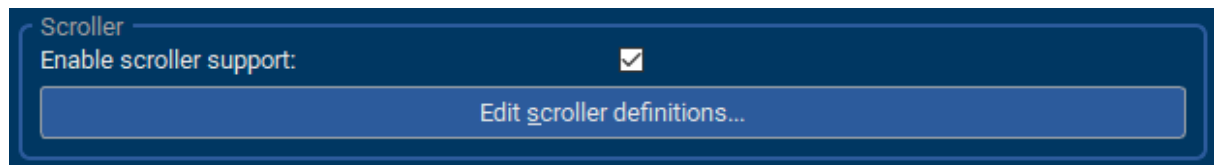
Supported objects

The following objects support the use of scrollers:

- Dropdown
- Listbox
- Listview
- Multiedit

Enabling scroller support

First, scrollers have to be enabled in the project properties. When this is done, all scrollable widgets (mentioned above) will have a scroller attached when they are being created.

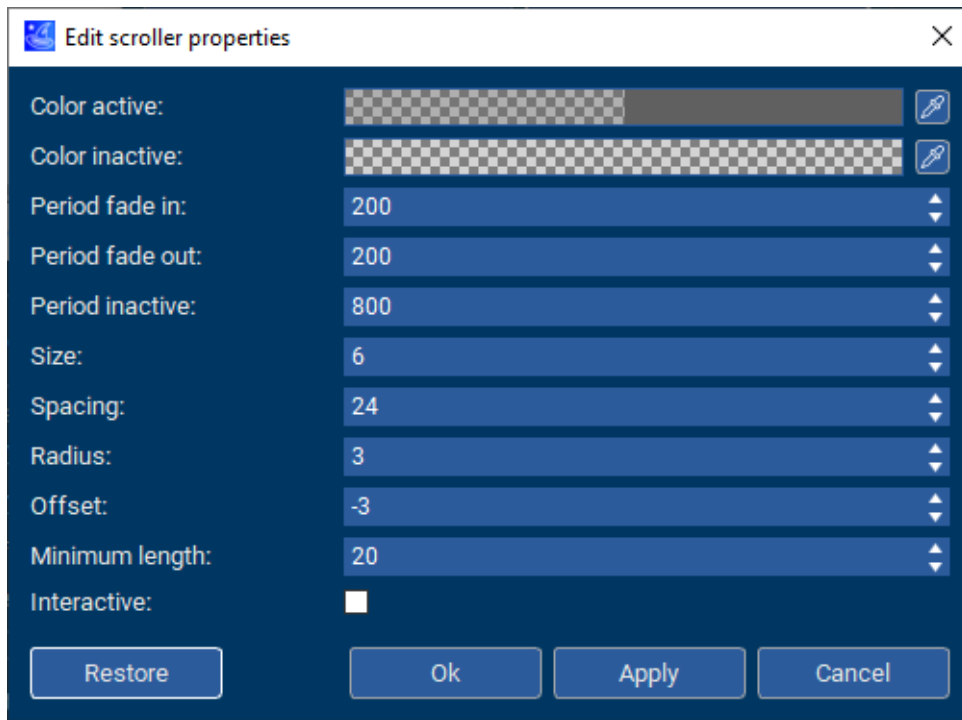


The scroller will be made visible when the size of the content allows scrolling and it will be hidden when the user cannot scroll.

Scroller appearance and properties

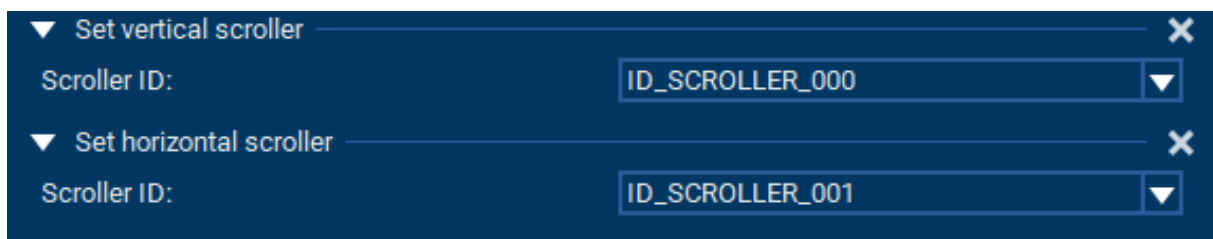
When added, scrollers will first have the default appearance defined in the default scroller properties in the AppWizard preferences.

The appearance of individual scrollers can be defined in the project properties. Clicking "Edit scroller definitions..." will open a window for defining multiple custom scrollers.

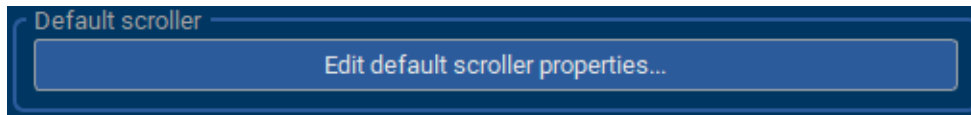


Property	Default value	Description
Color active	0x50606060	Color of the thumb rectangle when the scroller is moving (active state).
Color inactive	GUI_TRANSPARENT	Color of the thumb rectangle when the scroller is not moving (inactive state).
Period fade in	200	Period in ms for fading the color from the inactive state to the active state.
Period fade out	200	Period in ms for fading the color from the active state to the inactive state.
Period inactive	800	Period in ms it should take for the scroller to be faded into the inactive state after it has stopped moving.
Size	6	Breadth of the thumb rectangle.
Spacing	24	Spacing surrounding the thumb rectangle.
Radius	3	Radius of the thumb rectangle.
Offset	-3	Offset added to the thumb rectangle's position.
Minimum length	20	Minimum length of the thumb rectangle.
Interactive	No	Option if scroller should be able to be moved by touch or if it should be only a scrolling indicator.



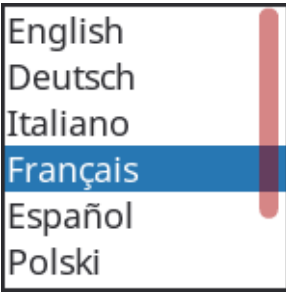
These custom defined scrollers can then be set to individual widgets in the widget's properties.



The default scroller properties can be changed in the AppWizard preferences (Edit → Preferences).

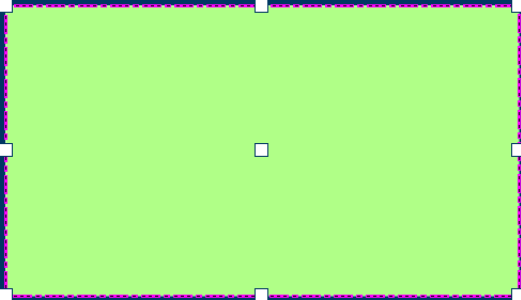


Examples

No scroller	Default scroller	Custom scroller
		

8.5 Box

A box object can be placed as the first object in a window/screen and simply serves for specifying a background color or a gradient. Horizontal and vertical gradients are supported. A gradient can have an unlimited number of colors. For each color the pixel position can be defined. Semi-transparent gradients are also supported.

Symbol	Example
	

Note

Semi-transparency is only recommended if a hardware is used which either has an accelerator for semi-transparent filling operations or is fast enough to mix up the colors per software.

Properties

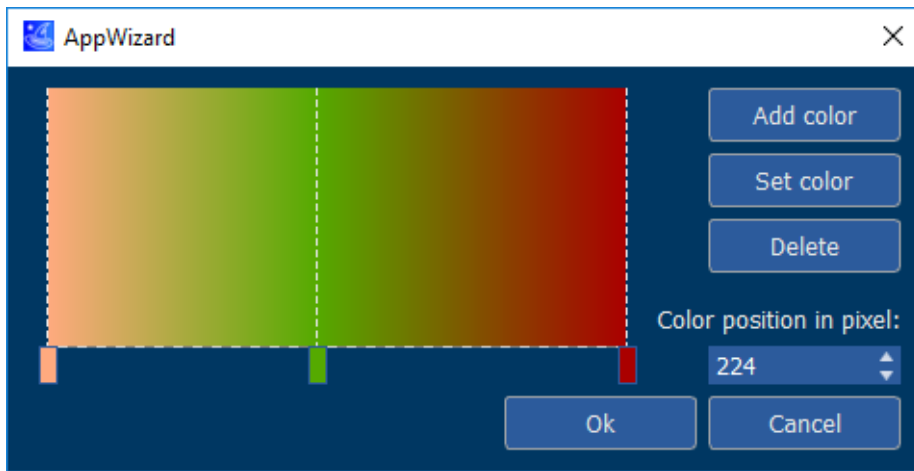
Property	Description
Color	Color to be shown in the box.
Horizontal gradient	Horizontal gradient to be shown in the box.
Vertical gradient	Vertical gradient to be shown in the box.
Radius	Radius of the rounded corners.
Untouchable	Sets the box object as untouchable and sends any touch input to the object below.

Gradients

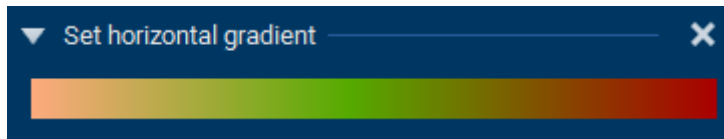
Horizontal and vertical gradients can be defined using two colors or more.

Gradient colors may be added via the **Add color** button. A gradient must contain at least two colors. The colors can be changed when the corresponding marker has been clicked. They can be edited using the **Set color** button and deleted via the **Delete button**.

The position of each color can be changed by specifying the position in the spinbox or by moving the markers.



The result of the above specified gradient looks like this below:



8.6 Button

The button object is very similar to its emWin counterpart. It is an object that can be clicked, so that its input may be processed by the application.

Symbol	Example
	

Properties


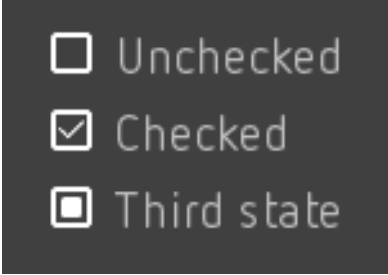
Property	Description
<code>Text colors</code>	<ul style="list-style-type: none"> • Unpressed: Text color for unpressed state. • Pressed: Text color for pressed state. • Disabled: Text color for disabled state.
<code>Background colors</code>	<ul style="list-style-type: none"> • Unpressed: Background color for unpressed state. • Pressed: Background color for pressed state. • Disabled: Background color for disabled state.
<code>Bitmaps</code>	<ul style="list-style-type: none"> • Unpressed: Bitmap for unpressed state. • Pressed: Bitmap for pressed state. • Disabled: Bitmap for disabled state.
<code>Bitmap alignment</code>	<ul style="list-style-type: none"> • Alignment: Bitmap alignment. • Offset x: Additional x-offset. • Offset y: Additional y-offset.
<code>Auto repeat</code>	<ul style="list-style-type: none"> • Start time: Starting time of auto repeating after button press. • Interval: Repeating time.
<code>Toggle mode</code>	By clicking the button its state is toggled between pressed and unpressed.
<code>Text</code>	Text to be shown.
<code>Text alignment</code>	<ul style="list-style-type: none"> • Alignment: Text alignment. • Offset x: Additional x-offset. • Offset y: Additional y-offset.
<code>Font</code>	Font to be used for the text.
<code>Focus options</code>	Disables the focus for the button or hides it.
<code>Radius</code>	Radius for rounded corners if the button is drawn without bitmaps.
<code>Opaque mode</code>	Sets the button to opaque and removes its transparency flag.

Auto repeat mode

The Button also offers an auto repeat mode. When holding the button pressed, it begins sending clicked events after the start time period in the given interval.

8.7 Checkbox

The checkbox object is very similar to its emWin counterpart. A check box may be checked or unchecked by the user, and any number of boxes may be checked at one time.

Symbol	Example
	

Properties


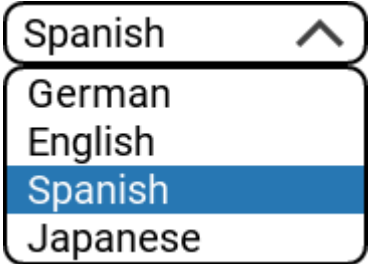
Property	Description
Bitmaps	<ul style="list-style-type: none"> • Unchecked: Bitmap for unchecked state. • Checked: Bitmap for checked state. • ThirdState: Bitmap for (optional) third state. • Unchecked (disabled): Bitmap for unchecked state in disabled mode. • Checked (disabled): Bitmap for checked state in disabled mode. • ThirdState (disabled): Bitmap for (optional) third state in disabled mode.
Colors	<ul style="list-style-type: none"> • Text: Text color for unpressed state. • Bitmaps: Bitmap color for all images. (Works only with alpha images) • Disabled: Text color for disabled state.
Font	Font to be used for the text.
Text	Text to be shown.
Text offset	Additional horizontal spacing between image and text.
3 state mode	Activates the third state.

3 state mode

The Checkbox also offers an optional third state. It can be activated on demand.

8.8 Dropdown

The Dropdown object is a control that allows a user to select a value from a given list of values. In inactive state it shows the currently selected item. When activating it via PID or keyboard, a 'drop-down' (or even 'drop-up') list pops with the selectable items. The list automatically disappears after selecting an item or when clicking elsewhere on the display.

Symbol	Example
	

Properties

Property	Description
<code>Bitmaps</code>	<ul style="list-style-type: none"> <code>Down</code>: Down arrow to be shown. <code>Up</code>: Down arrow to be shown.
<code>Content</code>	Content to be shown.
<code>Text colors</code>	<ul style="list-style-type: none"> <code>Enabled</code>: Text color for enabled state. <code>Disabled</code>: Text color for disabled state. <code>Cursor</code>: Color of cursor.
<code>Colors</code>	<ul style="list-style-type: none"> <code>Enabled</code>: Background color for enabled state. <code>Disabled</code>: Background color for disabled state. <code>Cursor</code>: Background color of cursor.
<code>Font</code>	Font to be used.
<code>Text alignment</code>	<ul style="list-style-type: none"> <code>Alignment</code>: Text alignment. <code>Offset x</code>: Additional x-offset. <code>Offset y</code>: Additional y-offset.
<code>Opening upwards</code>	Changes the dropDOWN to a dropUP.
<code>Height of list</code>	Height in pixels of the list.
<code>Frame color</code>	Color of frame of object.
<code>Frame radius</code>	Radius of rounded corners of the frame.
<code>Frame size</code>	Width of frame around object.
<code>Inner gap</code>	Border size between frame and text.
<code>Focus options</code>	Disables the focus for the object or hides it.
<code>Vertical scroller</code>	Custom scroller definition to be used on the vertical axis. If this is left empty and scrollers are enabled for the project, the default scroller properties will be used instead.

8.9 Edit

An Edit object provides, like the emWin EDIT widget, a box where the user can type text in, or numbers if decimal mode is activated.

Symbol	Example
	

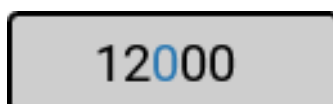
Properties

Property	Description
Text	Text to be displayed initially.
Decimal mode	Mask to be used for decimal mode.
Overwrite mode	Set the cursor mode to overwrite.
Text colors	<ul style="list-style-type: none"> Enabled: Text color for enabled state. Disabled: Text color for disabled state. Cursor: Color of cursor.
Background colors	<ul style="list-style-type: none"> Enabled: Background color for enabled state. Disabled: Background color for disabled state. Cursor: Background color of cursor.
Frame color	Color of frame of Edit.
Cursor inversion	Disables cursor inversion mode.
Blink period	Blinking period of cursor.
Font	Font to be used.
Text alignment	<ul style="list-style-type: none"> Alignment: Text alignment. Offset x: Additional x-offset. Offset y: Additional y-offset.
Inner gap	Spacing between frame and text.
Frame radius	Radius of rounded corners of the Edit's frame.
Frame size	Line size of frame around Edit.
Password mode	Enables password mode. See below for additional information.
Maximum length	Maximum length of characters that can be entered into the Edit.
Focus options	Disables the focus for the Edit or hides it.

Decimal mode

With decimal mode, the Edit object is only eligible of holding digits instead of characters. For this mode, a mask of zeros has to be specified which determines how many digits are shown by the object. More details about the usage of the mask is explained under *Decimal mode* on page 96.

Also, when using decimal mode, a range property is added to the object to limit the numbers that can be entered. More on the range property can be found under *Range* on page 114.

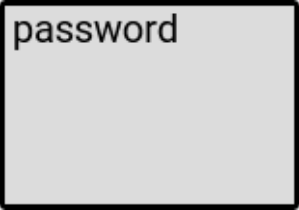
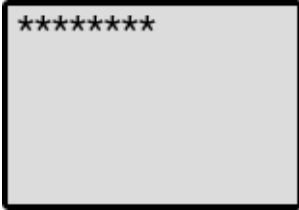


During runtime, the cursor is highlighting the currently selected digit. When the user types in a number, the cursor will move from its current position to the right until the last digit has been reached. If the entered number exceeds the maximum, the maximum number is put in.

The number can be increased using the <UP> and decreased using the <DOWN> key, whereas the cursor can be moved using the <LEFT> and <RIGHT> arrow keys.

Password mode



The password mode displays the text of the Edit as asterisks.

Password mode disabled	Password mode enabled
	

8.10 Gauge

A Gauge object is similar to a progress bar, although the progress is displayed in a radial manner. The object consists of two arcs that are drawn. The relation between these two arc lines shows the progress.

Two colors can be set for a Gauge object, for the background and and foreground line.




Symbol	Example
	

Properties

Property	Description
<code>Center alignment</code>	<ul style="list-style-type: none"> Alignment: Alignment of the Gauge within the object frame. Offset x: Additional x-offset of the Gauge. Offset y: Additional y-offset of the Gauge.
<code>Initial value</code>	Initial value of the Gauge.
<code>Start/end angle</code>	<ul style="list-style-type: none"> Angle 0: Start angle in 10th of degrees. Angle 1: End angle in 10th of degrees.
<code>Span of values</code>	<ul style="list-style-type: none"> Min: Lowest value the object should return. Max: Highest value the object should return.
<code>Radius</code>	Radius of the Gauge.
<code>Colors</code>	<ul style="list-style-type: none"> Item 0: Color of the background curve. Item 1: Color of the foreground curve.
<code>Line width</code>	<ul style="list-style-type: none"> Width 0: Width of the background curve. Width 1: Width of the foreground curve.
<code>Rounded value</code>	Enables rounded ends of the foreground (value) curve.
<code>Rounded ends</code>	Enables rounded ends of the background curve.
<code>Background color</code>	Background color of the object.
<code>CCW mode</code>	Enable counter-clockwise mode which inverts the direction the gauge will fill.

Rounded value/ends

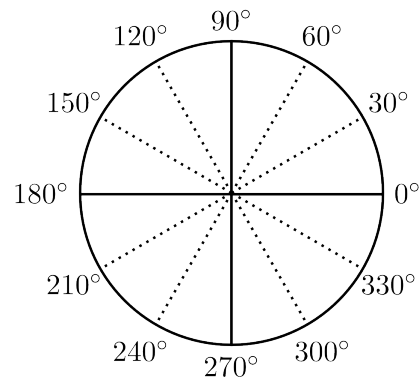
With this property, the ends of the value or end arc of a Gauge object can be set to have rounded edges.

Default	Rounded value	Rounded ends
		

Start/end angle



This property defines the angles, where the arc of a Gauge object should start and end. The values to be entered should be 10th of degrees ($1800 = 180^\circ$).

The entered degree values are based off the standard angle measurement.



8.11 Image

An Image object is similar to emWin's IMAGE widget. It can be used to display any images of the file types JPEG, GIF or BMP. Alternatively, a bitmap can be chosen as well.

Symbol	Example
	

Properties

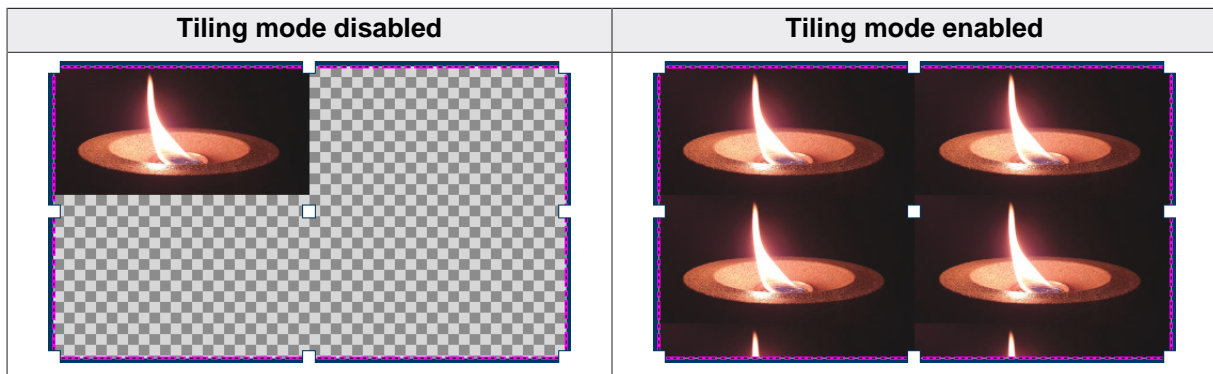
Property	Description
Bitmap	Sets a bitmap to the object.
Color	Sets the color to be used for an alpha bitmap.
JPEG	Sets a JPEG image to the object.
GIF	Sets a GIF image to the object.
BMP	Sets a BMP image to the object.
Tiling	Enables tiling mode for the object.
Opaque mode	Sets the image to opaque and removes its transparency flag.
Bitmap alignment	Alignment of the bitmap within the object.
Angle	Angle in 1000th of degrees to rotate the image counter-clockwise (1000 = 1°).
Scale	Image scale factor in promille (1000 = 100%).
Alpha	Alpha value (0-255) to be used to draw the image with semi-transparency.
Fast mode	Speeds up rotation, scaling and alpha blending operations but the result will have a lower quality.
Untouchable	Sets the image object as untouchable and sends any touch input to the object below.

Difference between bitmaps and images

In contrast to bitmaps, JPEG, GIF and BMP images are always displayed natively. Therefore JPEGs and GIFs are always decompressed before being displayed. This can lead to a notable difference in performance compared to bitmaps.

Tiling mode

Tiling mode will fill the entirety of the Image object with the selected image. In this example the purple frame surrounds the Image object.



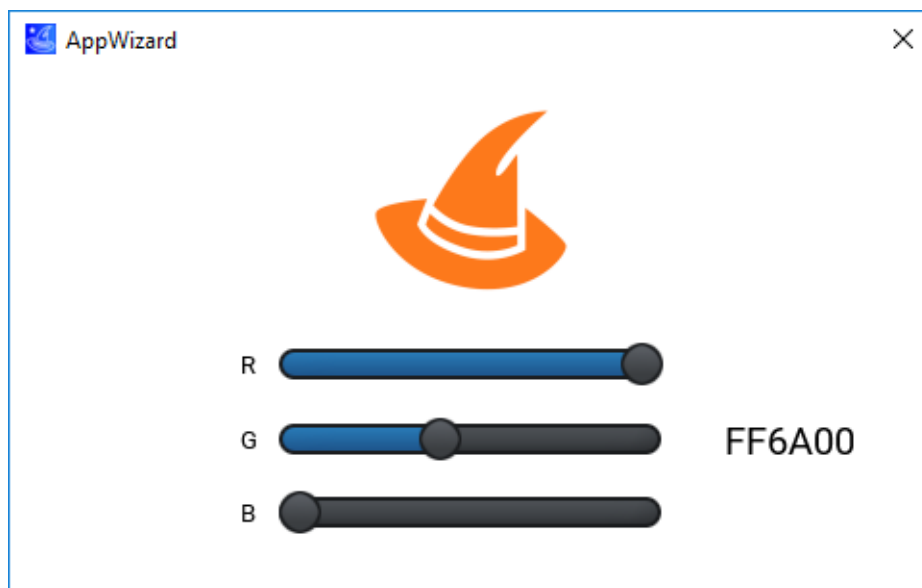
GIF support

Any GIF images are supported for this object, this includes animated GIFs.

Alpha bitmaps

To create an alpha bitmap to use it for an Image object, click the "Set bitmap" property of the Image. Then, add a new bitmap by opening the desired image. Now, select *Alpha channel, compr.* in the "Format" column to declare it as an alpha bitmap. Finally, select the bitmap for the Image.


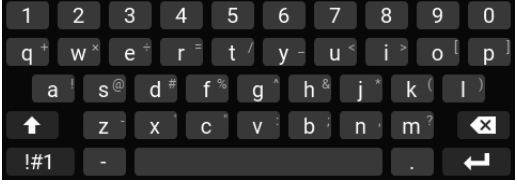
For the alpha bitmap, a desired color can now be selected. The color can also be set to the Image by an interaction, using the SETCOLOR job.



In the example shown above R, G and B values can be entered via the slider. These three values are put together by a variable calculation to form the RGB value. Finally the RGB value can be set to the alpha bitmap via an interaction.

8.12 Keyboard

A Keyboard object can be used to enter text or numbers.

Symbol	Example
	

Properties

Property	Description
Font for codes	Font used for keys.
Font for longpress codes	Font used for longpress characters on keys.
Keyboard layout	Keyboard layout used for the object.
Colors	<ul style="list-style-type: none"> Code: Color used for text shown on keys. Long: Color used for longpress characters shown on keys. Mark: Color used for selected character in long press dialog and locked shift key.
Background colors	<ul style="list-style-type: none"> Key: Background color for keys. F-Key: Background color for function keys. Pressed: Background color for pressed keys. BG: Background color of Keyboard object.
Periods for backspace key	<ul style="list-style-type: none"> Start time: Period between the press of backspace and deletion of the characters. Interval: Interval between each character deleted when holding backspace.
Radius for key outline	Radius used for rounded corners of the keys.
Space between keys	<ul style="list-style-type: none"> Space (X-axis): Space between each key on X-axis. Space (Y-axis): Space between each key on Y-axis.

Keyboard layout



The following layouts are available per default and can be set to a Keyboard object:

Layout	Description
SKEYBOARD_ARA	Layout for Arabic.
SKEYBOARD_DEU	QWERTZ layout, used for German.
SKEYBOARD_DEU_LP	QWERTZ layout with extra longpress characters.
SKEYBOARD_ENG	QWERTY layout, used for English.
SKEYBOARD_ENG_LP	QWERTY layout with extra longpress characters.
SKEYBOARD_FRA_LP	AZERTY layout, used for French.
SKEYBOARD_NUMPAD	Numpad layout.
SKEYBOARD_RUS	JCUKEN/ЙЦУКЕИ layout, main Cyrillic keyboard layout for the Russian language.

Files for streamed layout files are located in the project directory under `Resource\Keyboard`. The pattern files needed for specific layouts are located in the `ProgramData` directory of the AppWizard installation.

8.13 Listbox

A list box allows selecting an item from a multiple line text box.


Symbol	Example
	

Properties

Property	Description
<code>Text colors</code>	<ul style="list-style-type: none"> • Unselected: Text color for unselected state. • Selected: Text color for selected state. • Focused: Color of cursor. • Disabled: Text color for disabled state.
<code>Background colors</code>	<ul style="list-style-type: none"> • Unselected: Background color for unselected state. • Selected: Background color for selected state. • Focused: Background color of cursor. • Disabled: Background color for disabled state.
<code>Font</code>	Font to be used.
<code>Content</code>	Content to be shown.
<code>Text alignment</code>	<ul style="list-style-type: none"> • Alignment: Text alignment. • Offset x: Additional x-offset. • Offset y: Additional y-offset.
<code>Frame color</code>	Color of frame of object.
<code>Frame size</code>	Width of frame around object.
<code>Focus options</code>	Disables the focus for the object or hides it.
<code>Space between items</code>	Defines additional space between the items.
<code>Vertical scroller</code>	Custom scroller definition to be used on the vertical axis. If this is left empty and scrollers are enabled for the project, the default scroller properties will be used instead.

8.14 Listview

A Listview object allows selecting one line of of a list with several columns. Each column has its own width and text alignment.

Symbol	Example																								
	<table border="1"> <thead> <tr> <th>Col 0</th> <th>Col 1</th> <th>Col 2</th> <th>Col 3</th> </tr> </thead> <tbody> <tr> <td>Item 0/1</td> <td>Resource</td> <td>Item 2/1</td> <td>Item 3/1</td> </tr> <tr> <td>Item 0/9</td> <td>Item 1/9</td> <td>Item 2/9</td> <td>Item 3/9</td> </tr> <tr> <td>Item 0/8</td> <td>Item 1/8</td> <td>Item 2/8</td> <td>Item 3/8</td> </tr> <tr> <td>Item 0/7</td> <td>Item 1/7</td> <td>Item 2/7</td> <td>Item 3/7</td> </tr> <tr> <td>Item 0/6</td> <td>Item 1/6</td> <td>Item 2/6</td> <td>Item 3/6</td> </tr> </tbody> </table>	Col 0	Col 1	Col 2	Col 3	Item 0/1	Resource	Item 2/1	Item 3/1	Item 0/9	Item 1/9	Item 2/9	Item 3/9	Item 0/8	Item 1/8	Item 2/8	Item 3/8	Item 0/7	Item 1/7	Item 2/7	Item 3/7	Item 0/6	Item 1/6	Item 2/6	Item 3/6
Col 0	Col 1	Col 2	Col 3																						
Item 0/1	Resource	Item 2/1	Item 3/1																						
Item 0/9	Item 1/9	Item 2/9	Item 3/9																						
Item 0/8	Item 1/8	Item 2/8	Item 3/8																						
Item 0/7	Item 1/7	Item 2/7	Item 3/7																						
Item 0/6	Item 1/6	Item 2/6	Item 3/6																						

Properties

Property	Description
Text colors	<ul style="list-style-type: none"> Unselected: Text color for unselected state. Selected: Text color for selected state. Focused: Color of cursor. Disabled: Text color for disabled state.
Background colors	<ul style="list-style-type: none"> Unselected: Background color for unselected state. Selected: Background color for selected state. Focused: Background color of cursor. Disabled: Background color for disabled state.
Header colors	<ul style="list-style-type: none"> Text: Color for header text. Background: Color for header background. Frame: Color for header frame.
Colors	<ul style="list-style-type: none"> Grid: Color for grid lines. Focus: Color for the optional focus rectangle. Frame: Frame color of list.
Bitmaps	<ul style="list-style-type: none"> Down: Optional descending indicator to be shown. Up: Optional ascending indicator to be shown.
Font	Font to be used for content.
Header font	Font to be used for header. This is optional, if left empty, the same font set to the LISTVIEW is used.
Content	Content to be shown.
Row height	Row height be used for content.
Header height	Row height be used for header.
Header radius	Radius to be used for the header frame.
Frame size	Width of frame around the list.
Focus rectangle width	Width of the optional focus rectangle.
Show header grid	Show vertical grid lines in header.
Show horizontal grid lines	Show horizontal grid lines.
Show vertical grid lines	Show vertical grid lines.
Enable cell selection	Enables single cell selection instead of line selection.
Fixed columns	Sets the number of fixed columns.
Column sorting	Enables column sorting.
Inner gap	Horizontal border between text and cell.
Focus options	Disables the focus for the object or hides it.

Property	Description
<code>Vertical scroller</code>	Custom scroller definition to be used on the vertical axis. If this is left empty and scrollers are enabled for the project, the default scroller properties will be used instead.
<code>Horizontal scroller</code>	Custom scroller definition to be used on the horizontal axis. If this is left empty and scrollers are enabled for the project, the default scroller properties will be used instead.

Header radius

The upper corners of the header can be rounded with a separate radius. Further there are options for setting a separate color for frame of the header line:

Col 0	Col 1	Col 2	Col 3
Item 0/0	Item 1/0	Item 2/0	Item 3/0

Focus rectangle width

The focus rectangle width (and color) allows showing a separate focus rectangle surrounding the currently selected row or cell:

Col 0	Col 1	Col 2	Col 3
Item 0/0	Item 1/0	Item 2/0	Item 3/0
Item 0/1	Resource	Item 2/1	Item 3/1
Item 0/2	Item 1/2	Item 2/2	Item 3/2

Enable cell selection

Cell selection allows selecting single cells:

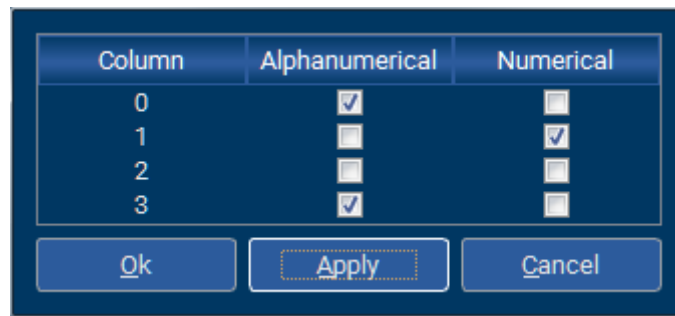
Col 0	Col 1	Col 2	Col 3
Item 0/0	Item 1/0	Item 2/0	Item 3/0
Item 0/1	Resource	Item 2/1	Item 3/1
Item 0/2	Item 1/2	Item 2/2	Item 3/2
Item 0/3	Item 1/3	Item 2/3	Item 3/3
Item 0/4	Item 1/4	Item 2/4	Item 3/4

Fixed columns

Fixes the given number of columns at their horizontal positions.

Column sorting

When selecting this option a dialog occurs. It allows setting sorting options for each column:



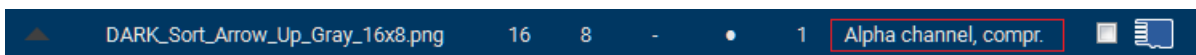
Sorting then can be initiated by touching the header of the desired column. After touching the header a sort indicator becomes visible:



Touching the header again inverts the sorting order:



The AppWizard provides a convenient option for showing the indicators with the same color as used for the text. This can be achieved with monochrome images only. To use that feature select 'Alpha channel, compr.' instead of the 'Auto'-option in the image dialog:



Inner gap

It defines the number of pixels between text and border of the cell. Makes sense with left or right alignment only.

8.15 Movie

The Movie object offers capability to display movie files while also providing basic controls like pause, play and a progress bar.

Symbol	Example
	

Properties

Property	Description
Bitmaps	<ul style="list-style-type: none"> • Center: Bitmap to be shown in the center when movie is stopped. • Play: Panel bitmap for 'Start' button. • Pause: Panel bitmap for 'Stop' button. • Begin: Panel bitmap for 'Begin' button.
Colors	<ul style="list-style-type: none"> • Center: Color of center bitmap (works only with alpha image). • Panel: Color of panel bitmaps (works only with alpha image). • Background: Used to fill the space between movie and border if the movie does not fill the complete area of the object. • Progress bar left: Color of progress bar left. • Progress bar right: Color of progress bar right.
Period for shifting in	Period for shifting in the panel.
Period for shifting out	Period for shifting out the panel.
Inactivity period	Inactivity period, panel will be shifted out after that period.
Bar height	Height of the progress bar.
Panel height	Height of the panel including the progress bar. Should be greater than bar height.
Endless loop	Plays the movie in endless loop.
Do not react on touch input	Movie does not react on touch input.

Play and pause

The widget can be paused and played by tapping anywhere inside the widget area.

If the control panel is not shown, the first tap will open the panel and a second tap will pause the video.

Center play button

While the video is paused, it shows a play symbol if a center bitmap has been set.

Control panel

When the widget is clicked, the control panel will move in with an animation. The periods for moving in and out can be set with the period properties. If the panel is already visible, it will start shifting out after the inactivity period is expired.

Progress bar

The lower part of the control panel contains a progress bar which displays the current frame position of the movie.



Colors can be set for the left and for the right part of the progress bar with the color properties.

By touching the progress bar or within the control panel, the user can move the current frame to a desired position, just like in commonly known video players or apps.


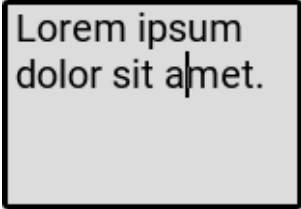
Control buttons

Control buttons can be added to the control panel. A button is added by setting the corresponding bitmap(s). We recommend to use alpha bitmaps. In this case the colors for the alpha bitmaps can be set with the color properties.

Button name	Example	Description
Pause/play button		Pauses or plays the video.
Begin button		Jumps to the beginning of the video.

8.16 Multiedit

The Multiedit object is a multi-line text input widget.

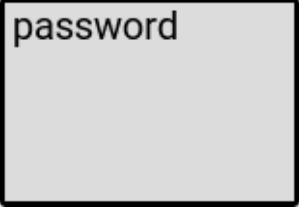

Symbol	Example
	

Properties

Property	Description
<code>Text</code>	Text to be displayed initially.
<code>Overwrite mode</code>	Set the cursor mode to overwrite.
<code>Text colors</code>	<ul style="list-style-type: none"> <code>Enabled</code>: Text color for enabled state. <code>Disabled</code>: Text color for disabled state. <code>Cursor</code>: Color of cursor.
<code>Background colors</code>	<ul style="list-style-type: none"> <code>Enabled</code>: Background color for enabled state. <code>Disabled</code>: Background color for disabled state. <code>Cursor</code>: Background color of cursor.
<code>Frame color</code>	Color of frame of Multiedit.
<code>Cursor inversion</code>	Disables cursor inversion mode.
<code>Blink period</code>	Blinking period of cursor.
<code>Font</code>	Font to be used.
<code>Text alignment</code>	Text alignment, only left and right alignment can be set to the Multiedit.
<code>Text wrapping</code>	Enables text wrapping.
<code>Inner gap</code>	Border size between frame and text.
<code>Frame radius</code>	Radius of rounded corners of the Multiedit's frame.
<code>Frame size</code>	Line size of frame around Multiedit.
<code>Password mode</code>	Enables password mode. See below for additional information.
<code>Read only mode</code>	Enables read only mode.
<code>Focus options</code>	Disables the focus for the Multiedit or hides it.
<code>Motion support</code>	Enables horizontal and/or vertical motion scrolling.
<code>Vertical scroller</code>	Custom scroller definition to be used on the vertical axis. If this is left empty and scrollers are enabled for the project, the default scroller properties will be used instead.
<code>Horizontal scroller</code>	Custom scroller definition to be used on the horizontal axis. If this is left empty and scrollers are enabled for the project, the default scroller properties will be used instead.



Password mode

The password mode displays the text of the Multiedit as asterisks.

Password mode disabled	Password mode enabled
	

8.17 Progbar

A Progbar object visualizes the progression of an operation.

Symbol	Example
	

Properties



Property	Description
<code>Bitmap</code>	Bitmap for the "filling" and the "empty" part of the Progbar.
<code>Tiling</code>	Uses tiling for the given bitmaps. The bitmaps should be 1 pixel wide.
<code>Bitmap alignment</code>	Alignment of the set bitmaps.
<code>Colors</code>	Bitmap for the "filling" and the "empty" part of the Progbar.
<code>Frame size</code>	Size of the frame of the Progbar. If 0 no frame is displayed.
<code>Initial value</code>	Sets the initial value for the Progbar.
<code>Invert direction</code>	Inverts the direction of the Progbar (left to right or right to left).
<code>Period</code>	Sets a period which describes the duration it takes to move a Bitmap through the object.
<code>Radius</code>	Sets the radius of the edges of the Progbar. Affects also the radius of the frame. Has no effect on gradients.
<code>Range</code>	Sets the range of the Progbar.
<code>Vertical mode</code>	Changes the direction of the Progbar from horizontal to vertical.

Additional information

The period is used to animate the "filling" bitmap of the progress bar. This way it is possible to indicate a state where the progress bar is waiting for data. The user has to make sure that the bitmap has the size of the Progbar object and the left and right endings of the bitmap match each other.

8.18 QRCode

A QRCode object displays a QR code. A custom text can be set which will then be converted into a QR code.

Symbol	Example
	

Properties

Property	Description
Error correction level	Error correction level for QR code.
Pixelsize	Size in pixels of one module for QR code.
Text	Text used to be encoded in QR code.
Version	Dimensions of the code.

Error correction level

The error correction level is a specific parameter of a QR code. The higher the error correction level, the more information is saved redundantly in the QR code in order to increase the chance to be read without errors.

emWin QR codes offer four error correction levels.

Error correction level	Description
GUI_QR_ECLEVEL_L	About 7% or less errors can be corrected.
GUI_QR_ECLEVEL_M	About 15% or less errors can be corrected.
GUI_QR_ECLEVEL_Q	About 25% or less errors can be corrected.
GUI_QR_ECLEVEL_H	About 30% or less errors can be corrected.

Pixelsize

The pixelsize property defines the size in pixels of one module in a QR code.

Version

The version of a QR code indicates the overall dimensions of the code. The entered value has to be between 0 and 40. If 0 is entered, the appropriate version is chosen automatically.

8.19 Radio

A radio button is a graphical element that allows users to choose only one option from a set of choices.

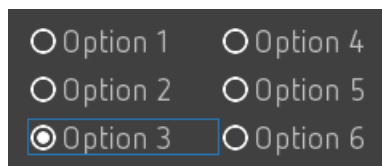
Symbol	Example
	

Properties

Property	Description
Bitmaps	<ul style="list-style-type: none"> <u>Unchecked</u>: Bitmap for unchecked state. <u>Checked</u>: Bitmap for checked state. <u>Unchecked (disabled)</u>: Bitmap for unchecked state in disabled mode. <u>Checked (disabled)</u>: Bitmap for checked state in disabled mode.
Colors	<ul style="list-style-type: none"> <u>Text</u>: Text color for selected and unselected state. <u>Bitmaps</u>: Bitmap color for all images. (Works only with alpha images) <u>Disabled</u>: Text color for disabled state.
Font	Font to be used for the text.
Content	Content to be shown.
Spacing	Vertical distance between the radio items.
Group Id	Id for combining multiple radio objects to one group.
Text offset	Additional horizontal spacing between image and text.

Group Id



If more than one column is required multiple objects have to be used:



Simply assign the same group ID to the desired radio objects. Radio objects with the same group ID then behave like one single radio object.

8.20 Rotary

A rotary object is similar to its emWin counterpart. A Rotary object is a circular object that can be rotated. The object consists of a background and a marker, both which make use of a bitmap. When rotating the object, the marker moves along the rotary axis. Depending on how the user set the scale, values are returned for the rotated degree.



Symbol	Example
	

Properties

Property	Description
Bitmaps	<ul style="list-style-type: none"> Background: Bitmap used for background of Rotary. Marker: Bitmap used for marker of Rotary.
Initial value	Initial value of Rotary.
Range	<ul style="list-style-type: none"> Positive: Rotation angle CCW in 10th of degrees. Negative: Rotation angle CW in 10th of degrees.
Span of values	<ul style="list-style-type: none"> Min: Value the object return at CCW limit. Max: Value the object return at CW limit.
Offset	This offset angle will make the Rotary appear rotated by that angle from the beginning. The offset is measured in 10th of degrees ($3600 = 360^\circ$).
Radius	Radius of the Rotary.
Rotate marker	Enables marker rotation.
Marker alignment	Sets an alignment and additional offset to the marker bitmap.
Period	Period how long the marker moves when released.
Snap position	Sets snap positions on the Rotary (in 10th of degrees).
Focus options	Disables the focus for the Rotary or hides it.

Rotate marker

When activated the marker bitmap is rotated when the Rotary is moved.


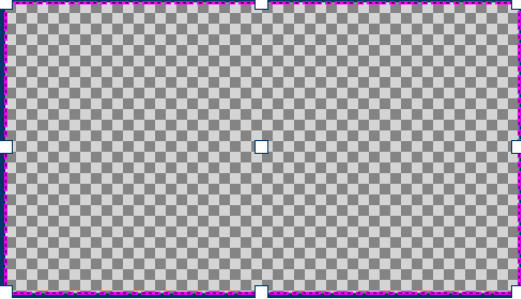
Without marker rotation	With marker rotation
	

Snap position

This property sets a position on the Rotary object at which it should snap in place. The snap position is specified in 10ths of degrees ($1800 = 180^\circ$).

8.21 Screen

A screen is an invisible parent object for all other objects. An application consists of one or more screens. Interactions are also assigned to one screen.

Symbol	Example
	

Properties

Property	Description
Horizontal motion	Horizontal motion screen/window partner.
Vertical motion	Vertical motion screen/window partner.
Persistent mode	Enables persistent mode for screen.

Horizontal / vertical motion

In case of enabling motion without specifying a partner, the window is simply movable (horizontal and/or vertical) within the range of the parent window.

Right before a "motion partner" of the screen becomes visible it will be created. As soon as the partner becomes invisible it will be deleted if not marked as persistent.

Persistent mode



The persistent mode property allows a screen to be persistent, so it does not get deleted during runtime when it is not visible anymore. It makes sense to use this mode, when the widgets in a screen are showing values which should not get deleted.

Note

All screens that are in persistent mode are created on start-up of the application.

8.22 Slider

A Slider object is, like the emWin SLIDER widget, a movable thumb on a shaft. By moving the thumb on the shaft, values can be selected.

Symbol	Example
	

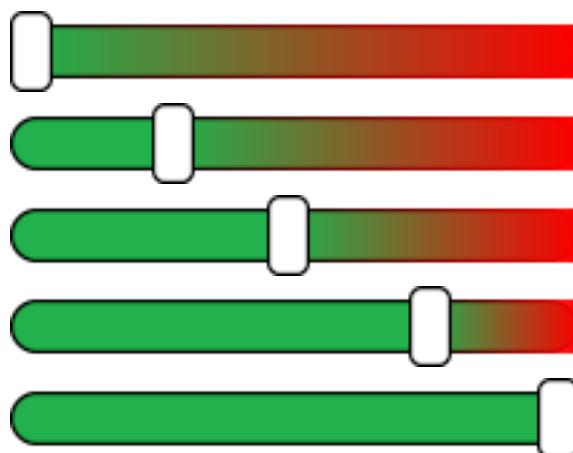
Properties

Property	Description
<code>Vertical mode</code>	Enables vertical mode.
<code>Invert direction</code>	Inverts direction of slider.
<code>Bitmaps</code>	<ul style="list-style-type: none"> <code>Shaft left</code>: Bitmap of left part of the shaft. <code>Shaft right</code>: Bitmap of right part of the shaft. <code>Thumb up</code>: Bitmap of unpressed thumb. <code>Thumb down</code>: Bitmap of pressed thumb.
<code>Blend colors</code>	<ul style="list-style-type: none"> <code>Shaft left</code>: Color on left part of the shaft to be blended in. <code>Shaft right</code>: Color on right part of the shaft to be blended in.
<code>Range</code>	<ul style="list-style-type: none"> <code>Min</code>: Minimum value of slider. <code>Max</code>: Maximum value of slider.
<code>Focus options</code>	Disables the focus for the Slider or hides it.

Blend colors

The blend colors setting makes it possible to choose a color for the left and/or right side of the shaft to be blended into the corresponding bitmap.

The below example uses the green shaft bitmaps provided as stock images by the AppWizard and has the color red set as the blend color for the right side of the shaft.



8.23 Switch

A switch object works like a switch present on most modern smartphones. It has two states and can be toggled by clicking on it.

Symbol	Example
	

Properties

Property	Description
Bitmaps	<ul style="list-style-type: none"> BG-Left: Background bitmap for left state. BG-Right: Background bitmap for right state. BG-Disabled: Background bitmap for disabled state. Thumb-Left: Thumb bitmap for left state. Thumb-Right: Thumb bitmap for right state. Thumb-Disabled: Thumb bitmap for disabled state.
Left text	Text displayed in left state.
Right text	Text displayed in right state.
Text colors	<ul style="list-style-type: none"> Text left: Text color in left state. Text right: Text color in right state.
Disable animation	Disables the animation when toggling between the states.
Font	Font to be used for the object.
Period	Animation period when clicking the switch.
Fade mode	Enables fade mode which fades the background bitmaps when switching from right to left state.
Focus options	Disables the focus for the Switch or hides it.

Fade mode and disclose mode

By default, a Switch object uses the **disclose mode**, which means that when the switch animation is performed or when the thumb is moved, the old state bitmap will disappear while the new state bitmap will be disclosed.



When set to **fade mode**, while the switch animation is performed or when the thumb is moved, the old state bitmap will fade into the new state bitmap.



8.24 Text

A text object is similar to its emWin counterpart, it is an object displaying a text resource or a decimal value at a specified position.

Symbol	Example
	Sample

Properties

Property	Description
<code>Text color</code>	Text color to be used.
<code>Background color</code>	Background color of the object.
<code>Framed font color</code>	Color of the object used for framed fonts.
<code>Text</code>	Text to be displayed.
<code>Decimal mode</code>	Enables decimal mode.
<code>Hexadecimal mode</code>	Enables hexadecimal mode.
<code>Text alignment</code>	Text alignment.
<code>Font</code>	Font to be used.
<code>Text wrapping</code>	Enables text wrapping.
<code>Text rotation</code>	Text rotation mode.
<code>Untouchable</code>	Sets the text object as untouchable and sends any touch input to the object below.

Decimal mode

Just as decimal mode for the Edit object, with this setting the Text object is only eligible of holding digits instead of characters. For this mode, a mask of zeros has to be specified which determines how many digits are shown by the object. More details about the usage of the mask is explained under *Decimal mode* on page 96.

Also, when using decimal mode, a range property is added to the object to limit the numbers that can be entered. More on the range property can be found under *Range* on page 114.

00123

8.25 Timer

A timer object represents a `GUI_TIMER` that can be set to a custom time period and optionally restarted.

The timer is an object, although unlike the other objects that represent widgets, the timer object does not have window-specific properties such as position and size. Because of that, it is also not visible on the screen.

However, it is visible in the hierarchical object tree on the left side of the AppWizard.





Properties

Property	Description
<code>Timer period</code>	Period of timer.
<code>Auto-restart mode</code>	When this mode is activated, the timer will be restarted again when it run out using the same time period.

8.26 Wheel

The Wheel object is a swipeable rotating list of items that can show multiple lists of texts and/or bitmaps.

There are two ways of how the Wheel can be created: **morph mode** and **plain mode**. Examples for each mode can be found further below.

Symbol	Example
	

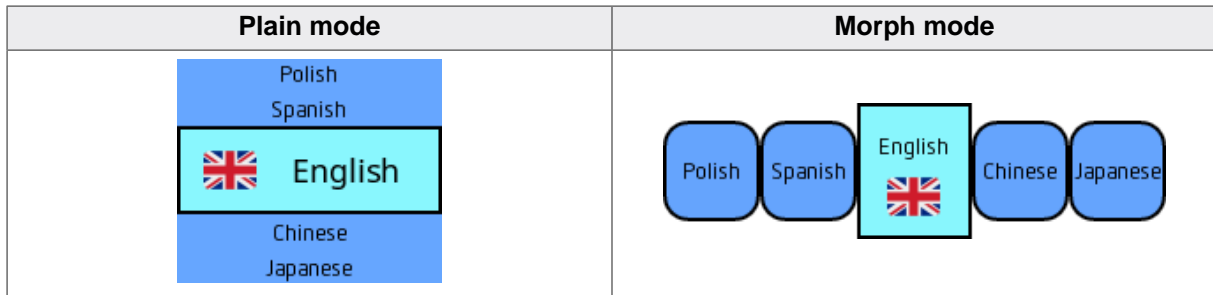
Properties

Property	Description
Morph mode	Toggles between morph mode and plain mode.
Endless mode	Toggles between endless mode and stop mode.
Horizontal mode	Sets the orientation of the widget to horizontal or vertical.
Period until stop	Period in ms it takes until the motion movement of the widget stops.
Cell- and center size	<ul style="list-style-type: none"> Cell: Width/height of non-center cells. Center: Width/height of center cells.
Colors	<ul style="list-style-type: none"> Background Cell: Background color of non-center cells. Background Center: Background color of center cells. Background: Background color used for the widget.
Text	Allows for adding lists of texts to the Wheel.
Images	Allows for adding lists of bitmaps to the Wheel.
Border	Horizontal/vertical border that can be used to reduce the cells in horizontal/vertical size.
Overlay bitmaps	Optional bitmaps that are drawn above the Wheel. A maximum of three overlay bitmaps can be added.
Overlay alignment	Alignment of the overlay bitmaps.
Overlay colors	Colors to be used for the overlay bitmaps (alpha bitmaps only).
Center frame	<ul style="list-style-type: none"> Center frame size: Frame width in pixels of the frame drawn around the center cell. Center frame radius: Frame radius in pixels of the frame drawn around the center cell.
Color of center frame	Color used to draw the frame around the center cell.
"Morph mode"-only properties	
Size of cutaway	Amount of pixels that are "cut away" (used as spacing) from the non-center cells.
Alignment of cutaway	Alignment of the cutaway.

Morph mode and plain mode

In plain mode, when a cell is moving into the center, its properties are clipped from the non-center properties to the center properties.

In morph mode, while a cell is moving into the center, its properties (like text position, bitmaps, frame radius, ...) are morphed into the different center properties.



Endless mode

If endless mode is active, the Wheel will not stop at the first or last cell when the user is swiping through the cells. Instead, it will scroll through the cells continuously.

If endless mode is toggled off, stop mode will be active which means that the Wheel will stop at the first and last item of the Wheel.

Text

All previously defined content objects defined in the lists window (see *Lists window* on page 52) can be added as text to the Wheel. It is possible to add multiple lists of text.

Based on whether the Wheel is in morph or plain mode, the following properties can be set to the text:

Cell type	Plain mode	Morph mode
Non-center cell	<ul style="list-style-type: none"> Text alignment Text alignment offset Text color Font 	<ul style="list-style-type: none"> Text alignment Text alignment offset Text color Font
Center cell	<ul style="list-style-type: none"> Text alignment Text alignment offset Text color Font 	<ul style="list-style-type: none"> Text alignment Text color

Content	Cell alignment	Cell Color	Cell Font	Center alignment	Center Color	Center Font
ID_CONTENT_001	Offset x: 0 Offset y: 0	[Color swatch]	NettoOT_16_NormalE...	Offset x: 65 Offset y: 0	[Color swatch]	NotoSans_24_Normal_EXT_AA4
ID_CONTENT_002	Offset x: 10 Offset y: 0	[Color swatch]	NettoOT_16_NormalE...	Offset x: 0 Offset y: 0	[Color swatch]	NotoSans_24_Normal_EXT_AA4

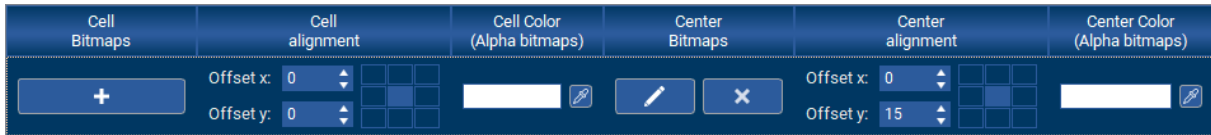
Images

Similarly to the text lists, multiple lists of bitmaps can also be added to the Wheel.

Bitmaps can be added to the non-center cells and/or the center cells.

The following properties can be set to center and non-center bitmaps:

- Bitmap alignment
- Bitmap alignment offset
- Bitmap color (only for alpha bitmaps)



Overlay

The overlay feature allows a maximum of three bitmaps to be added to the Wheel. These overlay bitmaps are drawn at a certain position above the Wheel items.

This can be used to create certain visual effects, e.g. a semi-transparent bitmap can be used as an overlay to highlight the center cell, as shown below.

Without overlay	With overlay


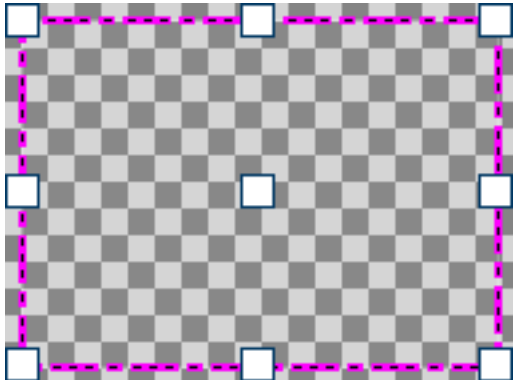
Cutaway

Cutaway is the distance that is removed from the non-center cells in morph mode. The cutaway alignment can be used to define how the cutaway distance should be distributed.

Cutaway options	Output
No cutaway	
Cutaway of 25px, top aligned	
Cutaway of 25px, center aligned	
Cutaway of 25px, bottom aligned	

8.27 Window

A window works similar to a screen. It is also invisible and serves as parent object for objects. Moving/animating the window also moves its objects. A window can have further child windows. That makes it possible to achieve a hierarchic structure for complex dialogs.

Symbol	Example
	

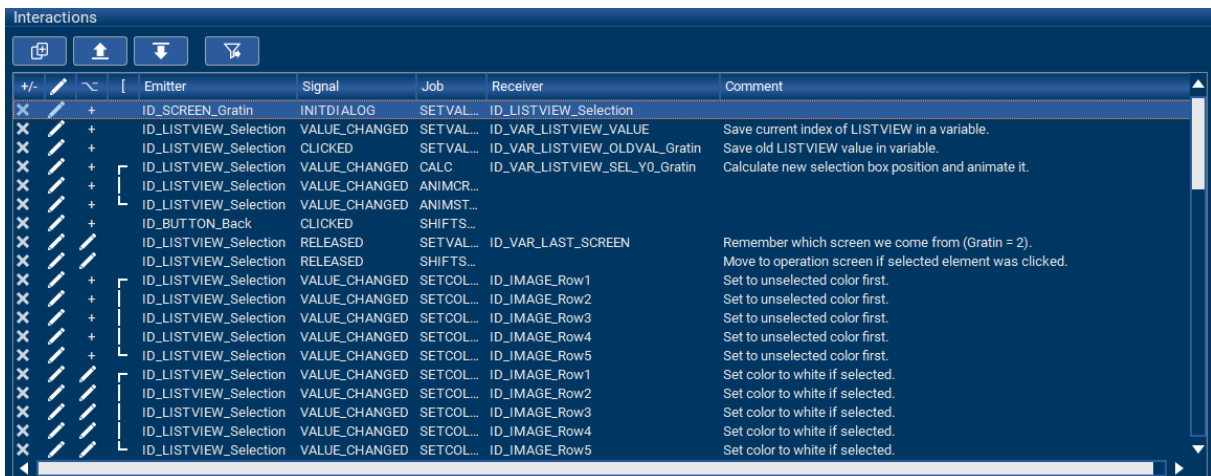
Properties

Property	Description
Horizontal motion	Horizontal motion screen/window partner.
Vertical motion	Vertical motion screen/window partner.
Opaque mode	Sets the button to opaque and removes its transparency flag.

Chapter 9

Interactions

The AppWizard's interaction window makes it possible to define the application's behavior on certain actions. Interactions are always assigned to a screen, meaning two different screens have different interactions.



The screenshot shows the 'Interactions' window in AppWizard. It contains a table with the following columns: +/-, a pencil icon, a checkmark, an expand/collapse icon, Emitter, Signal, Job, Receiver, and Comment. The table lists various interactions for different UI elements like ID_SCREEN_Gratin, ID_LISTVIEW_Selection, and ID_BUTTON_Back, detailing the signals and jobs triggered by these elements.

+/-				Emitter	Signal	Job	Receiver	Comment
X	/	+		ID_SCREEN_Gratin	INITDIALOG	SETVAL...	ID_LISTVIEW_Selection	
X	/	+		ID_LISTVIEW_Selection	VALUE_CHANGED	SETVAL...	ID_VAR_LISTVIEW_VALUE	Save current index of LISTVIEW in a variable.
X	/	+		ID_LISTVIEW_Selection	CLICKED	SETVAL...	ID_VAR_LISTVIEW_OLDVAL_Gratin	Save old LISTVIEW value in variable.
X	/	+		ID_LISTVIEW_Selection	VALUE_CHANGED	CALC	ID_VAR_LISTVIEW_SEL_Y0_Gratin	Calculate new selection box position and animate it.
X	/	+		ID_LISTVIEW_Selection	VALUE_CHANGED	ANIMCR...		
X	/	+		ID_LISTVIEW_Selection	VALUE_CHANGED	ANIMST...		
X	/	+		ID_BUTTON_Back	CLICKED	SHIFTS...		
X	/	+		ID_LISTVIEW_Selection	RELEASED	SETVAL...	ID_VAR_LAST_SCREEN	Remember which screen we come from (Gratin = 2).
X	/	+		ID_LISTVIEW_Selection	RELEASED	SHIFTS...		Move to operation screen if selected element was clicked.
X	/	+		ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row1	Set to unselected color first.
X	/	+		ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row2	Set to unselected color first.
X	/	+		ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row3	Set to unselected color first.
X	/	+		ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row4	Set to unselected color first.
X	/	+		ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row5	Set to unselected color first.
X	/	+		ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row1	Set color to white if selected.
X	/	+		ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row2	Set color to white if selected.
X	/	+		ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row3	Set color to white if selected.
X	/	+		ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row4	Set color to white if selected.
X	/	+		ID_LISTVIEW_Selection	VALUE_CHANGED	SETCOL...	ID_IMAGE_Row5	Set color to white if selected.

9.1 Introduction

This section will explain how to set up interactions and describe the terms.

❶ Select an emitter

First, an emitter for a signal has to be selected. The **emitter** specifies the ID of the widget or variable that has to send out a certain signal in order for the interaction's job to be executed.

❷ Select the signal

The second step is to select the signal. The **signal** is the event that has to occur for the job to be executed. This could be e.g. `WM_NOTIFICATION_CLICKED`, which occurs when a widget was clicked.

For a list of all available signals, see the chapter *List of signals* on page 167.

❸ Select the job

The third step is to select a job for this interaction. The **job** specifies a certain action that will be done when the above mentioned signal has occurred. This could for example be `SETTEXT` to set the text of an Edit object.

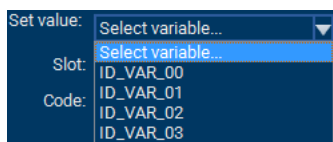
For a list of all available jobs, see the chapter *List of jobs* on page 190.

❹ Select the receiver

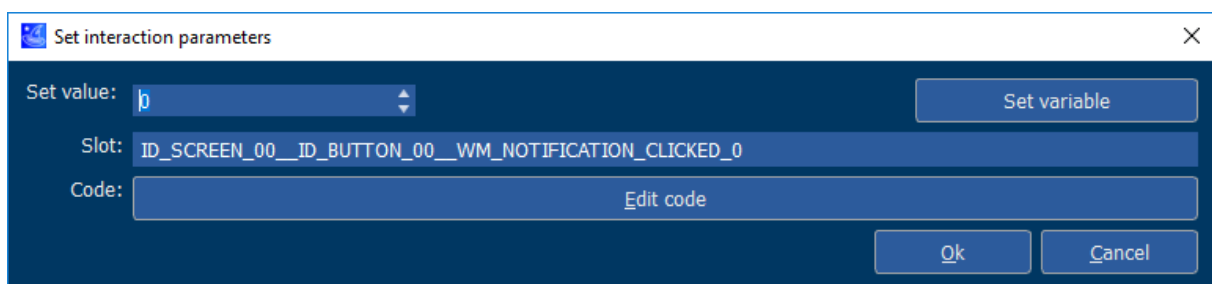
The last step is to select is a receiver for the interaction. The **receiver** specifies the ID of the widget or variable the job will be executed for. For example, if the job is `SETTEXT`, the receiver has to be an Edit object, whose text will then be set.

❺ Set up interaction parameters

The final step is to define what the action/job should do with the receiver. This can be done by clicking on the 'Edit' symbol of an interaction to set up interaction-specific parameters. For example for the job `ADDVALUE`, the user has to specify the value that will be added to the receiver.



Instead of a permanent value, the user is also able to select a variable. To do this, click the **Set variable** button and select a variable from the dropdown menu.



In the 'Slot' field, the user can see and may change the name of the slot routine. The slot routine is the routine, that will be executed for this interaction.

Note

The name of the slot routine must be unique! Otherwise the user code won't compile.

6 Add a condition to the interaction (optional)

Optionally, a condition can be set up for the interaction. This condition determines whether or not the job of the interaction will be executed.

To add a condition to an interaction, click the plus symbol in the condition column. More information about conditions can be read under *Conditions* on page 236.

+/-	Edit		Emitter	Signal	Job	Receiver
X	/	+	ID_BUTTON_00	RELEASED	SETVALUE	ID_VAR_00
X	/	/	ID_SLIDER_00	VALUE_CHANGED	SETCOLOR	ID_BOX_00

7 Add custom user code to the interaction (optional)

The user may edit/insert C code that will be executed upon this interaction. The code may be added via the "Edit code" dialog or externally via an editor or IDE. More information about slot routines and where they are located can be read in the chapter *Slot routines* on page 258.

Note

The user must not add custom routines to the C files that contain the generated slot routines! More information about how the user can properly add their own code can be read under *Custom user code* on page 260.

9.2 Tools for large projects

When creating screens with a large number of interactions, it can be very tedious to find a specific interaction. For this reason, we have added a filter to the Interactions window that makes it easy to reduce the number of visible interactions. We have also added the possibility to disable interactions. This can be useful for testing purposes, for example.

9.2.1 Using the filter

To opening the filter the filter button needs to be pressed:



This opens the filter:



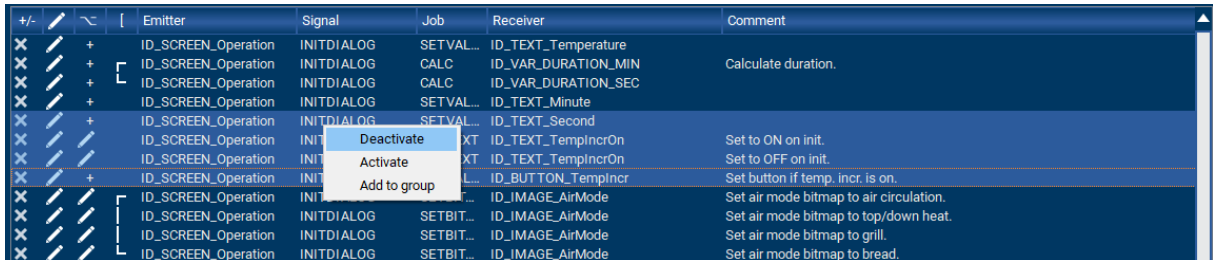
It shows the following fields:

- Emitter
- Signal
- Job
- Receiver
- Comment

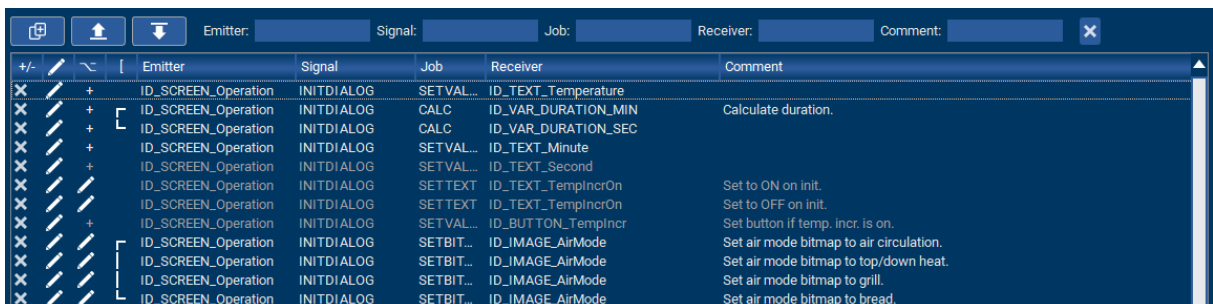
Entering text in one or more of them will reduce the shown interactions to those which include the given text. Please note that this is not case sensitive. Interactions that are hidden by the filter still remain active and are still executed and saved.

9.2.2 Deactivating interactions

For test purposes it could be helpful to be able to deactivate one or more interactions. This can be done by selecting them and then open the context window by right clicking the selection:



Deactivated interactions are shown in gray:



Deactivated interactions remain in the project but they will be ignored when executing the application.

9.3 List of signals

The following section will provide a list of all available signals the user can choose from for an interaction.

Signal	Description
ANIMEND	Emitted when an animation has ended.
ANIMSTART	Emitted when an animation has started.
CLICKED	When the user clicks on an object.
CREATE	Emitted when an object was created.
DELETE	Emitted when an object was deleted.
ENTER_PRESSED	Emitted when ENTER key is pressed.
FIXED	Emitted by a screen or window when a shifting operation has ended.
GOT_FOCUS	When an object gets the focus.
INITDIALOG	Emitted right after the application has started.
LOST_FOCUS	When an object lost its focus.
MOTION	Emitted when a user moves a screen by dragging it.
MOTION_STOPPED	When the motion of a Rotary object has stopped.
PIDPRESSED	Screen received touch input (pressed).
PIDRELEASED	Screen lost touch input (released).
RELEASED	Once a click on an object has been released.
ROTATED	Emitted by screens after the display orientation has been changed.
STARTED	Emitted after a movie begins playing.
STOPPED	Emitted after a movie has stopped playing.
TEXT_CHANGED	Emitted when the text of an object has changed.
TIMER	Emitted when a given timer has run out.
UNPINNED	Emitted by a screen or window when a shifting operation has started.
VALUE_CHANGED	If the value of an object has changed.

9.3.1 ANIMEND

Description

This signal is emitted by an object after an animation paired to the object has ended.

Emitting objects

- *Box* object
- *Button* object
- *Edit* object
- *Gauge* object
- *Image* object
- *Progbar* object
- *Rotary* object
- *Slider* object
- *Switch* object
- *Text* object
- *Variables*
- *Window* object

9.3.2 ANIMSTART

Description

This signal is emitted by an object after an animation paired to the object has started.

Emitting objects

- *Box* object
- *Button* object
- *Edit* object
- *Gauge* object
- *Image* object
- *Progbar* object
- *Slider* object
- *Switch* object
- *Text* object
- *Variables*
- *Window* object

9.3.3 CLICKED

Description

This signal is emitted when the user clicks on an object.

Emitting objects

- *Button* object
- *Edit* object
- *Rotary* object
- *Slider* object
- *Switch* object
- *Wheel* object

9.3.4 CREATE

Description

This signal is emitted right after an object has been created. The Window Manager equivalent is `WM_CREATE`.

Emitting objects

- *Screen* object

9.3.5 DELETE

Description

This signal is emitted right after an object has been deleted. The Window Manager equivalent is `WM_DELETE`.

Emitting objects

- *Screen* object

9.3.6 ENTER_PRESSED

Description

This signal is emitted when the ENTER key is pressed.

Emitting objects

- *Keyboard* object

9.3.7 FIXED

Description

This signal is emitted by screens or windows when a shifting operation has ended.

Emitting objects

- *Screen* object
- *Window* object

See also

- UNPINNED
- SHIFTSCREEN
- SHIFTWINDOW

9.3.8 GOT_FOCUS

Description

This signal is emitted when an object has gotten the focus.

Emitting objects

- *Button* object
- *Edit* object
- *Multiedit* object
- *Rotary* object
- *Slider* object
- *Switch* object
- *Wheel* object

9.3.9 INITDIALOG

Description

This signal is emitted right after the application has started. The Window Manager equivalent is `WM_INIT_DIALOG`.

Emitting objects

- *Screen* object

9.3.10 LOST_FOCUS

Description

This signal is emitted when an object lost its focus.

Emitting objects

- *Button* object
- *Edit* object
- *Multiedit* object
- *Rotary* object
- *Slider* object
- *Switch* object
- *Wheel* object

9.3.11 MOTION

Description

This signal is emitted when a screen object has been moved by the user dragging it. The Window Manager equivalent is `WM_MOTION`.

Emitting objects

- *Screen* object

9.3.12 MOTION_STOPPED

Description

This signal is emitted when the motion of a Rotary object has stopped.

Emitting objects

- *Rotary* object

9.3.13 PIDPRESSED

Description

This signal is emitted by screens when PID is pressed. The signal is sent to all existing screens in the project.

Emitting objects

- *Screen* object

9.3.14 PIDRELEASED

Description

This signal is emitted by screens when PID is released. The signal is sent to all existing screens in the project.

Emitting objects

- *Screen* object

9.3.15 ROTATED

Description

This signal is emitted by screens after the display orientation has been changed. The signal is sent to all existing screens in the project.

Emitting objects

- *Screen* object

9.3.16 STARTED

Description

This signal is emitted by a movie after it has been started.

Emitting objects

- *Movie* object

9.3.17 STOPPED

Description

This signal is emitted by a movie after it has been stopped.

Emitting objects

- *Movie* object

9.3.18 RELEASED

Description

This signal is emitted once a click on an object has been released.

Emitting objects

- *Button* object
- *Edit* object
- *Rotary* object
- *Slider* object
- *Switch* object
- *Wheel* object

9.3.19 TEXT_CHANGED

Description

This signal is emitted when the text of an object has been changed.

Emitting objects

- *Edit* object
- *QRCode* object
- *Text* object

9.3.20 TIMER

Description

This signal is emitted when a given timer has run out. To run a timer, a timer object has to be created and started using the job `START`.

Emitting objects

- *Timer* object

9.3.21 UNPINNED

Description

This signal is emitted by screens or windows when a shifting operation has started.

Emitting objects

- *Screen* object
- *Window* object

See also

- FIXED
- SHIFTSCREEN
- SHIFTWINDOW

9.3.22 VALUE_CHANGED

Description

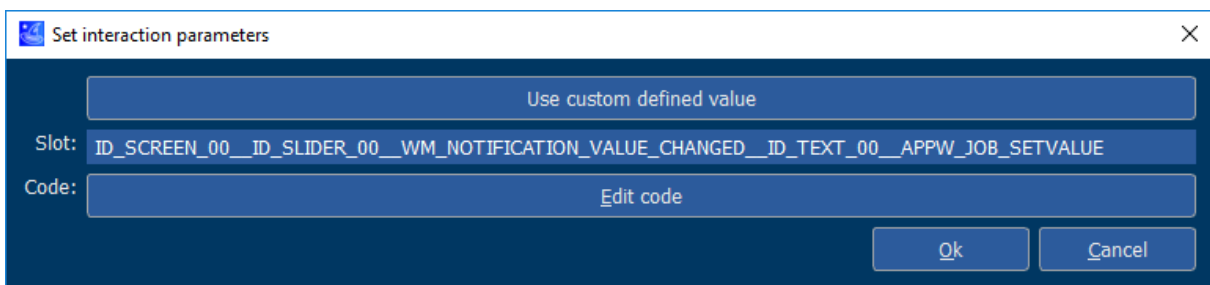
This signal is emitted when the value of an object has changed.

Emitting objects

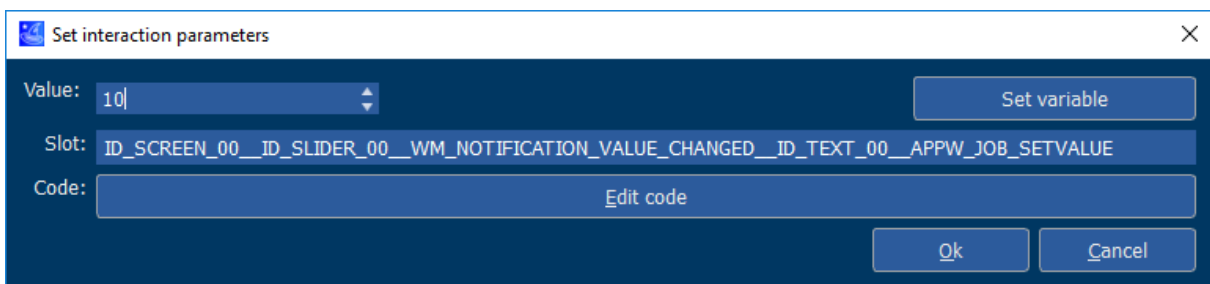
- *Button* object
- *Edit* object
- *Gauge* object
- *Progbar* object
- *Rotary* object
- *Slider* object
- *Switch* object
- *Text* object
- *Variables*
- *Wheel* object

Additional information

By default, the custom value option is disabled. This means, the value of the emitting object will be directly passed to the receiver and process the value depending on the selected job. This can be useful for jobs like SETVALUE, but it certainly does not work for all jobs.



When clicking the button **Use custom defined value**, a custom value can be entered, which will be sent to the receiver.



9.4 List of jobs

The table below shows all jobs available for interactions. These jobs can either be used within the AppWizard tool in the interaction window or executed by user code with the function `APPW_DoJob()`.

Job	Description
<code>ADDVALUE</code>	Adds a given increment to the given object.
<code>ADDITEM</code>	Appends text from object/resource to a given object.
<code>ANIMCREATE</code>	Creates an animation.
<code>ANIMSTART</code>	Starts an animation.
<code>ANIMSTOP</code>	Stops a running animation.
<code>CALC</code>	Calculates the value of a variable using the given term.
<code>CLEAR</code>	Clears the state of the given object.
<code>CLOSESCREEN</code>	Closes a given screen to go back to the screen that is behind.
<code>DELITEM</code>	Deletes the specified item from the given object.
<code>ENABLEPID</code>	Enables or disables PID input.
<code>INSITEM</code>	Insert text to a given object.
<code>INVALIDATE</code>	Invalidates a given object.
<code>MODALMESSAGE</code>	Creates and shows a modal dialog.
<code>MOVETO</code>	Moves an object's selection to a given value with an animation.
<code>ROTATEDISPLAY</code>	Rotates the display to the desired orientation.
<code>SET</code>	Sets the state of the given object.
<code>SETALPHA</code>	Sets the alpha value of an object.
<code>SETANGLE</code>	Sets the rotation angle of an object.
<code>SETBITMAP</code>	Sets a bitmap to an object.
<code>SETBKCOLOR</code>	Sets the background color of the given object.
<code>SETCOLOR</code>	Sets the color of the given object.
<code>SETCOORD</code>	Sets a coordinate.
<code>SETENABLE</code>	Enables the given object.
<code>SETEND</code>	Sets the end value/angle of the given object.
<code>SETFOCUS</code>	Sets focus to a given object.
<code>SETITEM</code>	Text transfer from/to a given object.
<code>SETLANG</code>	Sets the language index of an object.
<code>SETPERIOD</code>	Sets the period of an object.
<code>SETRANGE</code>	Sets the range of the given object.
<code>SETSCALE</code>	Sets a scaling value to an object.
<code>SETSIZE</code>	Sets the size of the given object.
<code>SETSTART</code>	Sets the start value/angle of the given object.
<code>SETTEXT</code>	Sets the text of the given object.
<code>SETVALUE</code>	Sets a value.
<code>SETVIS</code>	Makes the given object visible.
<code>SETX0</code>	Sets the x0-coordinate of an object.
<code>SETY0</code>	Sets the y0-coordinate of an object.

Job	Description
SETX1	Sets the x1-coordinate of an object.
SETY1	Sets the y1-coordinate of an object.
SHIFTSCREEN	Shifts into the given screen using the given method.
SHIFTWINDOW	Shifts in a window using the given method.
SHOWSCREEN	Makes the given screen visible.
START	Starts a given timer object.
STOP	Stops a given timer object.
TOGGLE	Toggles the state of the given object.
NULL	Used for only executing custom user code.

9.4.1 ADDVALUE

Description

Adds a given increment to the given object.

Receiving objects

- *Text* object
- *Progbar* object
- *Rotary* object

Interaction parameters of dialog

Parameter	Description
<code>Value</code>	Value to be added.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Value to be added.

9.4.2 ADDITEM

Description

This job is used for the transaction of text. The source of the text to be added can be either a text from the text resources or a text-based object from any screen. It should be noted that the corresponding object must be in memory. The object specified as 'Receiver' serves as the target of the transaction. The following dialog is used to specify the job:

The fields 'Source row' and 'Source column' are not used in any case. If the source is an object with only one field, such as an EDIT or a TEXT object, both specifications are ignored. If the source is a LISTBOX or a DROPDOWN object, only 'Source row' is used. Only with a LISTVIEW object both specifications are used. The same applies to the target object. The 'Destination column' field is only needed if the target object is a LISTVIEW object.

Receiving objects

- *Dropdown* object
- *Listbox* object
- *Listview* object
- *Wheel* object

Interaction parameters of dialog

Parameter	Description
Type of resource	Determines if an object or a text resource should be used.
Text/object id	Screen- and object id of object or resource id of text.
Source row	Used in case of a source object with multiple rows.
Source column	Used in case of a source object with multiple columns.
Destination column	Column to add the text to. Only used if target object is a LISTVIEW.

Job-specific parameters passed to slot-routine

Parameter	Description
aPara[0].v	Type: 0 = text resource, 1 = object
aPara[1].v	0: text resource Id, 1: HB/LB: Screen Id/Object Id
aPara[2].v	Source: Row index (Listview, Dropdown, Listbox)
aPara[3].v	Source: Column index (Listview only)
aPara[4].v	Destination: Column index (Listview only)

9.4.3 ANIMCREATE

Description

Creates an animation that has previously been defined in the animation interface.

Interaction parameters of dialog

Parameter	Description
<code>Animation Id</code>	ID of the predefined animation.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	ID of the predefined animation.

Additional information

More information about how animations can be created can be read under *Animations* on page 245.

9.4.4 ANIMSTART

Description

Starts an animation that has been previously defined and created with the job ANIMCREATE.

Interaction parameters of dialog

Parameter	Description
<code>Animation Id</code>	ID of the animation.
<code>Number of loops</code>	Number of loops the animation should run. -1 if it should run endlessly.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	ID of the animation.
<code>aPara[1].v</code>	Number of loops the animation should run. -1 if it should run endlessly.

9.4.5 ANIMSTOP

Description

Creates an animation that has previously been defined in the animation interface.

Interaction parameters of dialog

Parameter	Description
<code>Animation Id</code>	ID of the animation.
<code>Delete animation</code>	If the animation should be deleted after it has been stopped.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	ID of the animation.
<code>aPara[1].v</code>	1 if the animation should be deleted after stopping, 0 if it should remain.

9.4.6 CALC

Description

Calculates the new value of a variable using the set term. If this results in a changed value, the variable will emit a `VALUE_CHANGED` signal.

Receiving objects

- *Variables*

Term calculation

A detailed description on how a calculation term can be added to a variable can be found under Calculations.

9.4.7 CLEAR

Description

Sets the state of the given object to its default state. For example, when executing this job on a *Switch* object, it will be set to the 'left state'.

Receiving objects

- *Button* object
- *Switch* object

9.4.8 CLOSESCREEN

Description

Closes a given screen. When the screen is closed, the screen that was behind is shown again.

Interaction parameters of dialog

Parameter	Description
<code>Screen ID</code>	ID of the screen to be closed.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Screen Id.

Additional information

It should be made sure of that there is another screen behind the screen to be deleted, otherwise nothing will be shown. Therefore, the screen opening the screen that performs the `CLOSESCREEN` job should not be moved out via `SHIFTSCREEN`. Rather, the other screen should be shown using `SHOWSCREEN`.

9.4.9 DELITEM

Description

Deletes the given row from the given object.

Receiving objects

- *Dropdown* object
- *Listbox* object
- *Listview* object
- *Wheel* object

Interaction parameters of dialog

Parameter	Description
<code>Row index</code>	Index of row to be deleted.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Item index to be deleted.

9.4.10 ENABLEPID

Description

Enables, disables or toggles PID input for the application. This job has no receiving object, since it will alter the state of PID input for the entire application.

Interaction parameters of dialog

Parameter	Description
<code>Enabled state</code>	State of PID input for the job: <i>On</i> , <i>Off</i> or <i>Toggle</i> .

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	PID input. 1 = On, 0 = Off.

9.4.11 INSITEM

Description

This job is used to insert text into the given object. The source of the text to be set can be either a text from the text resources or a text-based object from any screen. It should be noted that the corresponding object must be in memory. The object specified as 'Receiver' serves as the target of the transaction. The following dialog is used to specify the job:

The fields 'Source row' and 'Source column' are not used in any case. If the source is an object with only one field, such as an EDIT or a TEXT object, both specifications are ignored. If the source is a LISTBOX or a DROPDOWN object, only 'Source row' is used. Only with a LISTVIEW object both specifications are used. The same applies to the target object.

Receiving objects

- *Dropdown* object
- *Listbox* object
- *Listview* object
- *Wheel* object

Interaction parameters of dialog

Parameter	Description
Type of resource	Determines if an object or a text resource should be used.
Text/object id	Screen- and object id of object or resource id of text.
Source row	Used in case of a source object with multiple rows
Source column	Used in case of a source object with multiple columns
Destination row	Used in case of a destination object with multiple rows
Destination column	Used in case of a destination object with multiple columns

Job-specific parameters passed to slot-routine

Parameter	Description
aPara[0].v	Type: 0 = Text resource, 1 = Object
aPara[1].v	0: Text resource Id, 1: HB/LB: Screen Id/Object Id
aPara[2].v	Source: Row index (Listview, Dropdown, Listbox)
aPara[3].v	Source: Column index (Listview only)
aPara[4].v	Destination: Row index (Listview, Dropdown, Listbox)
aPara[5].v	Destination: Column index (Listview only)

9.4.12 INVALIDATE

Description

Triggers a redraw (invalidates) the given object.

Receiving objects

This job can be executed for all objects.

9.4.13 MODALMESSAGE

Description

Shows a given screen above the current screen as a modal message dialog. To hide the modal screen, the job `CLOSESCREEN` has to be executed on the modal screen.

Receiving objects

- *Screen* object

Interaction parameters of dialog

Parameter	Description
<code>Screen ID</code>	ID of the screen to be shown as a modal message.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Screen ID.

9.4.14 MOVETO

Description

Moves an object's selection to a given value with an animation.

Receiving objects

- *Wheel* object

Interaction parameters of dialog

Parameter	Description
<code>Value</code>	New index.

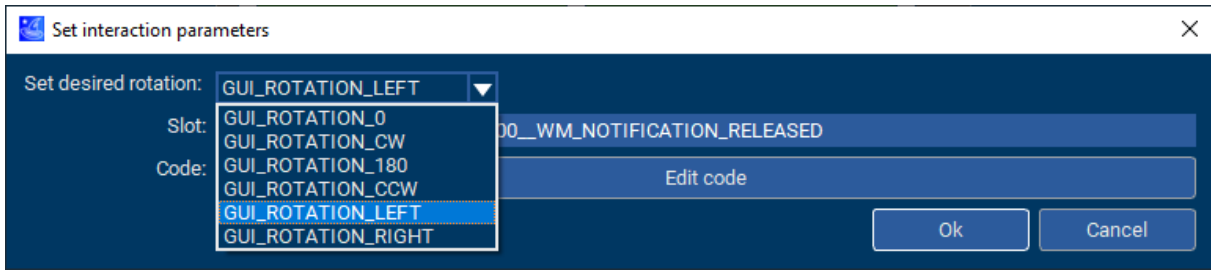
Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	New index.

9.4.15 ROTATEDISPLAY

Description

Rotates the display to the desired orientation.



The field 'Set desired orientation' allows to set a specific orientation or turn to the display clockwise (right) or counterclockwise (left).

Interaction parameters of dialog

Parameter	Description
Rotation	Rotation command.

Job-specific parameters passed to slot-routine

Parameter	Description
aPara[0].v	Rotation command

Rotation command

The following table shows the available commands:

Command	Description
GUI_ROTATION_0	Default orientation
GUI_ROTATION_CW	Clockwise rotation 90°
GUI_ROTATION_180	Upside down 180°
GUI_ROTATION_CCW	Counterclockwise rotation 90°
GUI_ROTATION_LEFT	Counterclockwise rotation to the left by 90°
GUI_ROTATION_RIGHT	Clockwise rotation to the right by 90°

Additional information

Note that in case of using hardware acceleration like D/AVE 2D or Chrom-ART the hardware acceleration will be disabled in other rotation modes than the default orientation.

9.4.16 SET

Description

Sets the state of the given object to its "pressed" state. This means, e.g. when executed on a Button object, it will be in its pressed state and when executed on a Switch object it will be in its 'right state'.

Receiving objects

- *Button* object
- *Switch* object

9.4.17 SETALPHA

Description

Sets the alpha value of an Image object. This job only has an effect if an alpha bitmap is set to the Image object.

Receiving objects

- *Image* object

Interaction parameters of dialog

Parameter	Description
Alpha	New alpha value to be set.

Job-specific parameters passed to slot-routine

Parameter	Description
aPara[0].v	New alpha value to be set.

9.4.18 SETANGLE

Description

Sets the rotation angle of an Image object.

Receiving objects

- *Image* object

Interaction parameters of dialog

Parameter	Description
<code>Angle</code>	New rotation angle to be set, angle in degrees * 1000 (e.g. 45000 equals 45°).

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	New rotation angle to be set.

9.4.19 SETBITMAP

Description

Sets a bitmap to an image or button object.

Receiving objects

- *Button* object
- *Image* object

Interaction parameters of dialog

Parameter	Description
Bitmap	New bitmap to be set.
Index	Index of button bitmap. See below for more information.

Bitmap index

The bitmap index parameter is only used if the receiving object is a Button. The index is used to determine the state the bitmap is used for.

Index	Bitmap
0	Bitmap for unpressed state.
1	Bitmap for pressed state.
2	Bitmap for disabled state.

9.4.20 SETBKCOLOR

Description

Sets the background color of the given object.

Receiving objects

- *Button* object

Interaction parameters of dialog

Parameter	Description
<code>Background color</code>	New background color to be used.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Background color to be used.

9.4.21 SETCOLOR

Description

Sets the color of the given object.

Receiving objects

- *Box* object
- *Button* object
- *Edit* object
- *Gauge* object
- *Image* object
- *Multiedit* object
- *Text* object

Interaction parameters of dialog

Parameter	Description
<code>Color</code>	New color to be used.
<code>Index</code>	Index of color. See table below.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Color to be used.
<code>aPara[1].v</code>	Index of state.

Setting the color of alpha bitmaps

If an alpha bitmap is added to an Image object, the color used for drawing the bitmap can be changed using this job. More information can be found under Image.

Color index

The color index parameter depends on the receiving object.

BUTTON

Index	Color
0	Color for unpressed state.
1	Color for pressed state.
2	Color for disabled state.

GAUGE

Index	Color
0	Color for background line.
1	Color for foreground line.

9.4.22 SETCOORD

Description

Sets a coordinate of an object.

Receiving objects

- *Box* object
- *Button* object
- *Edit* object
- *Image* object
- *Progbar* object
- *Rotary* object
- *Slider* object
- *Switch* object
- *Text* object
- *Window* object

Interaction parameters of dialog

Parameter	Description
Value	New coordinate of the object.
Coordinate	Axis of the coordinate to be set.

Job-specific parameters passed to slot-routine

Parameter	Description
aPara[0].v	Value.
aPara[1].v	Index of coordinate. See <i>Dispose indexes</i> on page for a list of legal values.

9.4.23 SETENABLE

Description

Sets the 'enabled' state of a given object. The receiving object will be either enabled or disabled, depending which 'enabled' state was specified in the interaction parameters.

Receiving objects

- *Button* object
- *Edit* object
- *Rotary* object
- *Slider* object
- *Switch* object

Interaction parameters of dialog

Parameter	Description
<code>Enable state</code>	New enable state of the object. This can be set to either on, off or toggled.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Enable state. 1 = on, 0 = off.

9.4.24 SETEND

Description

Sets the end angle of a given object.

Receiving objects

- *Gauge* object

Interaction parameters of dialog

Parameter	Description
Ang1	End angle.

Job-specific parameters passed to slot-routine

Parameter	Description
aPara[0].v	End angle in 10th of degrees.

9.4.25 SETFOCUS

Description

Sets the focus onto a given object.

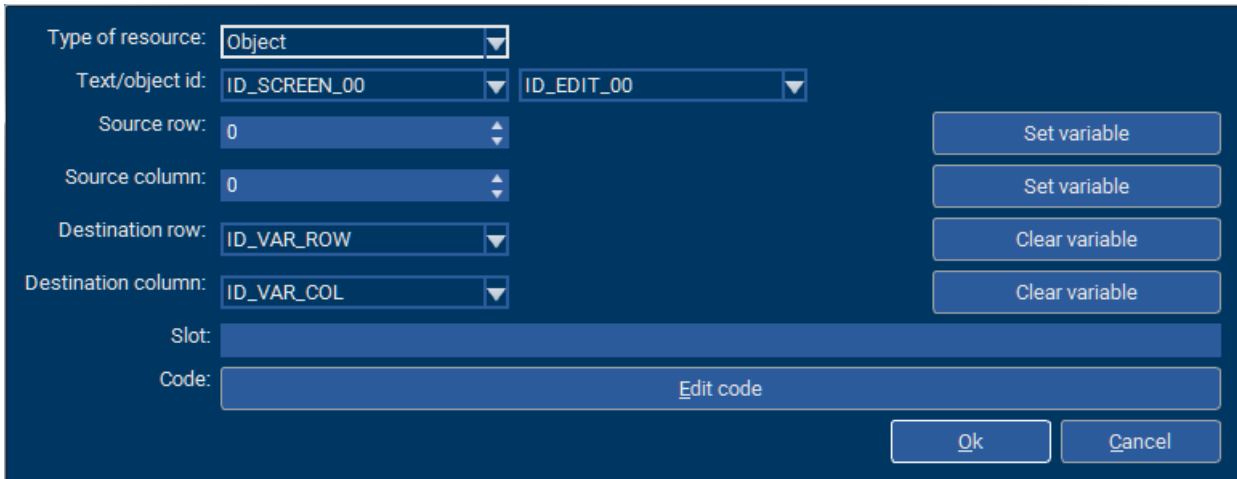
Receiving objects

- *Button* object
- *Edit* object
- *Multiedit* object
- *Rotary* object
- *Slider* object
- *Switch* object

9.4.26 SETITEM

Description

This job is used for the transaction of text. The source of the text to be set can be either a text from the text resources or a text-based object from any screen. It should be noted that the corresponding object must be in memory. The object specified as 'Receiver' serves as the target of the transaction. The following dialog is used to specify the job:



The fields 'Source row' and 'Source column' are not used in any case. If the source is an object with only one field, such as an EDIT or a TEXT object, both specifications are ignored. If the source is a LISTBOX or a DROPDOWN object, only 'Source row' is used. Only with a LISTVIEW object both specifications are used. The same applies to the target object.

Receiving objects

- *Button* object
- *Edit* object
- *Dropdown* object
- *Listbox* object
- *Listview* object
- *Multiedit* object
- *QRCode* object
- *Text* object
- *Wheel* object

Interaction parameters of dialog

Parameter	Description
Type of resource	Determines if an object or a text resource should be used.
Text/object id	Screen- and object id of object or resource id of text.
Source row	Used in case of a source object with multiple rows
Source column	Used in case of a source object with multiple columns
Destination row	Used in case of a destination object with multiple rows
Destination column	Used in case of a destination object with multiple columns

Job-specific parameters passed to slot-routine

Parameter	Description
aPara[0].v	Type: 0 = Text resource, 1 = Object
aPara[1].v	0: Text resource Id, 1: HB/LB: Screen Id/Object Id
aPara[2].v	Source: Row index (Listview, Dropdown, Listbox)

Parameter	Description
<code>aPara[3].v</code>	Source: Column index (Listview only)
<code>aPara[4].v</code>	Destination: Row index (Listview, Dropdown, Listbox)
<code>aPara[5].v</code>	Destination: Column index (Listview only)

9.4.27 SETLANG

Description

Sets the language of the application to the given index.

Interaction parameters of dialog

Parameter	Description
<code>Language index</code>	Index of the new language to be set. The index is the zero-based column number of the language seen in the text management dialog.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Index of language.

9.4.28 SETPERIOD

Description

Sets the period of an object.

Receiving objects

- *Wheel* object

Interaction parameters of dialog

Parameter	Description
<code>Period</code>	New period to be set.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	New period to be set.

9.4.29 SETRANGE

Description

Sets the range to be used for the given object.

Receiving objects

- *Gauge* object
- *Rotary* object
- *Progbar* object
- *Slider* object

Interaction parameters of dialog

Parameter	Description
<code>Start</code>	Start value to be used.
<code>End</code>	End value to be used.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Start value.
<code>aPara[1].v</code>	End value.

9.4.30 SETSCALE

Description

Sets a scaling value to an Image object.

Receiving objects

- *Image* object

Interaction parameters of dialog

Parameter	Description
<code>Value</code>	New scaling value to be set, scaling factor * 1000 (e.g. 2000 equals 200% scale).

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	New scaling value to be set.

9.4.31 SETSIZE

Description

Sets the size of the given object.

Receiving objects

- *Box* object
- *Button* object
- *Edit* object
- *Image* object
- *Progbar* object
- *Rotary* object
- *Slider* object
- *Switch* object
- *Text* object
- *Window* object

Interaction parameters of dialog

Parameter	Description
<code>Value</code>	New size value.
<code>Dimension</code>	Either X- or Y-axis where the new size value should be applied to.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Value to be used.
<code>aPara[1].v</code>	Index of axis.

Additional information

In order for this job to work, the size of the object must be editable. If all coordinates are relative, there is no size to be edited.

9.4.32 SETSTART

Description

Sets the start angle of a given object.

Receiving objects

- *Gauge* object

Interaction parameters of dialog

Parameter	Description
<code>Ang0</code>	Start angle.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Start angle in 10th of degrees.

9.4.33 SETTEXT

Description

Sets the text of a given object.

Receiving objects

- *Text* object
- *Button* object

Interaction parameters of dialog

Parameter	Description
Text	ID of the text to be used.

Job-specific parameters passed to slot-routine

Parameter	Description
aPara[0].v	Text Id. Only if <code>aPara[0].p = NULL</code> .
aPara[0].p	Handle. Only if <code>aPara[0].v < 0</code> .

9.4.34 SETVALUE

Description

With this job the value of an object can be set. For most objects, this is a numerical value, except for the Text and Edit objects, where this job sets the corresponding text.

Receiving objects

- *Button* object
- *Edit* object
- *Gauge* object
- *Progbar* object
- *Rotary* object
- *Slider* object
- *Switch* object
- *Text* object
- *Variables*
- *Wheel* object

Interaction parameters of dialog

Parameter	Description
<code>Value</code>	New value or text to be set to the object.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Value to be set.

Additional information

Instead of a permanent value, the user can also choose a variable.

9.4.35 SETVIS

Description

Sets the visibility of the given object to either on or off.

Receiving objects

- *Box* object
- *Button* object
- *Edit* object
- *Image* object
- *Progbar* object
- *Rotary* object
- *Slider* object
- *Switch* object
- *Text* object
- *Window* object

Interaction parameters of dialog

Parameter	Description
<code>Visibility</code>	New visibility of the object. This can be either set to on, off or toggled.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Visibility flag. 1 = on, 0 = off.

9.4.36 SETX0**9.4.37 SETY0****9.4.38 SETX1****9.4.39 SETY1****Description**

Sets the corresponding coordinate of an object.

Interaction parameters of dialog

Parameter	Description
<code>Value</code>	Value to be set.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Value to be set.

9.4.40 SHIFTSCREEN

Description

Shifts into the given screen with an animation that the user defines.

Receiving objects

- *Screen* object

Interaction parameters of dialog

Parameter	Description
Screen ID	ID of the screen to be shifted in.
Edge	Edge the old screen should be moved to.
Ease	Animation style to be used. See the chapter 'Animations' in the emWin manual for reference.
Period	Period in ms how long the animation will last.
Disclose	If disclose mode should be used.

Job-specific parameters passed to slot-routine

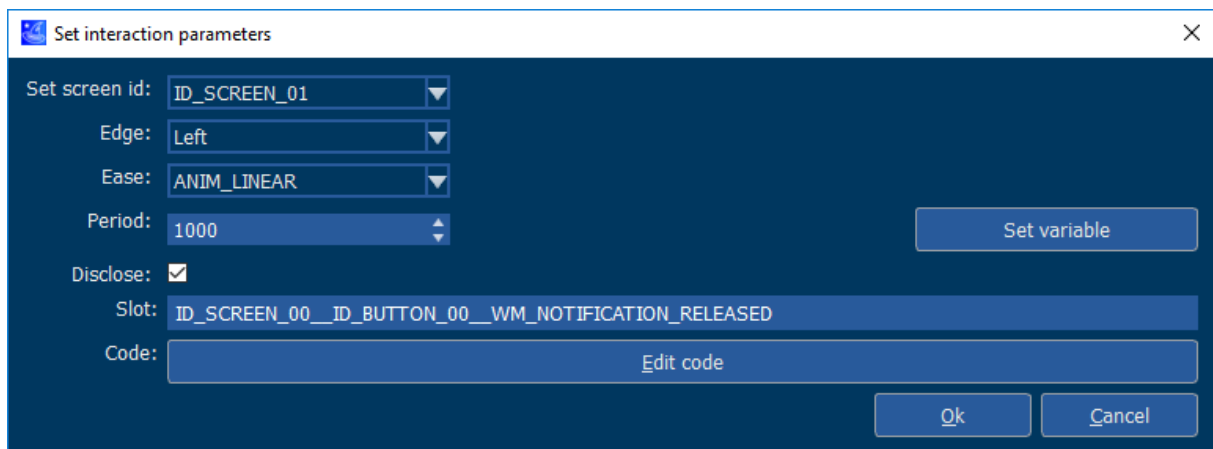
Parameter	Description
<code>aPara[0].v</code>	Screen Id.
<code>aPara[1].v</code>	Index of edge.
<code>aPara[2].pFunc</code>	Pointer to ease function.
<code>aPara[3].v</code>	Animation period.
<code>aPara[4].v</code>	If 1, disclose mode is used.

Additional information

Note that screens which are not being marked as persistent (see explanation in sub-chapter *Screen* on page 154) will be deleted after they have been shifted out of the display area.

Please note that this job requires a motion partner for the screen to work. A motion partner can be defined in the property *Motion partner* on page 106.

Example



9.4.41 SHIFTWINDOW

Description

Shifts a window in with a user-defined animation. This job is similar to SHIFTSCREEN.

Interaction parameters of dialog

Parameter	Description
<code>Window ID</code>	ID of the window to be shifted in.
<code>Edge</code>	Edge of the screen the window should be moved to.
<code>Ease</code>	Animation style to be used. See the chapter 'Animations' in the emWin manual for reference.
<code>Period</code>	Period in ms how long the animation will last.
<code>Disclose</code>	If disclose mode should be used.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Window Id.
<code>aPara[1].v</code>	Index of edge.
<code>aPara[2].pFunc</code>	Pointer to ease function.
<code>aPara[3].v</code>	Animation period.
<code>aPara[4].v</code>	If 1, disclose mode is used.

Additional information

Please note that this job requires a motion partner for the window to work. A motion partner can be defined in the property *Motion partner* on page 106.

9.4.42 SHOWSCREEN

Description

This job makes the given screen instantly visible. There are no animation options for this job.

Receiving objects

- *Screen* object

Interaction parameters of dialog

Parameter	Description
<code>Screen ID</code>	ID of the screen to be shown.

Job-specific parameters passed to slot-routine

Parameter	Description
<code>aPara[0].v</code>	Screen Id.

Additional information

After the new screen is visible the AppWizard tries to check which screen is 'behind' the new one. This is done by checking which screen is visible at the midpoint 'behind' the new screen. If there is a screen which is not marked as persistent (see explanation in sub-chapter *Screen* on page 154) it will be deleted.

9.4.43 START

Description

This job starts a given timer with the set period.

Receiving objects

- *Timer* object

9.4.44 STOP

Description

This job stops a given timer.

Receiving objects

- *Timer* object

9.4.45 TOGGLE

Description

Toggles the 'pressed' state of the given object. For example, when executing this job on a Switch it will toggle between its left and right state and when executing on a Button, it will toggle between its pressed and unpressed state.

Receiving objects

- *Button* object
- *Switch* object

9.4.46 NULL

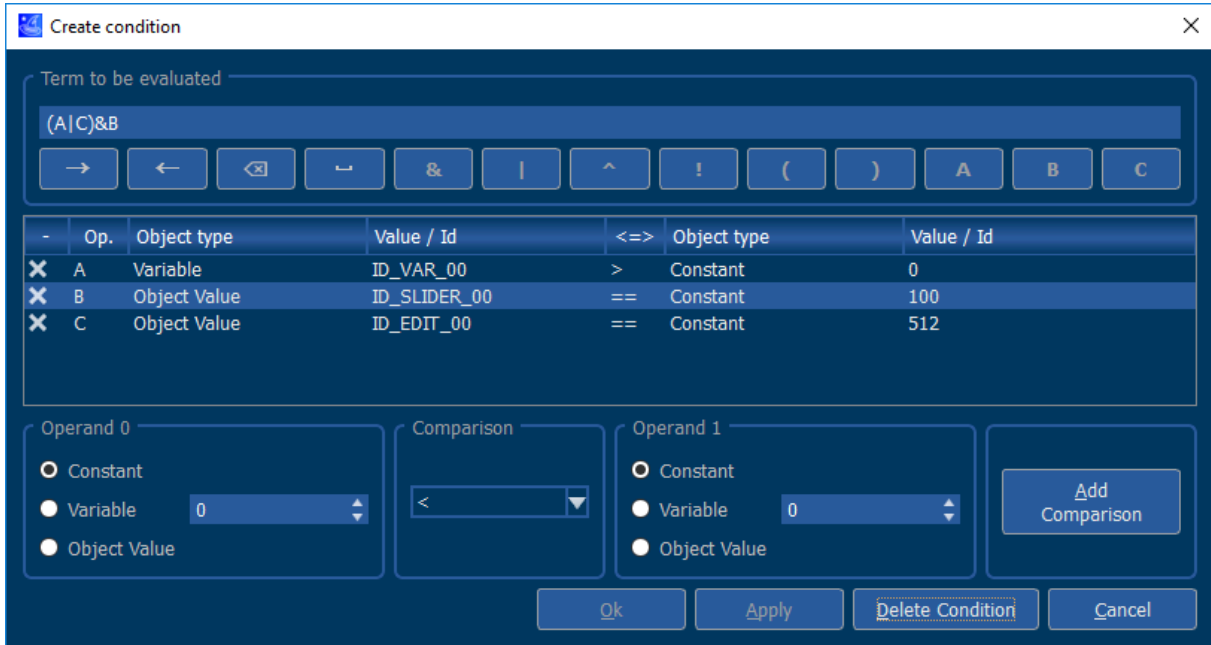
Description

Specifying a job to `NULL` gives the user the option to simply add custom code to the interaction and do nothing else.

9.5 Conditions

9.5.1 Introduction

A condition can be optionally added to an interaction. When a condition is added, the job of the interaction will only be executed, if the term of the condition is true. This allows the user to even add complex logic to the application.



Adding a condition

+/-	Edit	[-]	Emitter	Signal	Job	Receiver
X	/	+	ID_BUTTON_00	RELEASED	SETVALUE	ID_VAR_00
X	/	/	ID_SLIDER_00	VALUE_CHANGED	SETCOLOR	ID_BOX_00

As hinted before, a condition can only be added to an existing interaction. A condition can be added by clicking the plus symbol in the condition column in the interaction window.

Editing or deleting a condition

A condition can be edited or removed from an interaction by clicking on the pen icon in the condition column in the interaction window.

9.5.2 Terms and operands

A term is made up from operands (such as A, B, C, ...) and logical operators.

Comparisons and operands

Each operand is a validation of a comparison between two values. The values to be compared can be:

- **constants**,
- **variables** and
- **objects** (meaning objects that have a value, such as sliders, gauges, edits in decimal mode, ...).

The operators for comparison are:

- < (less than)
- <= (less than or equal)
- = (equal)
- >= (greater than or equal)
- > (greater than)
- != (not equal)

-	Op.	Object type	Value / Id	<=>	Object type	Value / Id
X	A	Object	ID_SLIDER_00	<	Constant	100

Operand 0

Constant

Variable

Object

Comparison

Operand 1

Constant

Variable

Object

Note

For each comparison, an operand is added. The operand is named by a letter of the alphabet, starting with A. This means, the maximum number of operands to be added for a condition is limited to 26.

Term

When the operands have been added, a term can be set up. The term consists of the added operands and logical operators. the logical operators that can be used are:

- & (AND)
- | (OR)
- ^ (XOR)
- ! (NOT)

Furthermore, brackets (and) can be used.

Term to be evaluated

→
←
✕
&
|
^
!
(
)
A
B
C

-	Op.	Object type	Value / Id	<=>	Object type	Value / Id
X	A	Variable	ID_VAR_00	>	Constant	0
X	B	Object	ID_SLIDER_00	==	Constant	100
X	C	Object	ID_EDIT_00	==	Constant	512

The interface allows the user to enter the term using the buttons or manually enter it via the keyboard. For each operand, an individual button is added.

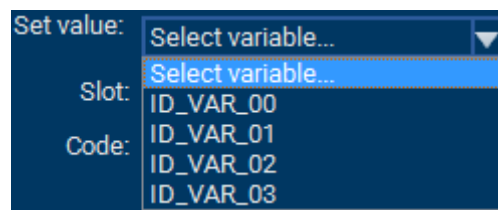
The four left-most buttons are used for moving the cursor, deleting characters and inserting spaces. Buttons with operators or operands that may not be inserted at the current position appear grayed out.

When using the keyboard to enter the term, any operators or operands not currently applicable to the term are ignored.

Chapter 10

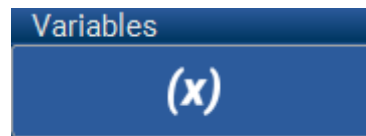
Variables

The user can add variables to the project which can be processed by the application. Variables can also be manipulated from outside of the application.



10.1 Variable management

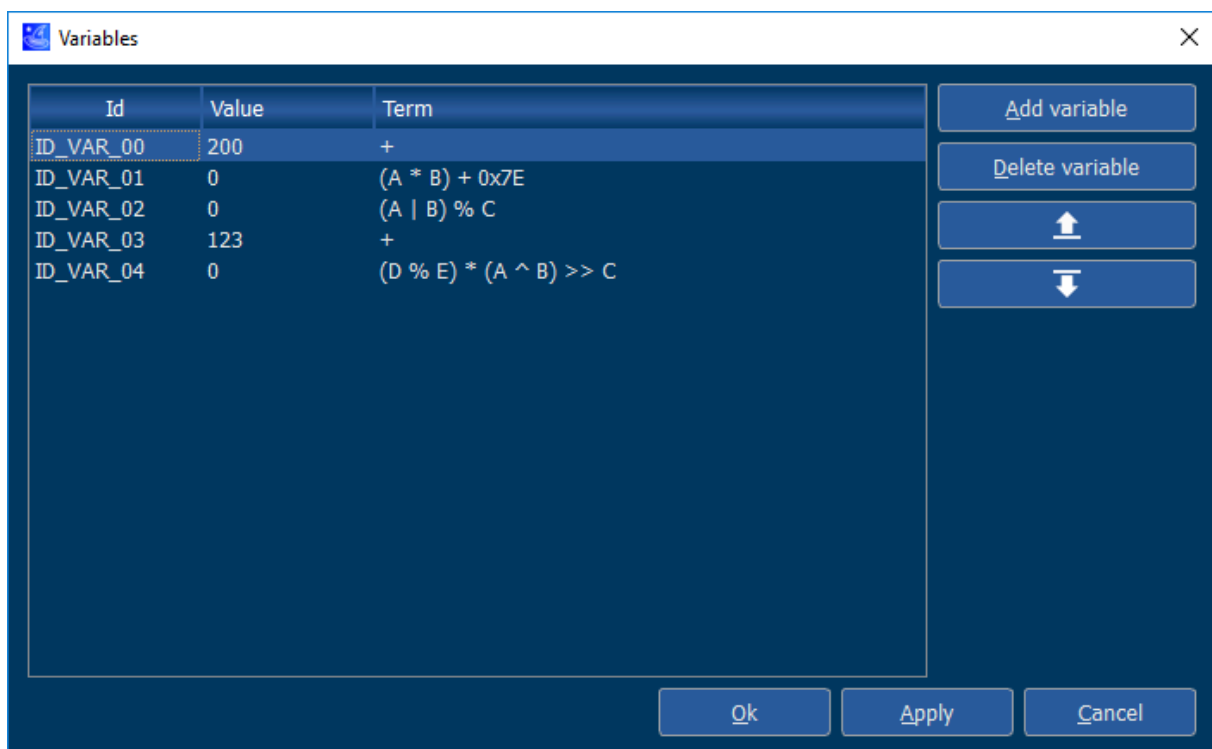
The variable window allows the user to manage the variables for the current project. The management dialog can be opened by clicking the “Variables” button in the bottom left corner of AppWizard.



New variables can be added by pressing the **Add variable** button and they can be deleted by pressing the **Delete variable** button.

Using the buttons with the upwards and downwards arrows will move the selected variable either up or down, depending on the button.

After a variable has been created, it may be used for an interaction or can be manipulated from user code using `APPW_SetValue()`.



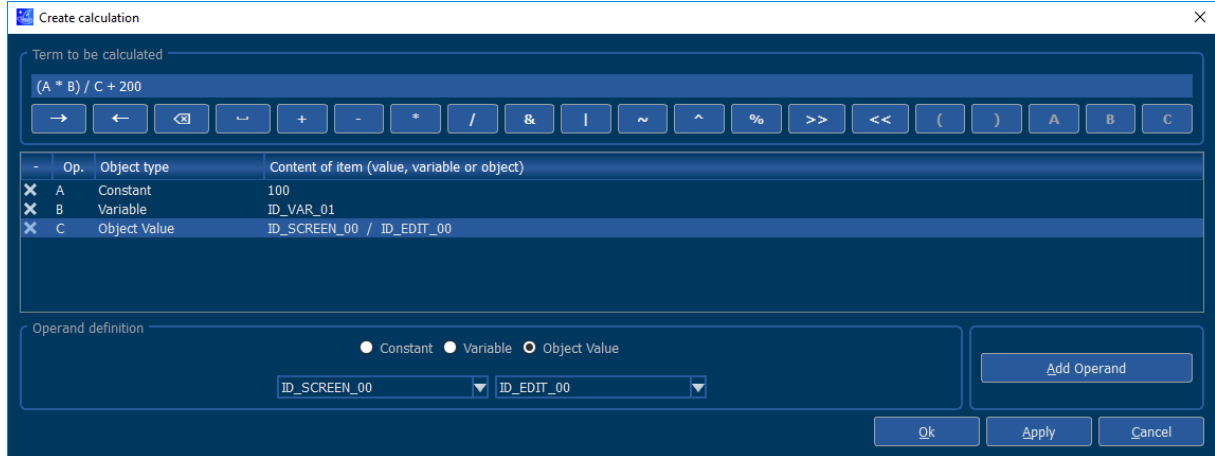
Initial value

The value in the “Value” column of a variable can be edited. This value will be assigned to the variable upon start of the application.

10.2 Calculations

Introduction

By adding a term to a variable, the value of that variable will be calculated using the given term. A term can be calculated from other variable values, object values or constant values. This allows for a much more detailed application logic inside AppWizard projects.



Adding a calculation

To add a term to a variable, click the "+" in the "Term" column of the variable management dialog.



10.2.1 Terms and operands

A term is made up from operands (such as A, B, C, ...) and operators.

10.2.1.1 Operands

To create a term, operands have to be added in the first place. Operands are values that can be derived from:

- Object Value
- Variable
- Object Geometry
- Screen Geometry
- Object Property
- Fixed Value

Object Value

They can be used from objects which have a 'value' available:

- *Button* object

Returns 1 for pressed and 0 for unpressed state.

- *Dropdown* object

Returns the index of the currently selected item.

- *Text* object
- *Edit* object

If object is in decimal mode the current value is returned.

- *Progbar* object

- *Rotary* object
- *Slider* object
- *Gauge* object

Returns the current value.

- *Listbox* object
- *Wheel* object

Returns the index of the currently selected item.

- *Listview* object

Returns the index of the currently selected row.

- *Movie* object

Returns the current frame index.

- *Switch* object

Returns 1 for on and 0 for off.

Variable

Means obviously the current value of the variable.

Object Geometry

The following properties can be used:

- X0
- Y0
- X1
- Y1
- XSize
- YSize

Screen Geometry

The following properties can be used:

- X0
- Y0
- X1
- Y1
- XSize
- YSize

Object Property

Currently only a few objects support properties:

- Listview - Number of columns and number of rows.
- Dropdown - Number of items.
- Listbox - Number of items.
- Movie - Number of frames and current frame.

Fixed Value

Simply a constant value.

Offset

At the end an offset can be added to the operand.

10.2.1.2 Operators

The following operators can be used for a calculation between the operands:

- + (add)

- - (subtract)
- * (multiply)
- / (divide)
- % (modulo)
- & (binary AND)
- | (binary OR)
- ^ (binary XOR)
- ~ (binary one's complement)
- << (binary left-shift)
- >> (binary right-shift)

10.2.1.3 Creating a term

Note

Multiplication and division **do not** have a higher precedence level than addition and subtraction! All operations are calculated from left to right, except when brackets are used. Brackets are mandatory to indicate a higher level of precedence.

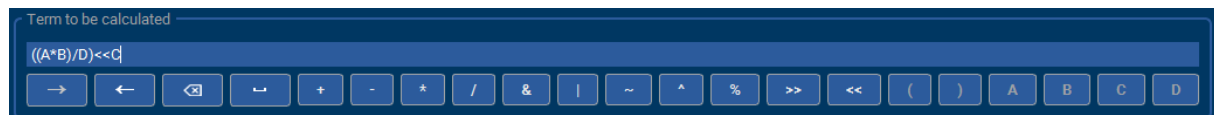
Example:

- $1 + 2 * 3 = 9$
- $1 + (2 * 3) = 7$

When all necessary operands have been added, a calculation between the operators can be set up. Each operand is equal to a letter (such as A, B, C, ...) and the operands are to be used within the term to be calculated.

Note

Numbers can also be used in the term. It is **not** mandatory to create an operand for a constant! Decimals are allowed, as well as hexadecimal numbers (prefixed by **0x**).



10.2.1.4 Calculating a variable

In order to calculate the new value of a variable with the term, the job **CALC** has to be executed with the desired variable as the receiver.

If this causes a change of the value, the variable will emit a **VALUE_CHANGED** signal.

10.3 Manipulating variables from user code

The routines `APPW_SetVarData()` and `APPW_GetVarData()` allow for reading and modifying variables from a project's user code. In combination with the signal `VALUE_CHANGED`, this feature can be utilized for various use cases.

Example



For example, in a weather forecast application, the temperature values can be stored in variables. When the user presses a button to refresh the temperature data, new data is polled and set to the variable using `APPW_SetValue()`.

By reacting on `VALUE_CHANGED`, the application would know when a temperature value has changed and if e.g. a different text or bitmap should be displayed.

Chapter 11

Animations

AppWizard allows the user to add complex animations to their project.

With AppWizard V1.20, animations have been completely reworked. This has been done because the previous process of defining complex animations was too complicated and not intuitive enough. Animations now support IDs which simplifies the use of animations within AppWizard and the ID makes them addressable.

With this animation rework, a couple of jobs and signals in AppWizard have been marked as obsolete. We recommend to not use these old mechanisms and to rework existing projects, eventually. Projects from previous versions with old animation interactions are still fully functional though.

Note

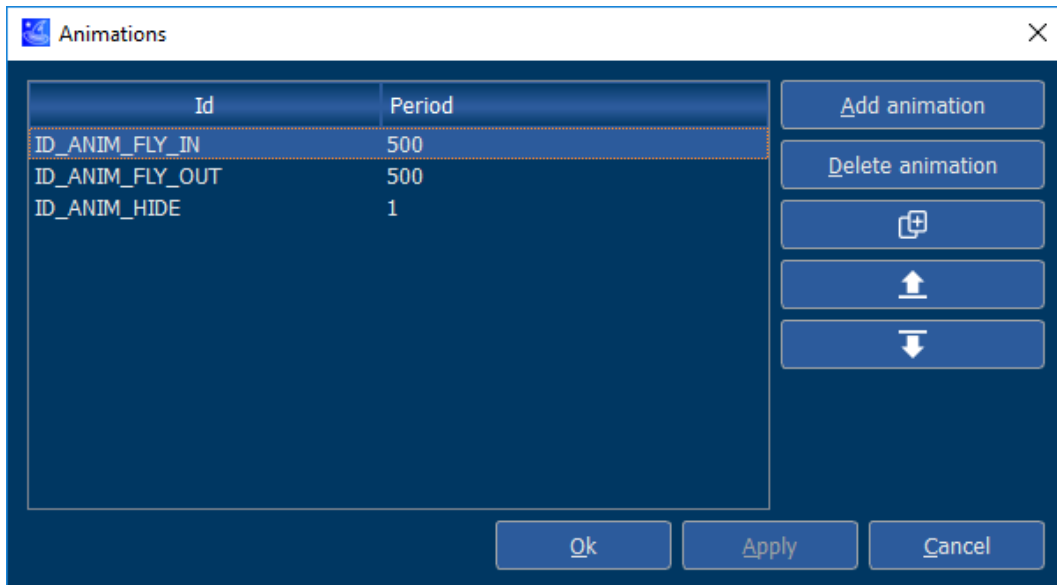
More information about the basics of emWin animations (such as animation items, animation ease, etc.) can be read in the document **UM03001 emWin User Guide & Reference Manual**.

11.1 Pre-defining animation IDs

The first step to adding animations to an AppWizard project is opening the animation dialog by clicking the icon in the lower left corner of the tool.

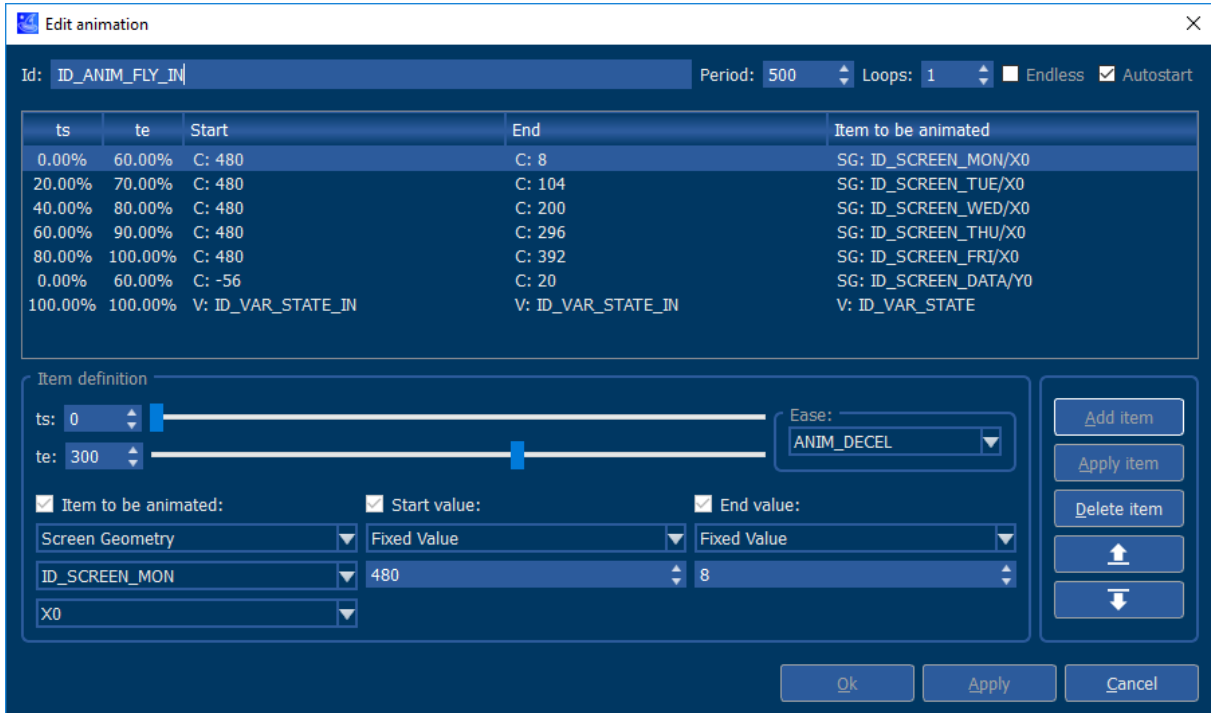


Now, the desired amount of animations can be added by clicking the "Add animation" button. The table shows the ID of the animation and its entire duration in milliseconds.



11.2 Edit animations

To define and edit an animation, click its ID in the animation dialog. Then, the edit dialog will open.



11.2.1 Animation properties



- **Autostart:** Automatically starts the animation after executing the job ANIMCREATE.
- **Period:** Animation duration in ms. Takes effect only after creation with the autostart option.
- **Endless/Loops:** The animation runs endlessly until stopped or for a number of loops. Takes effect only after creation with the autostart option.

11.2.2 Start and end time of animation items

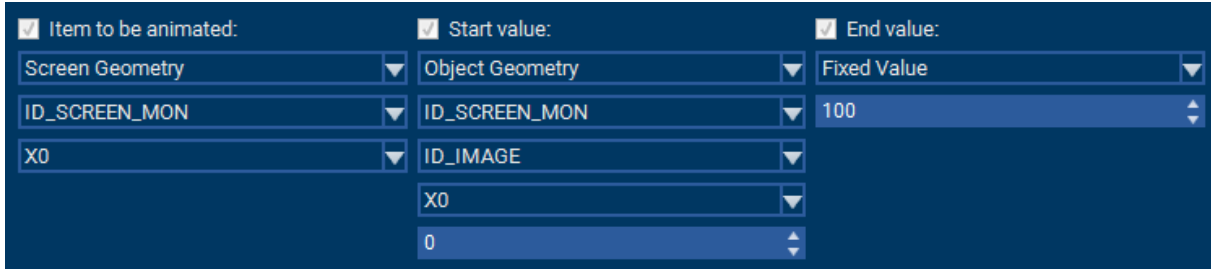


- **ts:** Start time of item in ms within the animation
- **te:** End time of item in ms within the animation

The item list shows the start and end time of each item in percent, relative to the duration of the entire animation.

ts	te
0.00%	60.00%
20.00%	70.00%
40.00%	80.00%
60.00%	90.00%
80.00%	100.00%
0.00%	60.00%
100.00%	100.00%

11.2.3 Animation values



An animation always “animates” a certain value, such as a window position.

For an animation item, three values need to be specified:

- **Start value:** Initial value of the item to be animated.
- **End value:** Final value of the item, when the animation has ended.
- **Item to be animated:** Item, that the animated value should be applied to.

The following types of values can be used for animation items:

Type	Abbreviated	Description
Object value	OV	Value of an object, such as a slider.
Variable	V	Value of an AppWizard variable.
Object Geometry	OG	Coordinate or size of an object.
Screen Geometry	SG	Coordinate or size of a screen.
Fixed Value	C	A constant value.

Object value, object geometry and screen geometry allows an additional offset in the low-ermost field.

Animating object coordinates

Note that when animating object coordinates the animation has to match the anchor point of the object. For example, when an object has the anchor point in the top left corner (x0, y0), animating the coordinate x1 will not have an effect.

ANIMSTART and ANIMEND signals

When object coordinates are animated, they emit ANIMSTART and ANIMEND signals when the associated animation item starts or ends.

11.2.4 Animation ease

The ease function of an animation defines how the animated value will change over time and thus how the animation will look like. More information about this can be read in the emWin manual in the chapter *Animations*.

Ease	Description
ANIM_LINEAR	Animated is performed linear.
ANIM_ACCEL	Animation is accelerating.
ANIM_DECEL	Animation is decelerating.
ANIM_ACCELDECEL	Animation is accelerating, then decelerating.

11.3 Running animations

Starting an animation

Once an animation has been defined, it first has to be created using the interaction job ANIMCREATE.

Then, the animation can be started using the job ANIMSTART.

Animations have unique IDs only one animation with a specific ID can be started and created at a time. For example, only one animation with `ANIM_ID_000` can be created. A second attempt to create or start the animation with `ANIM_ID_000` will not be ignored.

Note

Animations are only deleted automatically after they have run completely. If you want to stop and delete them by interaction, you have to activate the remove option in ANIMSTOP.

Stopping an animation

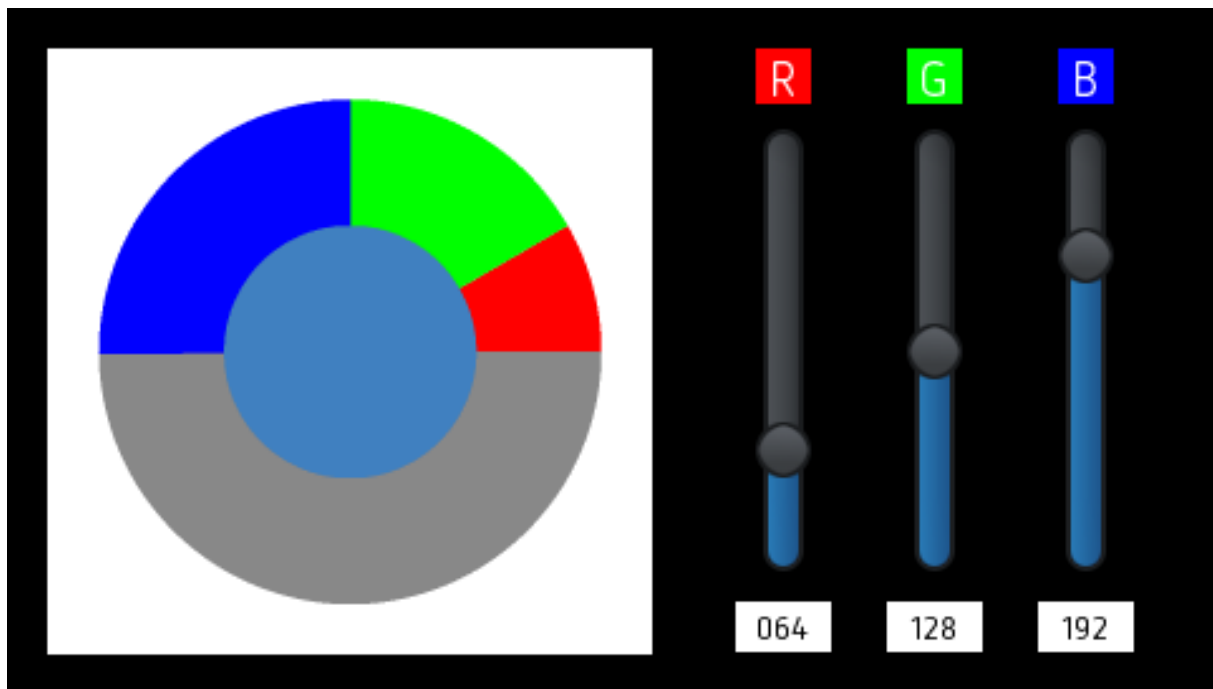
To stop an animation, the job ANIMSTOP can be used.

Optionally, the job can also remove an animation. To use it again, it would then have to be created again using ANIMCREATE.

Chapter 12

Drawings

AppWizard allows the user to add custom drawings to their application. The feature supports a wide array of emWin's drawing routines.



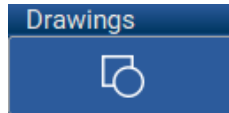
Note

This chapter will only provide information on how to use drawing routines within AppWizard. To learn more detail about a specific routine, please refer to the document **UM03001 emWin User Guide & Reference Manual**.

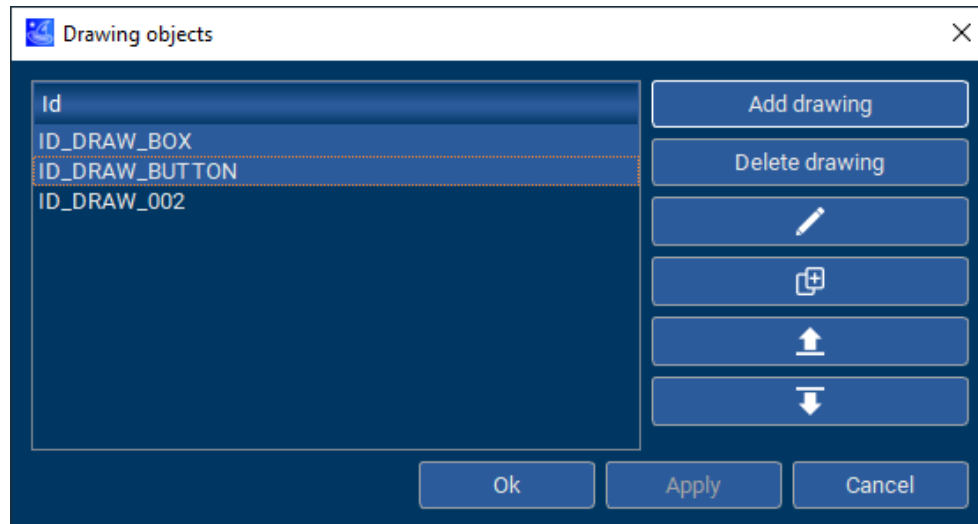
12.1 Creating a drawing object

Creating a drawing object

Before a drawing can be displayed, the first step is to create a drawing object. To do this, open the drawings dialog by clicking the quick access button in the lower left corner of AppWizard.



The dialog allows adding, editing and deleting drawing objects.



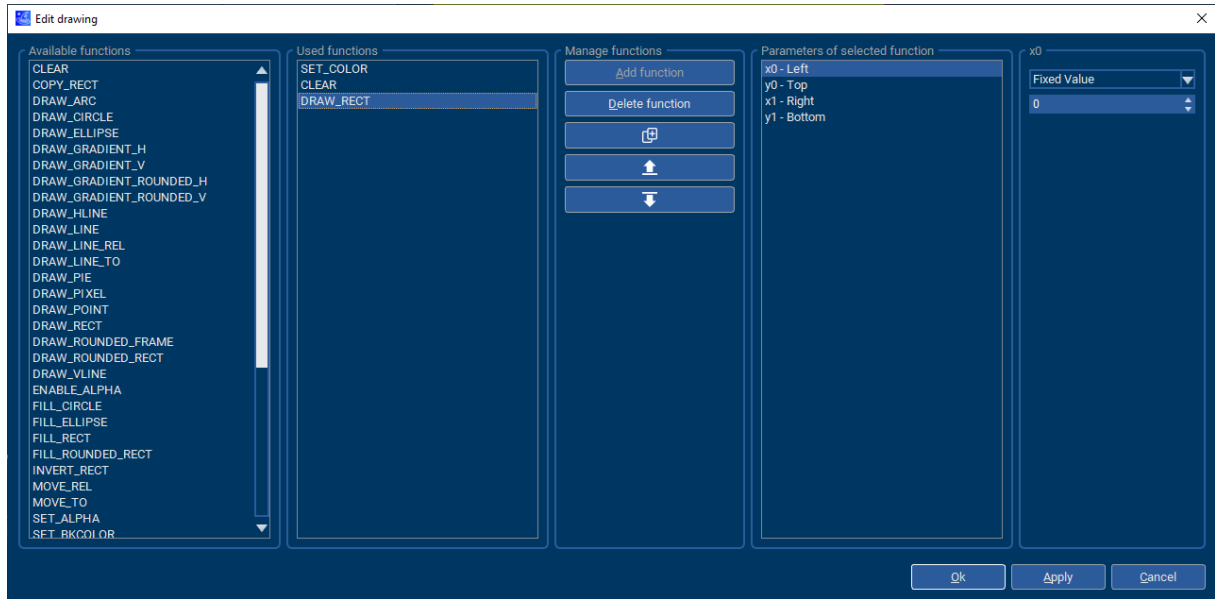
After an object has been created, it can be edited by clicking the 'Edit' icon.



12.2 Defining a drawing

Editing a drawing

To edit a drawing, select a drawing object in the above mentioned dialog and click the 'Edit' icon. A dialog will open, allowing to define what the drawing object should do.



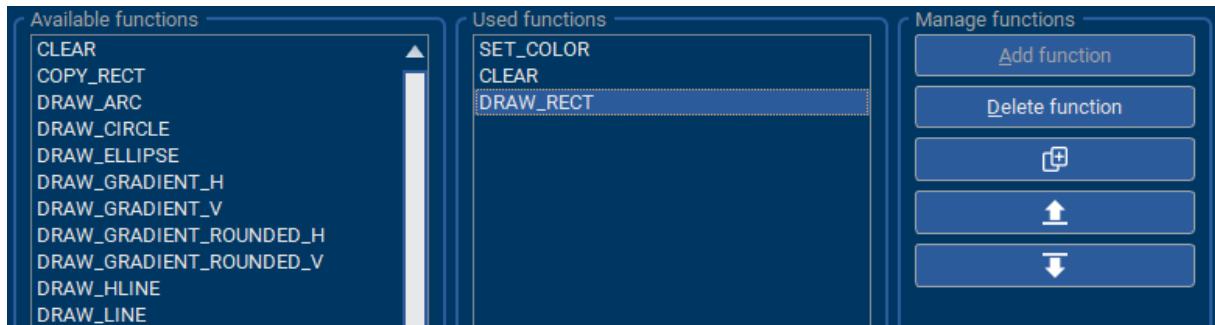
Adding functions

Note

A detailed list of all available functions and parameters can be found under Available drawing functions.

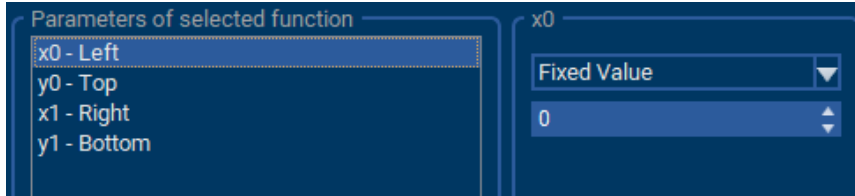
The left side of the dialog allows selecting the drawing operations that should be performed and adding them to the object. To do this, select a function under 'Available functions' and add it using the 'Add functions' button.

The function will then appear in the 'Used functions' section. This shows all the drawing operations that will be performed chronologically.



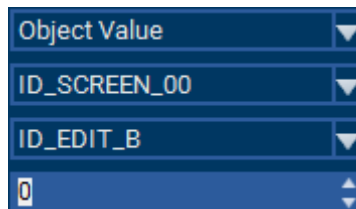
Setting the parameters

Some functions require parameters, such as coordinates for `DRAW_RECT`. To set the parameters of a function that has been added to a drawing object, select it under 'Used functions' and its corresponding parameters will be shown under 'Parameters of selected function'.



A parameter value can either be one of the following:

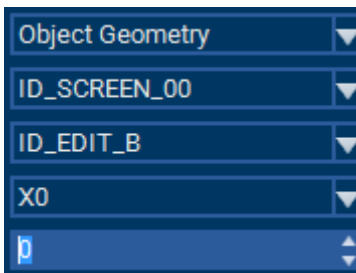
- an **object value**



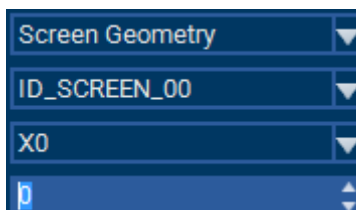
- a **variable**



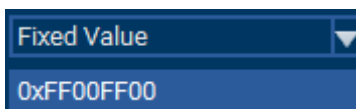
- **object geometry** (such as x0, y0, x1, y1, xSize, ySize)



- **screen geometry** (such as x0, y0, x1, y1, xSize, ySize)



- a **fixed value**



Object value, object geometry and screen geometry allows an additional offset in the low-ermost field.

12.3 Displaying a drawing

A drawing object can be displayed within any given widget of the application. It can either be drawn **before** or **after** `WM_PAINT` is executed for a given widget.

When a widget is selected, its **Predraw** and its **Postdraw** property can be set to display a given drawing object.

Predraw: ID_DRAW_BOX ▼ Postdraw: ID_DRAW_002 ▼

12.4 Available drawing functions

The following table lists all available drawing functions. The parameters of the functions are largely identical to the parameters of the analogously existing emWin functions described in the emWin manual. In order to avoid redundancy, we have therefore dispensed with a detailed documentation of the drawing functions at this point.

Function	Description
CLEAR	Clears the area using the background color.
COPY_RECT	Copies a rectangular area to a new position.
DRAW_ARC	Draws an arc.
DRAW_CIRCLE	Draws a circle.
DRAW_ELLIPSE	Draws an ellipse.
DRAW_GRADIENT_H	Draws a horizontal gradient.
DRAW_GRADIENT_V	Draws a vertical gradient.
DRAW_GRADIENT_ROUNDED_H	Draws a horizontal gradient with rounded corners.
DRAW_GRADIENT_ROUNDED_V	Draws a vertical gradient with rounded corners.
DRAW_HLINE	Draws a horizontal line.
DRAW_LINE	Draws a line.
DRAW_LINE_REL	Draws a relative line from the current position.
DRAW_LINE_TO	Draws a line from the current to the new position.
DRAW_PIE	Draws a circle section.
DRAW_PIXEL	Sets a single pixel.
DRAW_POINT	Draws a point.
DRAW_RECT	Draws a rectangle.
DRAW_ROUNDED_FRAME	Draws a rounded rectangle in the given rectangle.
DRAW_ROUNDED_RECT	Draws a rounded rectangle.
DRAW_VLINE	Draws a vertical line.
ENABLE_ALPHA	Enables or disables alpha drawing mode.
FILL_CIRCLE	Fills a circle.
FILL_ELLIPSE	Fills an ellipse.
FILL_RECT	Fills a rectangle.
FILL_ROUNDED_RECT	Fills a rectangle with rounded corners.
INVERT_RECT	Inverts the color of a rectangle area.
MOVE_REL	Relative move to a new position.
MOVE_TO	Move to screen coordinates.
SET_BKCOLOR	Sets the background color.
SET_COLOR	Sets the foreground color.
SET_PENSIZE	Sets the pen size.
SET_LINESTYLE	Sets the linestyle.
AA_DISABLE_HIRES	Disables high-resolution anti-aliasing.
AA_DRAW_ARC	Draws an anti-aliased arc.
AA_DRAW_CIRCLE	Draws an anti-aliased circle.
AA_DRAW_LINE	Draws an anti-aliased line.
AA_DRAW_PIE	Draws an anti-aliased pie.

Function	Description
AA_DRAW_ROUNDED_FRAME	Draws an anti-aliased rounded rectangle in the given rectangle.
AA_ENABLE_HIRES	Enables high-resolution anti-aliasing.
AA_FILL_CIRCLE	Fills an anti-aliased circle.
AA_FILL_ELLIPSE	Fills an anti-aliased ellipse.
AA_FILL_ROUNDED_RECT	Fills an anti-aliased rounded rectangle.
AA_SET_FACTOR	Sets the anti-aliasing factor.

Chapter 13

User Code

The following chapter explains how the user may add custom code to their AppWizard application. It will also be explained how variables and fonts created within AppWizard may be utilized for custom code and how slot routines can be used.

13.1 Slot routines

Slot routines are the routines that are executed with the job of an interaction.

Where to find a slot routine

The name of a slot routine can be accessed and changed in the 'Set interaction parameters' dialog. This routine is located in the file <ScreenID>_Slots.c in the directory \Custom-Code\Config\.

Prototype

```
void <ScrID>__<EmitID>__<SignID>__<RecvID>__<JobID>(APPW_ACTION_ITEM * pAction,
                                                    WM_HWIN          hScreen,
                                                    WM_MESSAGE       * pMsg,
                                                    int              * pResult);
```

- ScrID** Id of the screen where the objects are on.
- EmitID** Id of the emitting object.
- SignID** Id of the signal.
- RecvID** Id of the receiving object.
- JobID** Id of the job.

Parameters

Parameter	Description
<code>pAction</code>	Pointer to an APPW_ACTION_ITEM structure.
<code>hScreen</code>	Handle of the screen.
<code>pMsg</code>	Pointer to a WM_MESSAGE structure. <code>pMsg->hWin</code> is the handle to the receiver while <code>pMsg->hWinSrc</code> is the handle to the emitter.
<code>pResult</code>	Pointer to an int containing the 'result' value. This value is explained below.

Additional information

Each interaction has job-specific parameters. The parameters can be accessed via the `aPara` element of the APPW_ACTION_ITEM structure which is passed to a slot routine.

The parameter of each interaction is explained under **Job-specific parameters passed to slot-routine** for each job under *List of jobs* on page 190.

The parameter `pResult` points to an integer which by default is 0. If `*pResult = 0`, the interaction will be executed by the AppWizard. If `*pResult = 1`, only the custom code is executed.

13.1.1 APPW_ACTION_ITEM

Description

This structure is passed to an interaction slot routine.

Type definition

```
typedef struct {
    int          IdSrc;
    int          NCode;
    int          IdDst;
    int          IdJob;
    void         (* pfSlot)(APPW_ACTION_ITEM * pAction,
                          WM_HWIN          hScreen,
                          WM_MESSAGE       * pMsg,
                          int              * pResult);
    APPW_PARA_ITEM aPara[6];
} APPW_ACTION_ITEM;
```

Structure members

Member	Description
<code>IdSrc</code>	Id of the emitter.
<code>NCode</code>	Id of the signal.
<code>IdDst</code>	Id of the receiver.
<code>IdJob</code>	Id of the job.
<code>pfSlot</code>	Function pointer to a slot routine. Prototype explained under Slot routines.
<code>aPara</code>	Optional job specific parameters. The parameter for each job is explained in the <i>List of jobs</i> on page 190.

13.1.2 Custom user code

Slot routines

As mentioned earlier, the user may add their own code to slot routines, either via the “Edit code” dialog in the interaction dialog, or even from any editor or IDE.



Any user code within the generated slot routines stays persistent when for example exporting the AppWizard project another time.

Custom routines

If the user wants to add their own custom routines to the application, they should create a new C file and add it to their project.

AppWizard also adds the automatically generated files `Application.c` and `Application.h` to the simulation project. These files are intended to be used for user code.

Custom routines can also be added to a generated slot file, however they **must** be within the user code section, otherwise they will be overwritten once the project is exported again.

```
/**/ Begin of user code area ***/  
  
static void _FooBar(void) {}  
  
/**/ End of user code area ***/
```

13.2 Screen callback routines

Every screen object has its own generated callback routine. This callback will be called additionally, this means it isn't a requirement and may be left empty.

Where to find a screen callback

The callback is named after the format `cb<ScreenID>`, e.g. `cbID_SCREEN_00`. A screen callback routine can be found in the slot routine file, located in the project directory under `\Source\CustomCode`.

How to use them

Generally, a screen callback is very similar to an `emWin` window callback. This means, the callback may react on all types of window messages. To learn more about the different types of window messages, refer to the document **UM03001 emWin User Guide & Reference Manual**.

Note

However, a screen callback **must not** have a default case that calls `WM_DefaultProc()`, as a normal window callback would do.

Example

When reacting on the `WM_INIT_DIALOG` case, custom windows or widgets can be added to the application upon creation of the screen object. When creating a window/widget as a child to the screen, `WM_NOTIFY_PARENT` messages obviously get sent to the parent callback.

```

/*****
*
*       cbID_SCREEN_00
*/
void cbID_SCREEN_00(WM_MESSAGE * pMsg) {
    WM_HWIN hWin;
    int      Id, NCode;

    switch (pMsg->MsgId) {
    case WM_INIT_DIALOG:
        hWin = LISTVIEW_CreateEx(10, 10, 300, 200,
            pMsg->hWin, WM_CF_SHOW, 0, GUI_ID_LISTVIEW0);
        break;
    case WM_NOTIFY_PARENT:
        Id = WM_GetId(pMsg->hWinSrc);
        NCode = pMsg->Data.v;
        switch(Id) {
        case GUI_ID_LISTVIEW0:
            switch(NCode) {
            case WM_NOTIFICATION_CLICKED:
                break;
            case WM_NOTIFICATION_RELEASED:
                break;
            case WM_NOTIFICATION_MOVED_OUT:
                break;
            case WM_NOTIFICATION_SCROLL_CHANGED:
                break;
            case WM_NOTIFICATION_SEL_CHANGED:
                break;
            }
            break;
        }
        break;
    }
}

```

13.3 General AppWizard API

Routine	Description
APPW_CalcVar()	This function can be used to calculate a variable with term.
APPW_DoJob()	This function executes the given job with the given parameters.
APPW_Exec()	Keeps the AppWizard-application alive.
APPW_GetFont()	Fills a font structure using the addressed setup structure.
APPW_GetText()	This function stores the text of an object in the given buffer.
APPW_GetValue()	This function returns the value of an object.
APPW_SetCustCallback()	Sets a function pointer for a function which is executed at the end of <code>APPW_Exec()</code> .
APPW_SetPos()	Moves the given object to the given position.
APPW_SetStreamedBitmap()	Sets a streamed bitmap located in directly accessible memory.
APPW_SetStreamedBitmapEx()	Sets a streamed bitmap located in external memory.
APPW_SetText()	This function sets a text to an object.
APPW_SetValue()	This function sets the value of an object.

13.3.1 APPW_CalcVar()

Description

This function can be used to calculate a variable with term.

Prototype

```
void APPW_CalcVar(U16 IdVar);
```

Parameters

Parameter	Description
IdVar	ID of the variable to be calculated.

13.3.2 APPW_DoJob()

Description

This function executes the given job with the given parameters.

Prototype

```
int APPW_DoJob(U16          IdScreen,
              U16          IdWidget,
              int          IdJob,
              APPW_PARA_ITEM * pPara);
```

Parameters

Parameter	Description
IdScreen	Screen id of the object.
IdWidget	Object id within the screen.
IdJob	Id of the job to be executed.
pPara	Optional job specific parameters.

Usage

The parameters [IdScreen](#) and [IdWidget](#) are used to address the object to be used. A list of available jobs can be found in the *List of jobs* on page 190. Important is the parameter [pPara](#). It points to an array of `APPW_PARA_ITEM` structures. The required size of that array depends on the job to be used. Up to 6 elements can be required. The job descriptions under the *List of jobs* on page 190 contains detailed explanations about the parameters to be used.

13.3.3 APPW_Exec()

Description

Keeps the AppWizard-application alive. Needs to be called in the superloop of the application. More information can be found under *Stay alive loop* on page 76.

Prototype

```
void APPW_Exec(void);
```

Additional information

Please note that calling this function manually is normally not required. Calling it within the user code makes only sense if the generation of the 'Stay alive loop' is suppressed.

13.3.4 APPW_GetText()

Description

This function stores the text of an object in the given buffer.

Prototype

```
int APPW_GetText(U16    IdScreen,
                U16    IdWidget,
                char *  pBuffer,
                U32    SizeOfBuffer);
```

Parameters

Parameter	Description
IdScreen	ID of the parent screen the object belongs to.
IdWidget	ID of the object the text should be retrieved from.
pBuffer	Pointer to a buffer the text gets stored in.
SizeOfBuffer	The size of the buffer pBuffer points to.

Return value

If the return value is 1 no handle to the object was found.

Additional information

This function can be used for all objects which can have a text.

Available objects

This function can be used for the following objects:

- *Button* object
- *Edit* object
- *Text* object
- *QRCode* object

13.3.5 APPW_GetValue()

Description

This function returns the value of an object.

Prototype

```
int APPW_GetValue(U16 IdScreen,  
                 U16 IdWidget,  
                 int * pError);
```

Parameters

Parameter	Description
<code>IdScreen</code>	ID of the parent screen the object belongs to.
<code>IdWidget</code>	ID of the object the value should be retrieved from.
<code>pError</code>	Out pointer being used to indicate that something went wrong.

Return value

The current value of the given object.

Additional information

This function can be used for all objects which can have a value. If `pError` is 1 no handle to the object could be found.

Available objects

This function can be used for the following objects:

- *Button* object
- *Edit* object
- *Gauge* object
- *Progbar* object
- *Slider* object
- *Switch* object

13.3.6 APPW_SetCustCallback()

Description

Sets a function pointer for a function which is executed at the end of `APPW_Exec()`.

Prototype

```
void APPW_SetCustCallback(void (*pFunc)());
```

Parameters

Parameter	Description
<code>pFunc</code>	Pointer to the function which should be called.

Additional information

This function allows the user to set a function pointer which is being called from `APPW_Exec()`. This allows the user to execute his own code periodically.

Note

It is possible to set further callback and hook functions. Please refer to chapter 'Setting hook functions' in the emWin user manual [UM03001_emWin.pdf](#)

13.3.7 APPW_SetPos()

Description

Moves the given object to the given position.

Prototype

```
void APPW_SetPos(WM_HWIN hItem,  
                int      xPos,  
                int      yPos);
```

Parameters

Parameter	Description
<code>hItem</code>	Handle of object to be moved.
<code>xPos</code>	New x position in pixels relative to parent.
<code>yPos</code>	New y position in pixels relative to parent.

13.3.8 APPW_SetStreamedBitmap()

Description

Sets a streamed bitmap located in directly accessible memory.

Prototype

```
void APPW_SetStreamedBitmap(    U16    IdScreen,
                               U16    IdWidget,
                               int     Index,
                               const void * pData);
```

Parameters

Parameter	Description
IdScreen	Id of the parent screen.
IdWidget	Id of the object.
Index	Index the bitmap is used for.
pData	Pointer to the streamed bitmap.

Additional information

[pData](#) has to remain valid as long as the bitmap is in use.

`APPW_SetStreamedBitmapEx()` should be used to set bitmaps located in external memory.

Available objects

This function can be used for the following objects:

- *Button* object
- *Image* object
- *Movie* object

13.3.9 APPW_SetStreamedBitmapEx()

Description

Sets a streamed bitmap located in external memory.

Prototype

```
void APPW_SetStreamedBitmapEx(    U16    IdScreen,
                                U16    IdWidget,
                                int     Index,
                                const char * pFileName);
```

Parameters

Parameter	Description
IdScreen	Id of the parent screen.
IdWidget	Id of the object.
Index	Index the bitmap is used for.
pFileName	Pointer to string containing the file name.

Additional information

[pFileName](#) has to remain valid as long as the bitmap is in use.

`APPW_SetStreamedBitmap()` should be used to set bitmaps located in directly accessible memory.

Available objects

This function can be used for the following objects:

- *Button* object
- *Image* object
- *Movie* object

13.3.10 APPW_SetText()

Description

This function sets a text to an object.

Prototype

```
int APPW_SetText(U16    IdScreen,  
                U16    IdWidget,  
                char * pText);
```

Parameters

Parameter	Description
IdScreen	ID of the parent screen the object belongs to.
IdWidget	ID of the object the text should be set to.
pText	Pointer to the text which should be set.

Return value

If the return value is 1 no handle to the object was found.

Additional information

This function can be used for all objects which can have a text.

Available objects

This function can be used for the following objects:

- *Button* object
- *Text* object
- *Edit* object
- *QRCode* object

13.3.11 APPW_SetValue()

Description

This function sets the value of an object.

Prototype

```
int APPW_SetValue(U16 IdScreen,  
                 U16 IdWidget,  
                 int Value);
```

Parameters

Parameter	Description
IdScreen	ID of the parent screen the object belongs to.
IdWidget	ID of the object the value should be retrieved from.
Value	The value to be set to the object.

Return value

If the return value is 1 no handle to the object was found.

Additional information

This function can be used for all objects which can have a value.

Available objects

This function can be used for the following objects:

- *Button* object
- *Edit* object
- *Gauge* object
- *Progbar* object
- *Slider* object
- *Switch* object

13.4 Fonts

This chapter explains how fonts created within AppWizard can be used in custom user code.

Note

The chapter *Font management* on page 60 explains how fonts can be created using AppWizard.

13.4.1 How to use fonts

As already explained earlier in this manual, fonts can be easily created with AppWizard and used as often as the user wants to within a project. The following section will demonstrate, how these fonts can be accessed within custom C code.

Requirements

In order to be able to use a font in custom C code, it must have been created within the project. The font also has to have been referenced by an object on a screen, this means the "Set font" property for an object must be set with the desired font.

How to use a font

The following example will demonstrate, how a font can be used in user code.

The font has to be created using `APPW_GetFont()`. The ID of the object that references the font has to be stated as second parameter, the ID of the screen the object is on as first parameter.

The function will then fill a `GUI_FONT` and `GUI_XBF_DATA` structure. The variables that hold the font data should be located in ROM, so the font data stays persistent.

```

/*****
 *
 *     APP_cbWin
 */
void APP_cbWin(WM_MESSAGE * pMsg) {
    static GUI_FONT      Font;
    static GUI_XBF_DATA FontData;

    switch (pMsg->MsgId) {
    case WM_CREATE:
        APPW_GetFont(ID_SCREEN_00, ID_TEXT_00, &Font, &FontData);
        break;
    case WM_PAINT:
        GUI_SetFont(&Font);
        GUI_SetTextMode(GUI_TM_TRANS);
        GUI_DispStringAt("Test", 0, 0);
        break;
    default:
        WM_DefaultProc(pMsg);
    }
}

```

With the callback above, a window can be created. Custom window or widget callbacks should be located in the `Application.c` file and can then be used in a slot routine.

```

/*****
 *
 *     cbID_SCREEN_00
 */
void cbID_SCREEN_00(WM_MESSAGE * pMsg) {
    WM_HWIN hWin;

```

```
switch (pMsg->MsgId) {  
case WM_INIT_DIALOG:  
    hWin = WM_CreateWindowAsChild(10, 10, 100, 32, pMsg->hWin,  
    WM_CF_SHOW | WM_CF_HASTRANS, APP_cbWin, 0);  
    break;  
}  
}
```

Note

To learn more about slot routines and custom user code, refer to *Slot routines* on page 258.

13.4.2 Font API

The following table provides an overview of the routines related to fonts.

Routine	Description
<code>APPW_GetFont()</code>	Fills a font structure using the addressed setup structure.

13.4.2.1 APPW_GetFont()

Description

Fills a font structure using the addressed setup structure.

Prototype

```
int APPW_GetFont(U16          IdScreen,
                U16          IdWidget,
                GUI_FONT      * pFont,
                GUI_XBF_DATA * pData);
```

Parameters

Parameter	Description
IdScreen	ID of the screen.
IdWidget	ID of the widget.
pFont	GUI_FONT structure to be filled.
pData	Pointer to a GUI_XBF_DATA structure

Return value

0 Function has succeeded.
1 Function has failed.

Example

See *How to use fonts* on page 274 for an example.

13.5 Variables

13.5.1 How to use variables

Variables in the AppWizard can be used to store a value. They can be accessed and changed by the application or from outside of the application. The application can react on a change of a variable using interactions.

Creating variables

The user can manage (add and delete) their variables via the variable resource window. This window can be accessed by clicking the lower right quick access button, located in the lower left corner of the AppWizard.

Using variables for interactions

The main purpose for variables is to use them within an interaction, whether as an emitter or as a receiver.

If the variable is an emitter of an interaction, the signal to be reacted on can be a change of that variable. If the variable is instead the receiver of a signal, the job can be to change the value of the variable.

Reading and setting variables from outside of the application

Variables created with the AppWizard can be read from outside of the application via the method `APPW_GetVarData()` and set from outside of the application via the method `APPW_SetVarData()`.

13.5.2 Variables API

The following table provides an overview of the routines related to variables.

Routine	Description
<code>APPW_GetVarData()</code>	Returns the value of a variable.
<code>APPW_SetVarData()</code>	Sets the value of a variable.

13.5.2.1 APPW_GetVarData()

Description

Returns the value of a variable.

Prototype

```
I32 APPW_GetVarData(U16 Id,  
                    int * pError);
```

Parameters

Parameter	Description
Id	ID of the variable.
pError	Pointer to integer used to return error on demand.

Return value

Data value of the specified variable.

13.5.2.2 APPW_SetVarData()

Description

Sets the value of a variable.

Prototype

```
int APPW_SetVarData(U16 Id,  
                   I32 Data);
```

Parameters

Parameter	Description
Id	ID of the variable.
Data	Data value to set.

Return value

- 0 Function has succeeded.
- 1 Function has failed.

Chapter 14

Board support packages (BSPs)

As already mentioned in the chapter *Requirements* on page 20 the AppWizard can be used with any ANSI C compiler without any additional software library. To make things easy it comes with a couple of preconfigured BSPs. The following chapter explains in detail which software components need to be included in a BSP, how to create custom BSPs and how to import a BSP into the repository of the AppWizard.

14.1 Preconfigured BSPs included in the shipment

The AppWizard comes with some ready to use preconfigured BSPs to be used with SEGGER Embedded Studio, but also with other IDEs. They contain the following:

- A ready-to-use display configuration
- A ready-to-use touch screen configuration (if a touch screen exists)
- A ready-to-use file system configuration (if SD card is accessible)
- A binary version of embOS
- A binary version of emFile (if SD card is accessible)

Why does a BSP include embOS?

embOS is only used to get the CPU initialized and to give emWin a time base. An operating system is basically not a requirement for AppWizard. But from emWin's standpoint it makes more sense to use the embOS code, instead of rewriting it.

The same applies to the time base for emWin. Instead of using embOS a time base can be achieved with a simple timer interrupt routine.

Why does a BSP include emFile?

A file system is only required if resources should be outsourced to external media, for which we use the emFile system.

14.1.1 Example

The following example will explain how to open and run a project for a supported BSP on target hardware with SEGGER Embedded Studio.

14.1.1.1 Step 1: Select BSP

Select **Project** → **Edit Options** and click **Select BSP**. Choose any BSP for Embedded Studio and click **Select**. Confirm the selection with **Ok**.




After the BSP has been selected, a new folder in the project directory named `Target` will be created. This folder contains the complete BSP.

14.1.1.2 Step 2: Generate code

Choose **File** → **Export & Save**. By doing this the project file will be saved and the code will be exported to the sub folder `Source`.

14.1.1.3 Step 3: Run SEGGER Embedded Studio Project

The BSP contains a project file for SEGGER Embedded Studio. This project file has the suffix `.emProject` and can be found in the sub folder of the selected board under `\Target\BSP`.

 Start_STM32F746_ST_STM32F746G_Discovery.emProject	2021-01-06 9:38	SEGGER Embedde...	7 KB
---	-----------------	-------------------	------

Open the `.emProject` file with SEGGER Embedded Studio.

14.1.1.4 Step 4: Compile and run on target

The SEGGER Embedded Studio projects include all of the AppWizard code automatically, meaning normally no files need to be added or changed. Simply compile the project by pressing `F7` and run the project by pressing `F5`.

14.2 Creating custom BSPs

The following example shows how to create an AppWizard BSP. To be able to create a BSP for the AppWizard, we should already have an existing project with the following components:

- A ready-to-use display driver configuration
- A ready-to-use touch input configuration
- A ready-to-use time base for emWin
- A ready-to-use hardware initialization

For the sake of simplicity, we will use an already existing evaluation project for SEGGER Embedded Studio which is available on www.segger.com. This evaluation project is intended to be used with ST's STM32F429I-Discovery board.

Note

Although the example uses SEGGER Embedded Studio for demonstration, the following steps may also be applied to other IDEs.

14.2.1 Example

The following steps show how to create a reusable BSP for AppWizard based on that project.

14.2.1.1 Step 1: Create a project with AppWizard

After taking a look to the display configuration file of the above mentioned project, we know the display size and color conversion. Select **File → New project** and enter the following data:

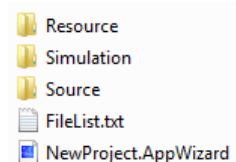
Property	STM32F429I-Discovery	Description
xSize	240	Target display x-size.
ySize	320	Target display y-size.
Color conversion	GUICC_M8888I	Desired color conversion.
Name	STM32F429I-Disco	Desired BSP name.
BSP	None	Must be left empty.
Enable Multibuffering	Yes	If the target supports multi-buffering.

14.2.1.2 Step 2: Create some elements

Fill the project with some elements such as Screen, Box and Button to make sure that there is something visible on the screen.

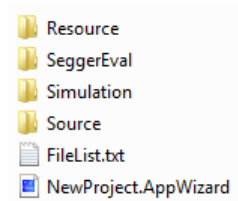
14.2.1.3 Step 3: Export & Save

Choose **File → Export & Save**. After that we should find the following directory structure in the project directory:



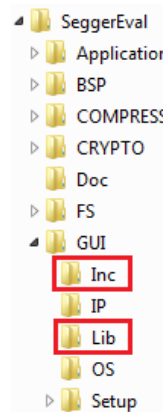
14.2.1.4 Step 4: Copy evaluation software package into project folder

By default a BSP is located in a directory named `Target` parallel to the directories `Resource`, `Simulation` and `Source`. For now, extract the evaluation software into the folder `SeggerEval`. The folder name can be any name during this step, but it **must not** be `Target`.



14.2.1.5 Step 5: Exchange libraries

The emWin libraries of the evaluation software are located in the directories `\GUI\Lib` and `\GUI\Inc`:



Note

The emWin library of the evaluation package is divided into two directories, `\GUI\Lib` and `\GUI\Inc`.

To be able to work with AppWizard, **all** files (library and header files) have to be located in a single directory which should be named `GUI_Lib`.

The directory level of the `GUI_Lib` folder does not matter, but the name is important. AppWizard will only update the libraries and header files if the folder name matches.

Deleting the old library and header files

Note

Before you delete the library and header files, check if there are any files prefixed with `BSP_*` in the `\GUI\Inc` folder. If so, make sure to not delete these files during this step.

It makes sense to move these files into the `Setup` directory. It is important to not delete these files as they are part of the BSP and not part of the emWin library.

Delete the folders `\GUI\Lib` and `\GUI\Inc` after you made sure that there are no BSP related files in these folders.

Updating the library

AppWizard comes with a couple of precompiled libraries which can be exported from within the AppWizard. To export the libraries go to "File → Export libraries..." and select a destination where the libraries should be exported to. The libraries will be exported and placed in their own directory as shown below.

- CCRX_V2
- CCRX_V3
- GCC_v7em_fpv4_hard
- GCC_v7em_fpv4_softfp
- VSWIN32

Since we are using SEGGER Embedded Studio in this example and the MCU is an ARM Cortex-M4 device, we can use the library located in `GCC_v7em_fpv4_softfp`. Copy the complete content (library and header files) from `GCC_v7em_fpv4_softfp` archive into the newly created folder `\GUI\GUI_Lib` in the evaluation project.

Using a custom-built library

If there is no precompiled library available, which may be the case when working with your own hardware, you have to create your own library instead. The emWin-documentation contains a detailed description how this can be done.

Note

The version number of emWin to be used to create the library must not be outdated and at least \geq the emWin version number of the AppWizard. Otherwise the AppWizard assumes that the library of a project using the BSP which we are creating here, needs to be updated each time we open it.

Removing duplicate headers (SEGGER evaluation projects only)

With any normal project, the step of exchanging the GUI libraries would be finished at this point. But for this example, we are using an evaluation project containing multiple SEGGER products. Because of that, the following files located in the `\GUI\GUI_Lib` directory need to be deleted, as they are already present in other project folders:

- `Global.h`
- `SEGGER.h`
- `IP_FS.h`

In case these files are not included in your project, do not delete them.

14.2.1.6 Step 6: Add file access routines

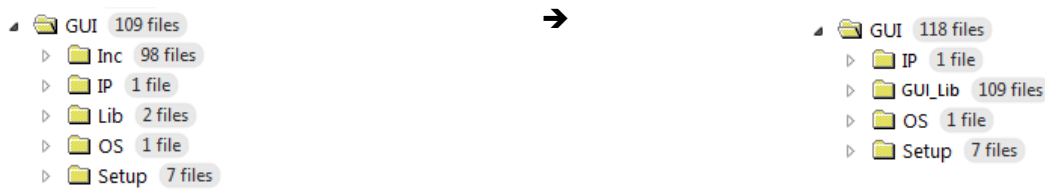
The next step is to add the file access routines to the project folder. Depending on whether you want to use a file system or not, you have to copy one of the `'APPW_X_...'` files located in the directory to the data directory. The directory can be found in `'C:\<_<Version>'`. The directory contains two `'APPW_X_...'` files:

`APPW_X_NoFS.c` to be used without a file system.
`APPW_X_emFile.c` to be used with a file system, in this case `emFile`.

Because this particular hardware does not have an SD card slot, we will add `APPW_X_NoFS.c` to the project. We copy the file over into the folder `\GUI\Setup\STM32F429_ST_ST-M32F429I_Discovery`.

14.2.1.7 Step 7: Add library to project

Now, open the project with SEGGER Embedded Studio. The project file is located under `BSP\ST\STM32F429_STM32F429I_Discovery`. Remove the directories `Lib` and `Inc` from the project and create a directory `GUI_Lib` in the project. Add the library and header files to this project folder.



Note

To remove the files, select them and press `DEL`. To add the new files, drag them from your file explorer into the correct folder in Embedded Studio.

14.2.1.8 Step 8: Add file access routines to the project

For the next step, add the file access routines to the project. The file should be located in `GUI\Setup\STM32F429_ST_STM32F429I_Discovery`.



14.2.1.9 Step 9: Adjust include files

Select the project in the Embedded Studio Project Explorer and press **ALT + ENTER** to open the project settings dialog and choose common options:



Go to the preprocessor options...

Preprocessor	
• Ignore Includes	No
• Preprocessor Definitions	STM32F429xx;_STM32F4xx_FAMILY;_STM32F429_SUBFAMILY;HSE_V/
• Preprocessor Undefinitions	
• System Include Directories	
• Undefine All Preprocessor Definitions	No
• User Include Directories	\$(ProjectDir)/../../../../../Application/GUI/SEGGERDEMO/Src;\$(ProjectD...

...and open the include directory dialog. Change `GUI\Inc` to `GUI\GUI_Lib`:

`$(ProjectDir)/../../../../../GUI/Inc` → `$(ProjectDir)/../../../../../GUI/GUI_Lib`

14.2.1.10 Step 10: Adjust data alignment

Please make sure that data alignment for const data is generally set to 4 byte alignment. In case of using SEGGER Embedded Studio that can be achieved as follows:

- Open the project.
- Open the project properties.
- Select the 'Common' configuration.
- Add `--min-ro-align=4` to additional linker options.

14.2.1.11 Step 11: Add support for run-time display rotation (optional)

If your hardware supports display rotation and you want your BSP to support run-time screen rotation through the interaction job `ROTATEDISPLAY`, you need to make some modifications in the files `LCDConf.c` and possibly `PIDConf.c`.

Note

Example configurations for this step can be viewed in most preconfigured BSPs which support run-time display rotation, like for instance the BSP `STM32F746G_Disco_ES`.

How it works

For each desired rotation to be available during run-time, emWin will create an instance of a rotated driver (a `GUI_DEVICE`). In the next step, the initialization logic of `LCD_X_Config()` is abstracted into a configuration callback. This configuration callback will be called every time a display driver instance is created.

Once the display is rotated through the job `ROTATEDISPLAY`, emWin will switch to the display driver with the desired rotation.

When this feature is used, touch input is also rotated automatically based on the selected rotation which means that the touch driver interface (`PIDConf.c`) does not need to rotate the touch input anymore.

Adapt `LCD_X_Config()` and create a configuration callback

For the first step, any initialization logic in `LCD_X_Config()` specific to the driver device needs to be moved into a new subroutine which we will call `_ConfigDriver()`.

Initialization logic specific to the driver device would be typically all function calls that have the prefix `LCD_`, e.g. `LCD_SetDevFunc()` or `LCD_SetVRAMAddrEx()`. General hardware acceleration functions that are set can be left in `LCD_X_Config()`, they only need to be executed once on start-up. These would be functions prefixed with `GUI_`, e.g. `GUI_AA_SetpfDrawCharAA4()`.

This routine will also serve as a callback for the `LCD_ROTATE` module to initialize a driver for a given orientation. The routine has to match the following prototype:

```
static void _ConfigDriver(GUI_DEVICE * pDevice, int Index, int LayerIndex);
```

Parameter	Description
<code>pDevice</code>	Pointer to <code>GUI_DEVICE</code> . May be ignored.
<code>Index</code>	Index of the rotation in the internal list. Corresponds to the order in which <code>LCD_ROTATE_AddDriverExOrientation()</code> has been called. In the below example the first call with <code>GUI_ROTATION_CCW</code> would equal <code>Index = 1</code> . <code>Index = 0</code> is the initial driver.
<code>LayerIndex</code>	Zero-based layer index.

Initialize driver and add orientations

The first driver to be initialized is the displays default orientation, which could be any of the four available orientations. In this example, it is the non-rotated 32bpp "Lin" driver, `GUIDRV_LIN_32` (see code below).

Then, the configuration callback `_ConfigDriver()` should be called once.

After that, the other desired orientations paired with their corresponding drivers get added to an internal list using the routine `LCD_ROTATE_AddDriverExOrientation()` (see code below).

Finally, the configuration callback is set for the current layer using `LCD_ROTATE_SetCallback()`.

When all these steps are done, the routine `LCD_X_Config()` should have the following structure:

```
/* *****  
*
```

```

*       LCD_X_Config
*
* Purpose:
*   Called during the initialization process in order to set up the
*   display driver configuration.
*/
void LCD_X_Config(void) {
    //
    // At first initialize use of multiple buffers on demand
    //
    #if (NUM_BUFFERS > 1)
        GUI_MULTIBUF_ConfigEx(0, NUM_BUFFERS);
    #endif
    //
    // Set display driver and color conversion for 1st layer
    //
    GUI_DEVICE_CreateAndLink(GUIDRV_LIN_32, COLOR_CONVERSION_0, 0, 0);
    //
    // Configure the initial driver
    //
    _ConfigDriver(NULL, 0, 0);
    //
    // If screen rotation is not required the following section could be switched
    // off to spare ROM
    //
    #if 1
        //
        // Create additional drivers and add them to the display rotation module
        //
        LCD_ROTATE_AddDriverExOrientation(GUIDRV_LIN_OSX_32, 0, GUI_ROTATION_CCW);
        LCD_ROTATE_AddDriverExOrientation(GUIDRV_LIN_OXY_32, 0, GUI_ROTATION_180);
        LCD_ROTATE_AddDriverExOrientation(GUIDRV_LIN_OSY_32, 0, GUI_ROTATION_CW);
        //
        // Set callback function to be used for layer 0
        //
        LCD_ROTATE_SetCallback(_ConfigDriver, 0);
    #endif
}

```

Adapt PID_X_Exec()

Finally, the PID processing routine may also need to be adapted. The routine is typically called `PID_X_Exec()` and is responsible for retrieving the touch state from the hardware and sending it to `emWin` using `GUI_TOUCH_StoreStateEx()`.

Quite often, this routine also mirrors and swaps the retrieved touch coordinates before sending it to `emWin`. But for projects supporting run-time rotation through the `LCD_ROTATE` module the routine `PID_X_Exec()` **must not** rotate the retrieved touch coordinates.

Instead, it should pass the coordinates to `emWin` “unrotated”, meaning that they match an unrotated driver like e.g. `GUIDRV_Lin_32`. The `LCD_ROTATE` module will rotate the touch input automatically based on the currently selected rotation.

Below is an example of what this routine may look like:

```

/*****
*
*       PID_X_Exec
*/
static void PID_X_Exec(void) {
    static TOUCH_STATE    TouchState;
    static GUI_PID_STATE  StatePID;
    static int            IsTouched;

    if (!_IsInitialized) {
        StatePID.Layer = _LayerIndex;
        _GetTouchState(&TouchState);
    }
}

```



```

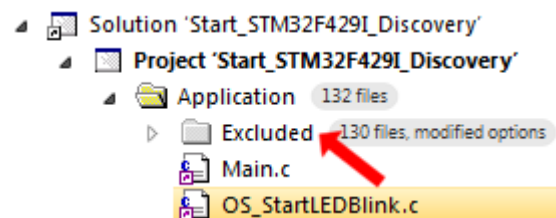
if (TouchState.NumTouch > 0) {
    IsTouched      = 1;
    StatePID.Pressed = 1;
    StatePID.x      = TouchState.x;
    StatePID.y      = TouchState.y;
    //
    // Pass information to emWin
    //
    GUI_TOUCH_StoreStateEx(&StatePID);
} else {
    if (IsTouched == 1) {
        //
        // Since StatePID is declared as static we use the x/y coordinate
        // from the down event to create an up event.
        //
        IsTouched      = 0;
        StatePID.Pressed = 0;
        GUI_TOUCH_StoreStateEx(&StatePID);
    }
}
}
}
}

```

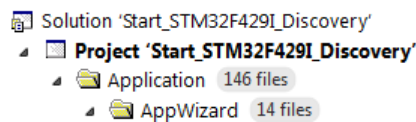
14.2.1.12 Step 12: Add application to project

Now, you should add your application to the project. If not already done, the currently selected program should be moved into the "Excluded" folder.

Then, the entire folder `Excluded` may be deleted, as the examples from the evaluation project are not required for the BSP. Make sure to leave the file `Main.c` untouched.



Right-click on the "Application" folder and select "New Folder", name the new folder *AppWizard*.

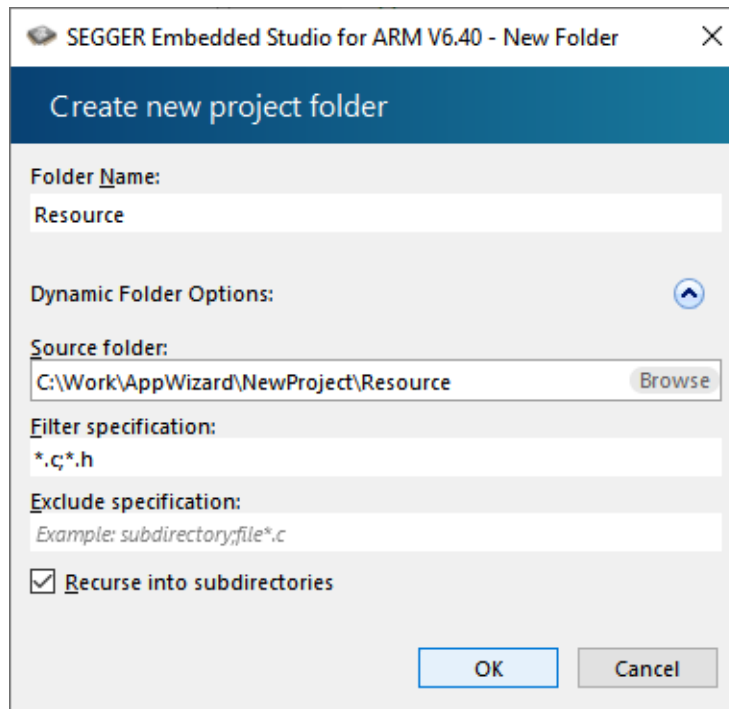


Add a resource and source directory

We now add two dynamic folders *Resource* and *Source* to the project. These folders contain the exported AppWizard application.

1. Right-click on the newly created folder *AppWizard* and select "New Folder".
2. Name the folder *Resource* and click "Dynamic Folder Options".
3. Add `*.c;*.h` to the filter specification.
4. Tick "Recurse into subdirectories".
5. Click on "Browse" to select a source folder.
6. Navigate to your AppWizard project, select the "Resource" folder and confirm with "Select Folder".
7. Click OK.

Now repeat all these steps but with the folder name *Source* and by selecting the folder "Source" from the AppWizard project.



14.2.1.13 Step 13: Compile and run on target

Now, verify that the application successfully compiles and runs on your target hardware. If the application runs without any problems, you have finished creation your own BSP.

Cleaning up the BSP folder

Finally, call **Build → Clean Solution**, to remove any unnecessary object code from your project, otherwise the object files would be part of the BSP and always copied into the `Target` folder when the project is exported. The same applies to any other files in the project that are not required, remove them from your BSP project.

Importing the BSP into AppWizard

Now, you can move on to the following tutorial called *Importing a custom BSP* on page 291 to add your new BSP to AppWizard.

14.3 Importing a custom BSP

To be able to have a custom BSP available in AppWizard's BSP repository, it has to be imported. The location of the repository folder can be found (and changed) under `Edit->Preferences->BSP stock`. To do that, select **File → Import BSP...** But before the above created BSP can be included, we have to move it into a different folder and add some further information and an image. The following steps demonstrate how this can be achieved.

14.3.1 Step 1: Create BSP folder

Create a folder somewhere with the exact name which should be shown into the BSP selection combo box. In this example, the folder is named `STM32F429I_Disco_ES`.

14.3.2 Step 2: Copy project into BSP folder

Take the folder `SeggerEval` of the above created project and copy it as a sub folder into the BSP directory and rename it to exactly the same name as its parent directory, in this case `STM32F429I_Disco_ES\STM32F429I_Disco_ES`.

14.3.3 Step 3: Add an image

When selecting a BSP in the AppWizard, an image is shown in the dialog. In this step, such an image is added to the BSP. The filename of the image must be `<Name of BSP>.jpg`, in this case the filename is `STM32F429I_Disco_ES\STM32F429I_Disco_ES.jpg`. Since the image shown in the dialog is quite small (80x80 pixels), it is recommended using a small image with dimensions of at least 80x80 pixels.

14.3.4 Step 4: Add information file

Each BSP contains a `.BSPInfo` file containing the following information:

- Display size
- Color conversion scheme
- Board name
- IDE
- MCU
- Manufacturer
- Availability of multi-buffering

Take one of the already existing `*.BSPInfo` files and copy it into the BSP folder. The file name must be of the format `<Name of BSP>.BSPInfo`. Open the file in a text editor and add the required information:

```
<!DOCTYPE emWin_AppWizard_BSP_Info>
<BSP>
  xSizeDisplay=240
  ySizeDisplay=320
  ColorConv=GUICC_M8888I
  BoardName=STM32F429I-Discovery
  IDE=Embedded Studio
  MCU=STM32F429IIT6U
  Manufacturer=STMicroelectronics
  MultibufAvail=1
  Library=GCC_v7em_fpv4_softfp
</BSP>
```

Save the file as `STM32F429I_Disco_ES\STM32F429I_Disco_ES.BSPInfo`.

MultibufAvail

If `MultibufAvail` is set to 1 (or missing entirely) the AppWizard knows that the BSP supports multi buffering. Multi buffering can then be enabled and disabled through the project options.

If the option is set to 0, the option to toggle multi buffering for a project will not be visible within AppWizard.

Library

This entry tells the AppWizard the name of the GUI-library used for the project. If this entry exists, it should be the name of the library from the repository which is required for the project. After updating the AppWizard to a newer version it checks if the library of a project needs to be updated after opening the project.

14.3.5 Step 5: Import the BSP into AppWizard

To import the BSP into AppWizard, select **File → Import BSP...** Then, select the folder `STM32F429I_Disco_ES` (the folder that contains the evaluation project, the image and the `.BSPInfo` file).

This process can take a minute or longer. After that, a new BSP should be available within AppWizard:



14.4 Using the emWin source code

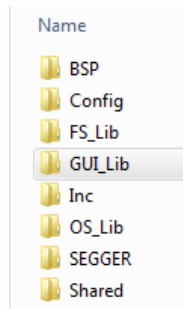
If you have purchased a emWin PRO you have access to the emWin source code. This source code can also be used within an AppWizard BSP. Either in a custom one or a BSP which comes along with the AppWizard. In general description on how to add the source code to a BSP should work for both cases.

For easiness this description is done by using the BSP for the STM32F746 Discovery which comes along with the AppWizard.

We assume that you are already familiar in using your IDE. Especially with adding new folder and setting new include paths.

14.4.1 Step 1: Remove the pre-compiled static libraries

At first delete the 'GUI_Lib' folder from the 'Target' directory within the AppWizard project.



14.4.2 Step 2: Add the source code to the project directory

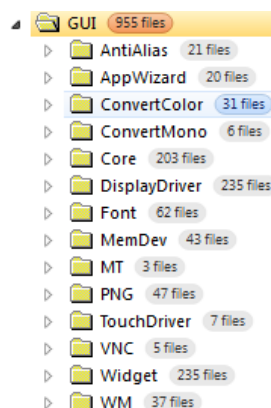
Copy the complete 'GUI' folder from your emWin shipment (found under `emWin_ship`) into the 'Target' directory. The 'GUI' folder contains the complete source code of emWin.

If you are using a BSP coming from SEGGER (either directly from the AppWizard or from our website) you should make sure that the following files in the 'GUI' directory are not present multiple times within the project. If they are delete those from the "GUI" directory.

- `Global.h`
- `SEGGER.h`
- `IP_FS.h`

14.4.3 Step 3: Add the source code to the project

Within the Embedded Studio project for the STM32F746 Discovery add a new folder and name it 'GUI'. Now add all the subdirectories from the 'GUI' directory on your hard disk drive to the newly created 'GUI' folder in your Embedded Studio project. Although, it would be possible to use a different folder structure we strongly recommend to the structure as it is. This will make it easier if you intend to update to a newer emWin version.



14.4.4 Step 4: Set include paths

After adding the source code to the Embedded Studio project you have to set the proper include paths.

Add the include paths to the following directories in the Embedded Studio project.

- GUI\AppWizard
- GUI\Core
- GUI\DisplayDriver
- GUI\Widget
- GUI\WM

Chapter 15

AppWizard SPY

AppWizard SPY is an integration of the emWin SPY tool. This feature makes it possible to monitor the memory usage of the application, as well as the window properties of the application's widgets.

AppWizard SPY also offers the possibility to record any input to the application. A recording can be run and any findings can be written into a log file.

When recording, screenshots can be taken of the application and the state of variables and objects can be written to external files.

All SPY related files of a project are located in the project's `SPY` directory.

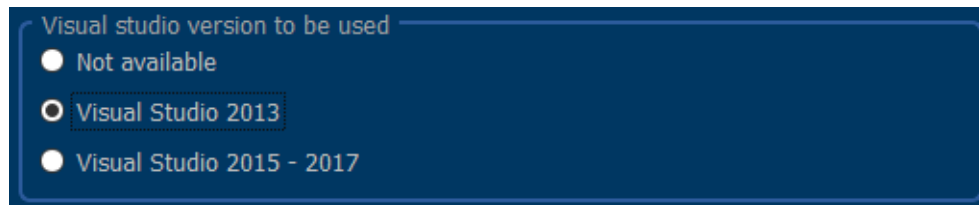
15.1 Requirements

Before using AppWizard SPY for the first time, a few requirements should be made sure of.

- The usage of AppWizard SPY requires **Microsoft Visual Studio 2013 or later** to be installed.
- The Visual Studio version must be selected in the Preferences dialog.
- The path to MSBuild.exe must be set.

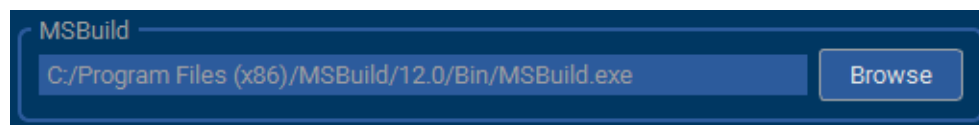
Select Visual Studio version

Open the Preferences dialog by clicking "Edit → Preferences". Then, select the version that fits to your installation.



Path to MSBuild.exe

The path to MSBuild.exe must also be set in the Preferences dialog. This is required for AppWizard to build projects and use SPY.

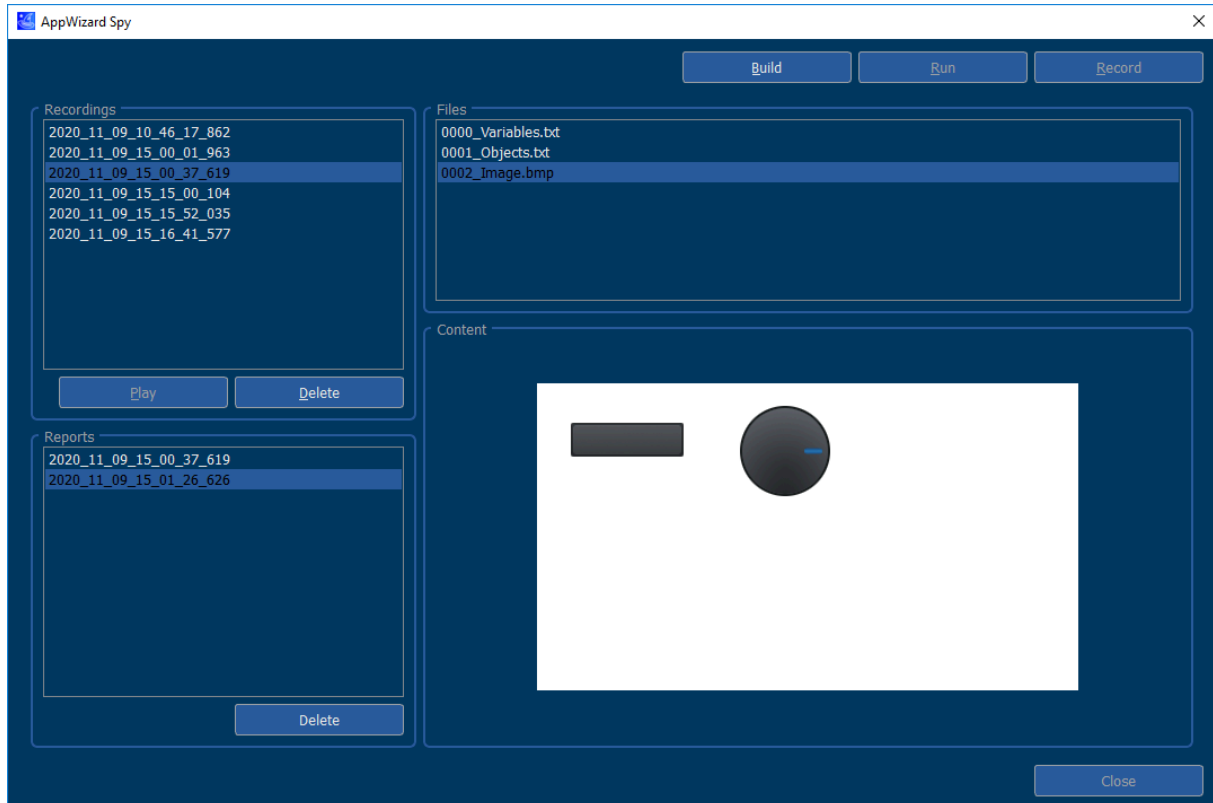


15.2 Opening the SPY dialog

The AppWizard SPY dialog can be opened by clicking **Project** → **Start Spy**. Alternatively, the **F7** key can be pressed to open the dialog.

The left side of the dialog shows previously made recordings of the project. When a recording is selected, the associated log files are shown below under **Reports**.

In case a selected report has associated files, such as screenshots, they are shown under **Files** and can be viewed under **Content**.



15.3 Building a project

Before a project can be run with AppWizard SPY, it has to be built. To do this, click the **Build** button.

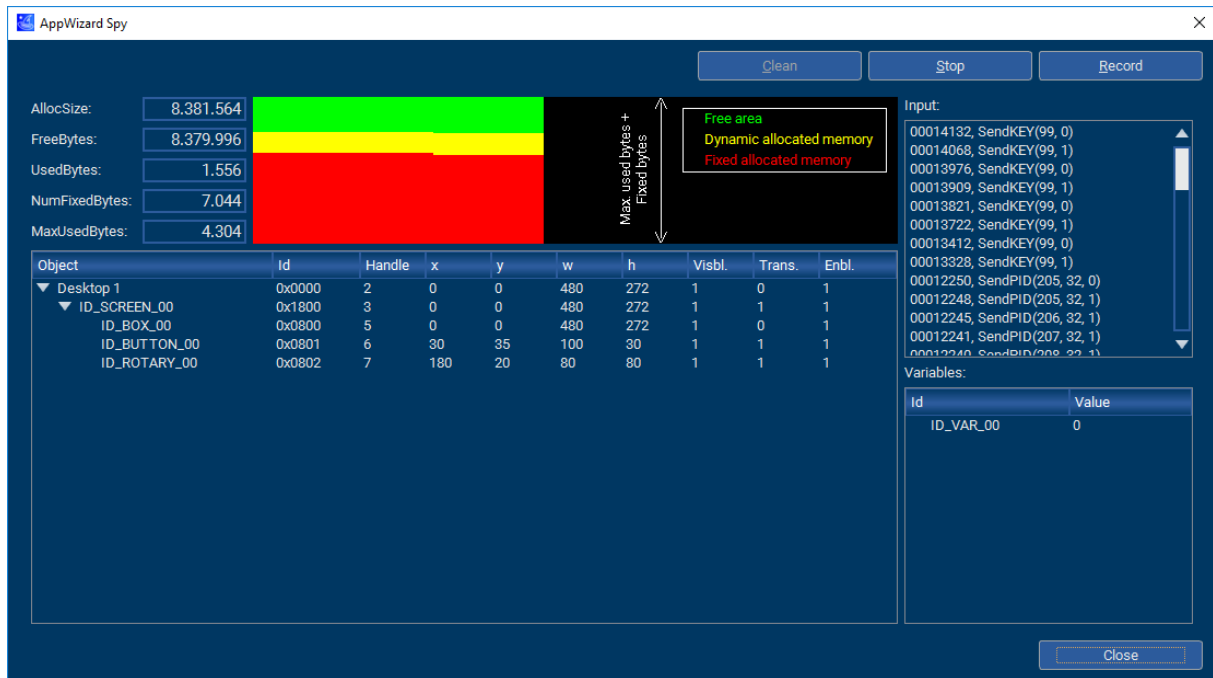
A blue rectangular button with the word "Build" in white text.

If the project should be rebuilt, the **Clean** has to be pressed first. Then, the project can be built again, as described above.

A blue rectangular button with the word "Clean" in white text.

15.4 Running a project

When the build process has finished, the **Run** button can be clicked to start the application. The built `Simulation.exe` of the project is opened and the according data is shown in the SPY window.



Allocated memory

In the upper section of the SPY dialog, the available and allocated memory of the application is displayed by numbers and by a graph.

Objects

Below the allocated memory, the user finds a detailed tree of objects that are present in the application. The columns of the table contain the following object information:

Column	Description
Object	Hierarchical list of each object in the application. Shown is the ID of the object.
Id	AppWizard ID of the object as a hex number.
Handle	Window Manager handle number of the object.
x	X-position of the object.
y	Y-position of the object.
w	Width of the object.
h	Height of the object.
Visbl.	1 if the object is currently made visible, 0 if not.
Trans.	1 if the object has transparency (<code>WM_CF_HASTRANS</code> flag), 0 if not.
Enbl.	1 if the object is currently enabled, 0 if not.

Input

Any form of input (PID and keys) the application receives is shown here, as well as commands. If recording is active, this input will be saved to a file, so that it can be run at a later time. A more detailed description on the input and commands can be read further on under *Recording* on page 301.

Variables

All AppWizard variables are shown in the lower right corner of the dialog and showing their current value, as the application is running.

15.5 Recording

The AppWizard SPY makes it possible to record a running AppWizard application. A recording is saved in the project directory and can be played at a later time.

To start a recording, the application has to be running. A recording can be started by pressing the **Record** button.



Logging of user input

The following input information is logged:

- PID information (touch pressed and released)
- Key information (key pressed and released)

Commands during recording

The following commands can be issued while recording an application.

Command	Hotkey	Description
Request Variables	F9	Writes the current state of all project variables into a dedicated file.
Request Objects	F11	Writes the contents of the object table into a dedicated file.
Request Screenshot	F12	Takes a screenshot of the application.

File syntax

Note

The syntax used for `.AppRec` files is not final and could change in future versions!

Every input the application receives or command that is issued during a recording is saved in a `.AppRec` file. The following base syntax is used:

```
<Ticks>, <Command>(<Params>)
```

The following commands exist, with their corresponding parameters:

Command	Parameters	Description
SendPID	(x, y, Pressed)	PID state at a given position.
SendKEY	(Key, Pressed)	A given key has been pressed or released.
RequestVARIABLES	None.	Saves the state of all project variables in a file.
RequestOBJECTS	None.	Saves the data of the object table in a file.
RequestSCREEN	None.	Takes a screenshot of the application.

Directory structure and file naming

All SPY-related files of a project are located in the `spy` directory.

Every record file is saved in the following syntax format:

```
<YYYY>_<MM>_<DD>_<HH>_<MM>_<SS>_<MS>.AppRec
```

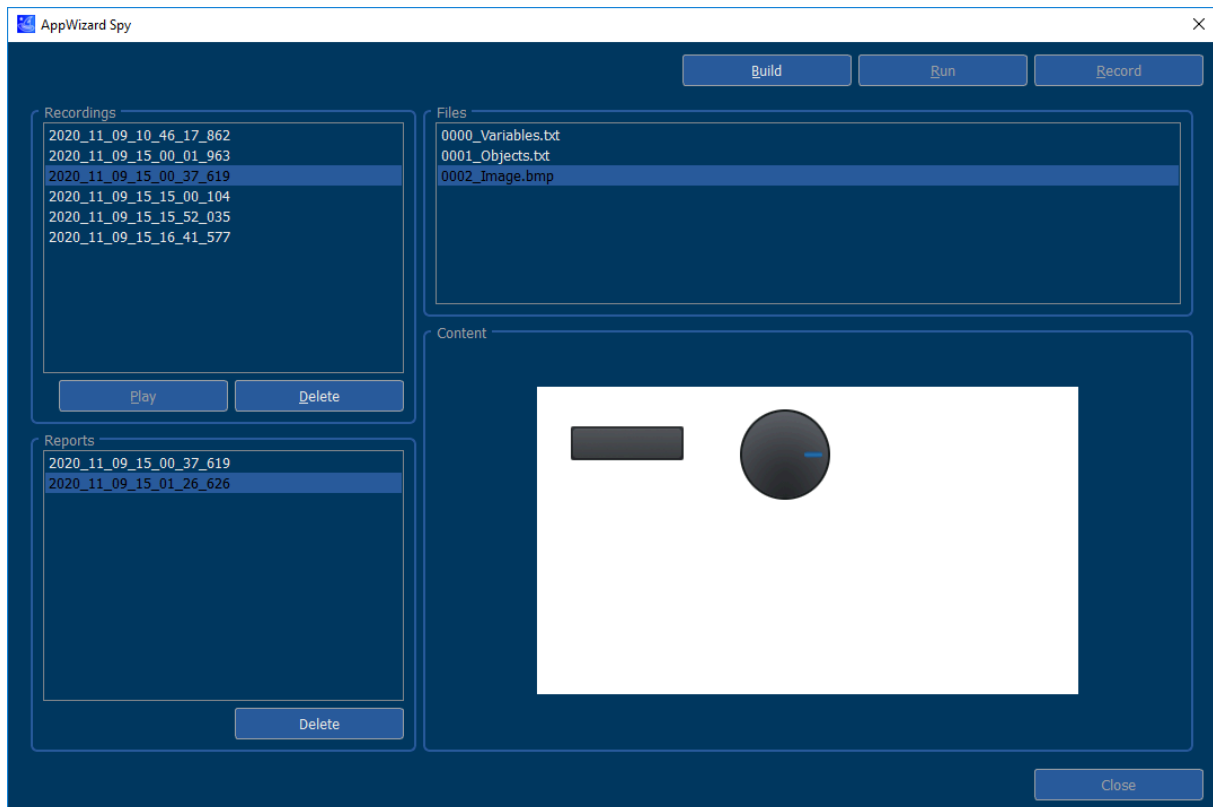
Example:

```
2020_11_09_10_46_17_862.AppRec
```

For each recording, a directory is created that contains any saved object or variable data, as well as screenshots of the application.

15.6 Playing a recording

To play an existing recording of a project, open the main AppWizard SPY dialog.



To play a recording, selected the desired recording and press the **Play** button.

The application will be executed and all the recorded data, such as PID, pressed keys, etc. will be applied to the application. Any exported data (e.g. screenshots) will be saved in a newly generated folder according to the current time in the same format as described above. This new sub-folder will be located in the main folder of the recording.

Chapter 16

Command line usage

AppWizard offers some commands that can be executed via the command prompt.

16.1 Command format

Commands are entered using the following format:

```
AppWizard.exe <ProjectName>.AppWizard <-command>
```

Example

The example below will open a given AppWizard project, export its code and then close AppWizard once the export is done.

```
AppWizard.exe ClimateControl.AppWizard -export -exit
```

16.2 Command line options

Command	Description
<code>-export</code>	Exports the code of a given AppWizard project.
<code>-exit</code>	Closes AppWizard.

Chapter 17

Glossary

BSP

Board support package.

embOS

[Embedded real-time operating system.](#)

emFile

[Embedded file system.](#)

emWin

[Embedded graphics library.](#)

Hierarchic tree

Widget on the left side of the AppWizard that displays the object hierarchy of the application.

MCU

Microcontroller unit.

Object

AppWizard equivalent of an emWin widget.

RAM

Random access memory.

ROM

Read-only memory.

SES

[SEGGER Embedded Studio.](#)

SPY

Tool for monitoring memory usage of the application. See *AppWizard SPY* on page 295.

WM_PAINT

Message sent to a window/widget by emWin's window manager. This message executes drawing operations of a window/widget.

WYSIWYG

What You See Is What You Get.